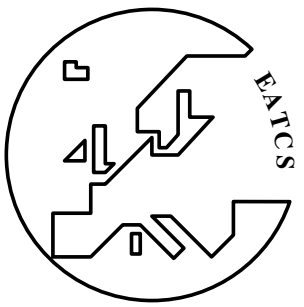
46th International Colloquium on Automata, Languages, and Programming

ICALP 2019, July 9–12, 2019, Patras, Greece

Edited by
Christel Baier
Ioannis Chatzigiannakis
Paola Flocchini
Stefano Leonardi



Editors

Christel Baier
TU Dresden, Germany
christel.baier@tu-dresden.de

Ioannis Chatzigiannakis
Sapienza University of Rome, Italy
ichatz@diag.uniroma1.it

Paola Flocchini
University of Ottawa, Canada
paola.flocchini@uottawa.ca

Stefano Leonardi
Sapienza University of Rome, Italy
leonardi@diag.uniroma1.it

ACM Classification 2012
Theory of computation

ISBN 978-3-95977-109-2

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at https://www.dagstuhl.de/dagpub/978-3-95977-109-2.

Publication date
July, 2019

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

Digital Object Identifier: 10.4230/LIPIcs.ICALP.2019.0

ISBN 978-3-95977-109-2 **ISSN 1868-8969** **https://www.dagstuhl.de/lipics**

LIPIcs – Leibniz International Proceedings in Informatics

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

ISSN 1868-8969

https://www.dagstuhl.de/lipics

Contents

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Track B: Automata, Logic, Semantics, and Theory of Programming

Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Preface

This volume contains the papers presented at ICALP 2019, the 46th edition of the International Colloquium on Automata, Languages and Programming, held in Patras, Greece during July 8–12, 2019. ICALP is a series of annual conferences of the European Association for Theoretical Computer Science (EATCS), which first took place in 1972. This year, the ICALP program consisted of three tracks:

- Track A: Algorithms, Complexity, and Games,
- Track B: Logic, Semantics, Automata and Theory of Programming,
- Track C: Foundations of Networked Computation: Models, Algorithms, and Information Management.

In response to the call for papers, a total 490 submissions were received: 316 for track A, 103 for track B, and 71 for track C. Each submission was assigned to at least three Program Committee members, aided by many subreviewers. Out of these, the committee decided to accept 146 papers for inclusion in the scientific program: 94 papers for Track A, 31 for Track B, and 21 for Track C. The selection was made by the Program Committees based on originality, quality, and relevance to theoretical computer science. The quality of the manuscripts was very high, and many deserving papers could not be selected.

The EATCS sponsored awards for both a best paper and a best student paper for each of the three tracks, selected by the Program Committees.

The best paper awards were given to the following papers:

- Track A: Bingkai Lin. "A Simple Gap-producing Reduction for the Parameterized Set Cover Problem".
- Track B: Christof Löding and Anton Pirogov. "Determinization of Büchi Automata: Unifying the Approaches of Safra and Muller-Schupp".
- Track C: Keren Censor-Hillel and Mikaël Rabie. "Distributed Reconfiguration of Maximal Independent Sets".

The best student paper awards, for papers that are solely authored by students, were given to the following papers:

- Track A: Joran van Apeldoorn & András Gilyén. "Improvements in Quantum SDP-Solving with Applications".
- Track B: Marie Fortin. "FO = FO3 for linear orders with monotone binary relations".

Apart from the contributed talks, ICALP 2019 included invited presentations by Michal Feldman, Martin Grohe, Ola Svensson, Frits Vaandrager and Mihalis Yannakakis. This volume of the proceedings contains all contributed papers presented at the conference together with the abstracts of the invited speakers.

The program of ICALP 2019 also included presentation of the EATCS Award 2019 to Thomas Henzinger, the Alonzo Church Award 2019 to Murdoch J. Gabbay and Andrew M. Pitts, the Presburger Award 2019 to Karl Bringmann and Kasper Green Larsen, and the EATCS Distinguished Dissertation Awards.

Four satellite events of ICALP were held on July 8th, 2019:

- Workshop on Theoretical Aspects of Fairness (WTAF)
- Parameterized Approximation Algorithms Workshop (PAAW)

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

- Workshop on Algorithmic Aspects of Temporal Graphs II
- Logic and Computational Complexity Workshop (LCC 2019)

We wish to thank all authors who submitted extended abstracts for consideration, the Program Committees for their scholarly effort, and all referees who assisted the Program Committees in the evaluation process. We are also grateful to the Conference Co-Chairs Sotiris Nikoletseas and Christos Zaroliagis and all the support staff of the Organizing Committee from the University of Patras and the Computer Technology Institute & Press "Diophantus" for organizing ICALP 2019.

We are grateful for generous support from University of Patras and the Department of Computer Engineering & Informatics for their support for the conference. We also thank the Center for Perspicuous Computing CPEC (supported by CPEC - TRR 248) for their support for the travelling costs of the invited speakers.

We would like to thank Anca Muscholl for her continuous support and Paul Spirakis, the president of EATCS, for his generous advice on the organization of the conference.

July 2019

Christel Baier
Ioannis Chatzigiannakis
Paola Flocchini
Stefano Leonardi

Organization

Program Committee

Track A

Stefano Leonardi	Sapienza University of Rome, Italy, Chair
Yossi Azar	Tel-Aviv University, Israel
Aaron Bernstein	Massachusetts Institute of Technology, United States
Sayan Bhattacharya	Duke University, United States
Karl Bringmann	Max Planck Institute for Informatics, Germany
Gerth Stølting Brodal	Aarhus University, Denmark
Jaroslaw Byrka	University of Wroclaw, Poland
Parinya Chalermsook	Aalto University, Finland
Paul Duetting	London School of Economics, United Kingdom
Uriel Feige	Weizmann Institute, Israel
Claudio Gentile	Google Research, United States
Mohsen Ghaffari	ETH Zurich, Switzerland
Antoine Joux	Fondation Partenariale de l'UPMC, IMJ-PRG, France
Telikepalli Kavitha	Tata Institute of Fundamental Research, Mumbai, India
Thomas Kesselheim	University of Bonn, Germany
Michal Koucky	Czech Academy of Sciences, Czechia
Alexander Kulikov	St. Petersburg Department of Steklov Institute of Mathematics, Russia
Sophie Laplante	IRIF, Université Paris Diderot Paris 7, France
Francois Le Gall	Kyoto University, Japan
Ramanujan M. Sridharan	The University of Warwick, United Kingdom
Evangelos Markakis	Athens University of Economics and Business, Greece
Renato Paes Leme	Google, United States
Marcin Pilipczuk	Institute of Informatics, University of Warsaw, Poland
Adi Rosén	CNRS and Université Paris Diderot, France
Eva Rotenberg	Technical University of Denmark, Denmark
Rahul Santhanam	University of Oxford, United Kingdom
Alessandra Scafuro	North Carolina State University, United States
Sandeep Sen	Dept of CSE, IIT Delhi, India
Francesco Silvestri	University of Padova, Italy
Paul Spirakis	University of Liverpool and University of Patras, Greece
Leen Stougie	Centrum voor Wiskunde en Informatica (CWI), Netherlands
Chaitanya Swamy	University of Waterloo, Canada
Stefan Szeider	Vienna University of Technology, Austria
Rico Zenklusen	ETH Zurich, Switzerland

Track B

Christel Baier	TU Dresden, Germany, Chair
Parosh Aziz Abdulla	Uppsala University, Sweden
Krishnendu Chatterjee	Institute of Science and Technology (IST), Austria
Thomas Colcombet	CNRS, France
Pedro R. D'Argenio	Universidad Nacional de Córdoba - CONICET, Argentina

Laure Daviaud	City, University of London, United Kingdom
Rocco De Nicola	IMT - School for Advanced Studies Lucca, Italy
Josee Desharnais	Laval University, Canada
Mariangiola Dezani-Ciancaglini	Dipartimento di Informatica, Università di Torino, Italy
Amina Doumane	PPS, France
Nathanaël Fijalkow	CNRS, LaBRI, University of Bordeaux, United Kingdom
Wan Fokkink	Vrije Universiteit Amsterdam, Netherlands
Christoph Haase	University of Oxford, United Kingdom
Ichiro Hasuo	National Institute of Informatics, Japan
Radha Jagadeesan	DePaul University, United States
Markus Lohrey	University of Siegen, Germany
P. Madhusudan	University of Illinois at Urbana-Champaign, United States
Radu Mardare	Aalborg University, Denmark
Matteo Mio	CNRS/ENS-Lyon, France
Mickael Randour	F.R.S.-FNRS & UMONS - Université de Mons, Belgium
Sven Schewe	University of Liverpool, United Kingdom
Lutz Schröder	Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
Helmut Seidl	Technical University of Munich, Germany
Michał Skrzypczak	University of Warsaw, Poland
Pawel Sobocinski	University of Southampton, United Kingdom
Christine Tasson	Laboratoire IRIF - Université Paris 7, France

Track C

Paola Flocchini	University of Ottawa, Canada, Chair
Amotz Bar-Noy	City University of New York, United States
Vittorio Biló	University of Salento, Italy
Bogdan Chlebus	University of Colorado Denver, United States
Xiaotie Deng	Peking University, China
Leszek Gasieniec	University of Liverpool, United Kingdom
Olga Nikolaevna Goussevskaia	Federal University of Minas Gerais, Brazil
Magnús Halldórsson	Reykjavik University, Iceland
Tobias Harks	Augsburg University, Germany
Christos Kaklamanis	University of Patras, Greece
Ralf Klasing	CNRS and University of Bordeaux, France
Max Klimm	Humboldt Universität zu Berlin, Germany
Ravi Kumar	Google, United States
Silvio Lattanzi	Google, Switzerland
Toshimitsu Masuzawa	Osaka University, Japan
Ruta Mehta	University of Illinois at Urbana-Champaign, United States
Ariel Orda	Department of Electrical Engineering, Technion, Israel
Gopal Pandurangan	University of Houston, United States
Pino Persiano	Università degli Studi di Salerno, Italy
Maria Potop-Butucaru	UPMC Sorbonne Universités, LIP6, Paris, France
Giuseppe Prencipe	Università di Pisa, Italy
Andrea Richa	Arizona State University, United States
Nicola Santoro	Carleton University, Canada
Jukka Suomela	Aalto University, Finland
Patrick Thiran	Ecole Polytechnique Fédérale de Lausanne, Switzerland
Peter Widmayer	ETH Zurich, Switzerland, Switzerland

Organizing Committee

Efstratios Gallopoulos	University of Patras, Greece
John Garofalakis	University of Patras and CTI, Greece
Christos Kaklamanis	University of Patras and CTI, Greece
Sotiris Nikoletseas	University of Patras and CTI, Greece, Conference Co-Chair
Christos Zaroliagis	University of Patras and CTI, Greece, Conference Co-Chair

Steering Committee

Javier Esparza	TUM Munich, Germany
Leslie Ann Goldberg	Oxford University, UK
Thore Husfeldt	Lund University, Sweden and IT University of Copenhagen, Denmark
Giuseppe Italiano	Università di Roma Tor Vergata, Italy
Christos Kaklamanis	University of Patras and CTI Diophantus, Greece
Daniel Marx	Hungarian Academy of Sciences, Hungary
Emanuela Merelli	University of Camerino, Italy
Anca Muscholl	Bordeaux University, France), Steering Committee Chair
Luke Ong	Oxford University, UK
Jiří Sgall	Charles University, Prague, Czech Rep.
Paul Spirakis	University of Liverpool, UK and University of Patras, Greece

Financial Sponsors

University of Patras, Greece
Center for Perspicuous Computing CPEC, Germany

Additional Reviewers

Abboud Amir
Abrahamsen Mikkel
Accattoli Beniamino
Adamczyk Marek
Aduri Pavan
Aggarwal Divesh
Agrawal Akanksha
Ahmadian Sara
Ahrens Benedikt
Ailon Nir
Ajwani Deepak
Akhremtsev Yaroslav
Akrida Eleni C.
Albert Michael
Aldridge Matthew
Allender Eric
Allouah Amine
Almagor Shaull
Alman Josh
Amanatidis Georgios
Ambainis Andris
Amy Matthew
An Hyung-Chan
Anagnostopoulos Aris
Angelidakis Haris
Annamalai Chidambram
Applebaum Benny
Arseneva Elena
Arunachalam Srinivasan
Asadpour Arash
Ashtiani Hassan
Assadi Sepehr
Atserias Albert
Augustine John
Aumüller Martin
Avin Chen
Avner Orly
Awasthi Pranjal
Aziz Haris
Babenko Maxim
Bacci Giorgio
Bacci Giovanni
Backurs Arturs
Badanidiyuru Varadaraja Ashwinkumar
Balkanski Eric
Balko Martin
Bampas Evangelos
Banik Aritra
Bansal Nikhil
Barbero Fausto
Barenbaum Pablo
Barmpalias George
Bartholdi Laurent
Barto Libor
Bartoletti Massimo
Baswana Surender
Batra Jatin
Becher Veronica
Beffara Emmanuel
Behnezhad Soheil
Beimel Amos
Belovs Aleksandrs
Ben-Amram Amir
Ben-David Shalev
Benadè Gerdus
Bera Debajyoti
Berkholz Christoph
Bertrand Nathalie
Bhagat Subhash
Bhangale Amey
Bhaskara Aditya
Bhattacharya Anup
Bhattiprolu Vijay
Bilò Davide
Bishnu Arijit
Bitansky Nir
Björklund Andreas
Blahoudek František
Bliznets Ivan
Blocq Gideon
Blondin Michael
Blumensath Achim
Bodlaender Hans L.
Bodwin Greg
Boker Udi
Bollig Benedikt
Bonchi Filippo
Boodaghians Shant
Boyle Elette
Brakensiek Joshua
Brand Cornelius
Brandt Sebastian
Brázdil Tomáš
Bredereck Robert
Broadbent Anne
Broutin Nicolas
Bruni Roberto
Bu Gewu
Buchbinder Niv
Buttkus Matthias
Cabello Sergio
Cadilhac Michaël
Canonne Clément
Capelli Florent
Caragiannis Ioannis
Carton Olivier
Cassuto Yuval
Castellan Simon
Cazaux Bastien
Ceccarello Matteo
Censor-Hillel Keren
Chailloux André
Chakrabarti Amit
Chakrabarti Shouvanik
Chakrabarty Deeparnab
Chakraborty Shantanav
Chan T-H. Hubert
Chan Timothy M.
Chang Yi-Jun
Charatonik Witold
Chatterjee Soumyottam
Chattopadhyay Eshan
Chechik Shiri
Chen Ho-Lin
Chen Lijie
Chen Yixin
Chen Yu
Chen Yu-Fang
Cheraghchi Mahdi
Chillara Suryajith
Chistikov Dmitry
Chitnis Rajesh
Chonev Ventsislav
Choudhary Keerti
Chouquet Jules
Christiani Tobias
Chuzhoy Julia
Clifford Raphael
Cohen-Addad Vincent
Coja-Oghlan Amin
Colini Baldeschi Riccardo
Corsten Jan
Cseh Ágnes
Czerwiński Wojciech
Dabrowski Konrad Kazimierz
Dal Lago Ugo
Dalmau Victor
Damian Mirela
Dani Varsha
Dartois Luc
Das Debarati
Das Syamantak
Datta Samir
Daymude Joshua
De Minati
Dehornoy Patrick
Delic Dejan
Deligkas Argyrios
Dell Holger
Deng Xiaotie
Dereniowski Dariusz
Deshpande Apoorvaa
Di Luna Giuseppe Antonio
Diakonikolas Jelena
Diaz Josep
Dickens Charlie
Doerr Benjamin
Doty David
Dou Zehao
Douéneau-Tabot Gaëtan
Doyen Laurent

Drineas Petros
Drmota Michael
Duan Ran
Dudycz Szymon
Dujmovic Vida
Dulek Yfke
Dyer Martin
Earl Roberson David
Efraimidis Pavlos
Efthymiou Charilaos
Elder Murray
Emek Yuval
Ene Alina
Eppstein David
Epstein Leah
Erlebach Thomas
Evald Jacob
Faenza Yuri
Fahrbach Matthew
Fakcharoenphol Jittat
Fanelli Angelo
Faonio Antonio
Farhadi Alireza
Feier Cristina
Feldman Michal
Feldman Moran
Felsner Stefan
Ferraioli Diodato
Fervari Raul
Figueira Santiago
Filmus Yuval
Fineman Jeremy
Fischer Carsten
Fischer Manuela
Fogarty Seth
Forster Sebastian
Fortin Marie
Fortnow Lance
Fox Kyle
Freksen Casper Benjamin
Friggstad Zachary
Fulla Peter
Furber Robert
Gadekar Ameet
Gagie Travis
Galanis Andreas
Galesi Nicola
Galletta Lillo
Gálvez Waldo
Gamlath Buddhima
Ganardi Moses
Gańczorz Michał
Ganguly Sumit
Garg Deepak
Garg Mohit
Gawrychowski Pawel
Ghica Dan
Ghorbal Khalil
Ghosh Arijit
Gimbert Hugo
Gmyr Robert
Gogacz Tomasz
Goharshady Amir Kafshadr
Goldberg Leslie Ann
Göller Stefan
Golovnev Alexander
Gopi Sivakanth
Goranci Gramoz
Gouleakis Themis
Grandoni Fabrizio
Green Larsen Kasper
Grenet Bruno
Grochow Joshua
Grohe Martin
Grossi Roberto
Gualà Luciano
Gudmundsson Joachim
Guerrieri Giulio
Guillon Pierre
Guo Heng
Gupta Anupam
Gupta Manoj
Gur Tom
Habermehl Peter
Hagerup Torben
Hamoudi Yassine
Hampson Christopher
Hanzlik Lucjan
Haque Abida
Harris David
Harsha Prahladh
Harvey Nick
Hatami Hamed
Heindel Tobias
Hendrian Diptarama
Hernández Vélez César
Hirahara Shuichi
Hirai Hiroshi
Hirvonen Åsa
Hirvonen Juho
Hitchcock John
Hoefer Martin
Hoeksma Ruben
Hofman Piotr
Holland Joshua
Holm Jacob
Horne Ross
Hosseini Kaave
Hoyrup Mathieu
Hsu Justin
Huang Chien-Chung
Hubáček Pavel
Huisman Marieke
Hunkenschröder Christoph
Husfeldt Thore
Ilcinkas David
Im Sungjin
Ismaili Anisse
Istrate Gabriel
Ivanyos Gabor
Ivkin Nikita
Jaiswal Ragesh
Jansen Klaus
Jayram T.S.
Jeandel Emmanuel
Jerrum Mark
Jeż Artur
Jeż Łukasz
Jiamjitrak Wanchote
Jiang Shaofeng
Jindal Gorav
Jordan Charles
Kalaitzis Christos
Kamali Shahin
Kamath Akshay
Kamma Lior
Kammer Frank
Kanj Iyad
Kannan Sampath
Kapralov Michael
Karczmarz Adam
Karpov Nikolai
Karrenbauer Andreas
Karthik C. S.
Katoen Joost-Pieter
Kawamura Akitoshi
Kazda Alexandr
Kelk Steven
Kempa Dominik
Khan Arindam
Khanna Sanjeev
Kiefer Stefan
Kim Eunjung
Kincaid Zachary
Kjos-Hanssen Bjørn
Klin Bartek
Klonowski Marek
Kobayashi Koji M.
Koivisto Mikko
Kokainis Martins
Kolesnichenko Ignat
Kolla Alexandra
Kollias Kostas
König Barbara
Konrad Christian
Kopelowitz Tsvi
Korhonen Janne H.
Korman Amos
Kothapalli Kishore
Kothari Robin
Koutecky Martin
Kowalik Łukasz
Kowalski Darek
Kozik Marcin
Kozma Laszlo
Kragl Bernhard
Král Karel

Kranakis Evangelos
Kraska Artur
Kretinsky Jan
Krokhin Andrei
Krysta Piotr
Kučera Antonín
Kufleitner Manfred
Kulkarni Janardhan
Kulkarni Pooja
Kulkarni Rucha
Kumar Amit
Künnemann Marvin
Kuperberg Denis
Kurpisz Adam
Kyng Rasmus
Laarhoven Thijs
Łącki Jakub
Laekhanukit Bundit
Lagerqvist Victor
Lamprou Ioannis
Lange Julien
Lanvin Victor
Lasota Sławomir
Laurent Monique
Lauria Massimo
Lazic Ranko
Lee David
Lee Euiwoong
Lehtinen Karoliina
Lengler Johannes
Lenzner Pascal
Leroux Jérôme
Leucci Stefano
Levy Jordi
Lewandowski Mateusz
Lhote Nathan
Li Jason
Li Shi
Li Yi
Lianeas Thanasis
Lin Bingkai
Lin Huijia
Liu Chih-Hung
Liu Yang
Livanos Vasilis
Löding Christof
Loff Bruno
Lohrey Markus
Lonsing Florian
Loreti Michele
Lovett Shachar
Lu Yuxuan
Lutz Jack H.
Mahadev Urmila
Mai Tung
Makarychev Konstantin
Makriyannis Nikolaos
Malyshev Dmitriy
Mamageishvili Akaki
Maneth Sebastian
Mansfield Shane
Manuel Amaldev
Manurangsi Pasin
Marcinkowski Jan
Marino Andrea
Markey Nicolas
Markou Euripides
Martin Barnaby
Martini Simone
Masařík Tomáš
Maslov Dmitri
Masopust Tomas
Mastrolilli Monaldo
Mathieu Claire
Mauras Simon
Mazowiecki Filip
McCauley Samuel
McGregor Andrew
Mehlhorn Kurt
Mehraban Saeed
Meirom Eli
Melissourgos Themistoklis
Merker Martin
Merkle Wolfgang
Mertzios George
Mery Daniel
Meyer Ulrich
Michail Othon
Michielini Vincent
Mihalák Matúš
Mikulas Szabolcs
Milovanov Alexey
Miltzow Till
Misra Neeldhara
Misra Pranabendu
Mitrovic Slobodan
Mitsou Valia
Mnich Matthias
Molla Anisur Rahaman
Molter Hendrik
Mömke Tobias
Monaco Gianpiero
Monien Burkhard
Montanaro Ashley
Morimae Tomoyuki
Moscardelli Luca
Moses Jr. William K.
Mottet Antoine
Mucha Marcin
Murlak Filip
Musco Cameron
Muzi Irene
Nagarajan Viswanath
Nandy Subhas
Naranayan Anand Kumar
Nederlof Jesper
Nelson Jelani
Neumann Stefan
Nies Andre
Nikoletseas Sotiris
Nolin Alexandre
Norman Gethin
Nusser André
O'Donnell Ryan
Ochremiak Joanna
Odor Gergely
Ohlmann Pierre
Okhotin Alexander
Okudono Takamasa
Oliveira Igor Carboni
Olivetti Dennis
Olver Neil
Omri Eran
Onak Krzysztof
Otachi Yota
Otop Jan
Oualhadj Youssouf
Paat Joseph
Padberg Julia
Padovani Luca
Padro Carles
Pagano Miguel
Pagh Rasmus
Pajak Dominik
Paluch Katarzyna
Pananjady Ashwin
Pandurangan Chandrasekharan
Panigrahi Debmalya
Panolan Fahad
Paperman Charles
Parikh Rohit
Parotsidis Nikos
Parter Merav
Parys Paweł
Paschos Vangelis
Patel Viresh
Patt-Shamir Boaz
Paul Christophe
Paulusma Daniel
Paz Ami
Pedersen Mathias Ruggaard
Penelle Vincent
Penna Paolo
Peressotti Marco
Perkins Will
Peters Kirstin
Petrisan Daniela
Petrucciani Tommaso
Piedeleu Robin
Pilipczuk Michał
Pinsker Michael
Pisanti Nadia
Pissis Solon
Place Thomas
Platzer André
Podolskii Vladimir
Popa Alexandru

Potapov Igor
Potechin Aaron
Pouly Amaury
Pous Damien
Pradic Pierre
Prakash Anupam
Pratap Rameshwar
Praveen M.
Probst Maximilian
Pruekprasert Sasinee
Pudlak Pavel
Puppis Gabriele
Purohit Manish
Qiao Youming
Quatmann Tim
Rabie Mikaël
Radhakrishnan Jaikumar
Radzik Tomasz
Rafiey Arash
Raghvendra Sharath
Raichel Benjamin
Raman Rajiv
Raman Venkatesh
Rampersad Narad
Raptopoulos Christoforos
Raskin Jean-Francois
Rasmussen Peter Michael Reichstein
Rathke Julian
Ravi R
Rawitz Dror
Ray Saurabh
Ray Chaudhury Bhaskar
Raymond Jean-Florent
Razgon Igor
Reiter Fabian
Riba Colin
Ricciotti Wilmer
Robinson Peter
Rodaro Emanuele
Roditty Liam
Rogers Ryan
Rojas Cristobal
Romashchenko Andrei
Romero Orth Miguel
Ron Dana
Rosa-Velardo Fernando
Rösner Clemens
Rossmanith Peter
Rote Günter
Roth Marc
Rothvoss Thomas
Roy Arnab
Roytman Alan
Rubin Natan
Rubinstein Aviad
Rzążewski Paweł
Saberi Amin
Sachdeva Sushant
Sadakane Kunihiko
Saivasan Prakash
Sajenko Andrej
Sala Pietro
Sandeep R.B.
Sanders Peter
Sanita Laura
Santos de Lima Murilo
Saranurak Thatchaphol
Saurabh Saket
Savani Rahul
Sawa Zdeněk
Schewior Kevin
Schmid Andreas
Schmid Laura
Schmidt Paweł
Schmitz Sylvain
Schmude Janusz
Schneider Jon
Schöpp Ulrich
Schwartz Roy
Schwiegelshohn Chris
Sebastien Labbe
Seddighin Saeed
Seeber Jens
Segoufin Luc
Selivanova Svetlana
Sen Sandeep
Serna Maria
Serrano Llerena Yamilet R.
Seshadhri C.
Seto Kazuhisa
Sgall Jiří
Sgouritsa Alkmini
Shahrasbi Amirbehshad
Shallit Jeffrey
Shalom Mordechai
Shayeghi Ala
Shen Alexander
Shi Jonathan
Shirmohammadi Masha
Siala Mohamed
Sickert Salomon
Sidford Aaron
Sidiropoulos Anastasios
Siebertz Sebastian
Sikora Jamie
Simon Hans
Simonov Kirill
Sitters Rene
Sivan Balasubramanian
Skopalik Alexander
Skretas George
Smal Alexander
Sokolov Dmitry
Sokolova Ana
Solomon Shay
Sorge Manuel
Sornat Krzysztof
Soto José A.
Spoerhase Joachim
Srinathan Kannan
Srinivasan Srikanth
Srivastava Piyush
Stachowiak Grzesiek
Stamatiou Yannis
Starikovskaya Tatiana
Starnberger Martin
Staton Sam
Stefanesco Léo
Stein Clifford
Sudo Yuichi
Sun Yihan
Sundaram Aarthi
Suresh Ananda Theertha
Syed Mohammad Meesum
T. Vasconcelos Vasco
Talbot Jean-Marc
Talebanfard Navid
Talmon Nimrod
Tamaki Suguru
Tamo Itzhak
Tamuz Omer
Tang Zhihao Gavin
Tavenas Sébastien
Teng Yifeng
Tesson Pascal
Thapen Neil
Thapper Johan
Thoma Daniel
Tini Simone
Tönnis Andreas
Tonoyan Tigran
Torres Vieira Hugo
Toth Csaba
Totzke Patrick
Touitou Noam
Traub Vera
Tribastone Mirco
Tsai Ming-Hsien
Tsakalidis Konstantinos
Tsikiridis Artem
Tulsiani Madhur
Turan Gyorgy
Turrini Andrea
Tzameret Iddo
Uitto Jara
Umboh Seeun William
Uniyal Sumedha
Urabe Natsuki
Uramoto Takeo
Uznański Przemysław
Vaikuntanathan Vinod
Vainstein Danny
Vakilian Ali
van Ee Martijn
van Iersel Leo
van Leeuwen Erik Jan
Van Oostrom Vincent

van Stee Rob
Vandin Fabio
Vargaftik Shay
Vargas Koch Laura
Vassilevska Williams Virginia
Vassilvitskii Sergei
Vaz Daniel
Vaze Rahul
Velan Dominik
Veltri Niccolò
Venturi Daniele
Vergnaud Damien
Vetta Adrian
Viglietta Giovanni
Vigny Alexandre
Villagra Marcos
Vinci Cosimo
Vinyals Marc
Vladu Adrian
von Gleissenthall Klaus
Vondrak Jan
Vorotnikova Sofya
Voudouris Alexandros
Vredeveld Tjark
Vusirikala Satyanarayana
Waga Masaki
Wahlström Magnus
Wang Joshua
Warode Philipp
Węgrzycki Karol
Weil Pascal
Weimann Oren
Wein Nicole
Wellnitz Philip
Weltge Stefan
Wiese Andreas
Wille Robert
Williams Ryan
Wimmer Karl
Winter Sarah
Wlodarczyk Michal
Worrell James
Wrochna Marcin
Wrona Michał
Wu David
Wu Steven
Wulff-Nilsen Christian
Xiao Mingyu
Xu Chao
Yamauchi Yukiko
Yao Penghui
Yingchareonthawornchai Sorrachai
Zamaraev Viktor
Zampetakis Manolis
Zandieh Amir
Zanuttini Bruno
Zarei Alireza
Zehavi Meirav
Zetzsche Georg
Zeume Thomas
Zhan Naijun
Zhong Fangwei
Zhou Linfeng
Zhou Zixin
Zhu Shufang
Živný Stanislav

List of Authors

Scott Aaronson (6)
Department of Computer Science, University of Texas at Austin, USA

Amir Abboud (7, 8)
IBM Almaden Research Center, California, USA

Mikkel Abrahamsen (9)
BARC, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen, Denmark

Bader Abu Radi (100)
School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel

Peyman Afshani (10)
Computer Science Department, Aarhus University, Denmark

Akanksha Agrawal (11)
Ben-Gurion University of the Negev, Beersheba, Israel

Eleni C. Akrida (131)
Department of Computer Science, University of Liverpool, UK

Noga Alon (12)
Department of Mathematics, Princeton University, Princeton, NJ 08544, USA; Schools of Mathematics and Computer Science, Tel Aviv University, Tel Aviv 69978, Israel

Aris Anagnostopoulos (136)
Sapienza University of Rome, Italy

Bertie Ancona (13)
MIT, Cambridge, MA, USA

Alexandr Andoni (14, 15)
Columbia University, New York City, NY, USA

Anuja Meetoo Appavoo (149)
Department of Computer Science, National University of Singapore

Srinivasan Arunachalam (16)
Center for Theoretical Physics, MIT, Cambridge, MA, USA

Sepehr Assadi (17)
Department of Computer Science, Princeton University, NJ, USA

Pranjal Awasthi (18)
Rutgers University, Piscataway, NJ, USA

Kyriakos Axiotis (19)
MIT, Cambridge, MA, USA

Mohamed-Amine Baazizi (101)
Sorbonne Université, CNRS, LIP6 UMR 7606, Paris, France

Ainesh Bakshi (18)
Carnegie Mellon University, Pittsburgh, PA, USA

Nikhil Balaji (102)
University of Oxford, UK

Maria-Florina Balcan (18)
Carnegie Mellon University, Pittsburgh, PA, USA

Pablo Barceló (103, 104)
Department of Computer Science, University of Chile, Santiago, Chile; IMFD, Santiago, Chile

Yair Bartal (20)
Department of Computer Science, Hebrew University of Jerusalem, Israel

David Basin (127)
Department of Computer Science, ETH Zürich, Universitätstrasse 6, 8092, Switzerland

Paul C. Bell (105)
Department of Computer Science, Byrom Street, Liverpool John Moores University, Liverpool, L3-3AF, UK

Giulia Bernardini (21)
Department of Informatics, Systems and Communication, University of Milano - Bicocca, Italy

Ivona Bezáková (22)
Department of Computer Science, Rochester Institute of Technology, Rochester, NY, USA

Therese Biedl (23)
David R. Cheriton School of Computer Science, University of Waterloo, Canada

Andreas Björklund (24, 25, 26)
Lund University, Lund, Sweden

Mikołaj Bojańczyk (106)
Institute of Informatics, University of Warsaw, Poland

Matthias Bonne (132)
Department of Computer Science, Technion, Haifa, Israel

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Fernando G. S. L. Brandão (27)
Institute of Quantum Information and Matter, California Institute of Technology, USA

Raimundo Briceño (29)
School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel

Paul Brunet (107)
University College London, UK

Andrei A. Bulatov (29)
School of Computing Science, Simon Fraser University, Canada

Mark Bun (30)
Simons Institute for the Theory of Computing, Berkeley, CA, USA; Boston University, MA, USA

Angel A. Cantu (31)
Department of Computer Science, University of Texas - Rio Grande Valley, USA

Ioannis Caragiannis (133)
University of Patras & CTI "Diophantus", Patras, Greece

Titouan Carette (108)
Université de Lorraine, CNRS, Inria, LORIA, F 54000 Nancy, France

Katrin Casel (109)
Hasso Plattner Institute, University of Potsdam, Germany

Arnaud Casteigts (134)
LaBRI, Université de Bordeaux, CNRS, Bordeaux INP, France

Keren Censor-Hillel (132, 135)
Department of Computer Science, Technion, Haifa, Israel

Shantanav Chakraborty (33)
QuIC, Université libre de Bruxelles, Belgium

Sourav Chakraborty (16)
Indian Statistical Institute, Kolkata, India

Jérémie Chalopin (34)
CNRS, Aix-Marseille Université, Université de Toulon, LIS, Marseille, France

Arkadev Chattopadhyay (35)
School of Technology and Computer Science, Tata Institute of Fundamental Research, Mumbai, India

Shiri Chechik (12)
Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel

Xue Chen (36)
Northwestern University, Evanston, IL, USA

Kuan Cheng (37)
Department of Computer Science, Johns Hopkins University, USA

Siu-Wing Cheng (38)
Department of Computer Science and Engineering, HKUST, Hong Kong

Victor Chepoi (34)
Aix-Marseille Université, CNRS, Université de Toulon, LIS, Marseille, France

Mahdi Cheraghchi (39)
Department of Computing, Imperial College London, London, UK

Eden Chlamtáč (40)
Ben Gurion University of the Negev, Beersheva, Israel

Laura Ciobanu (110)
Heriot-Watt University, Edinburgh EH14 4AS, Scotland

Ilan R. Cohen (136)
CWI, Amsterdam, The Netherlands

Sarel Cohen (12)
Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel

Vincent Cohen-Addad (41, 42)
CNRS & Sorbonne Université, Paris, France

Amin Coja-Oghlan (43)
Goethe University, Frankfurt, Germany

Alexandru Cojocaru (6)
School of Informatics, University of Edinburgh, UK

Dario Colazzo (101)
Université Paris-Dauphine, PSL, LAMSADE, France

Thomas Colcombet (44)
IRIF, CNRS, Université Paris Diderot, France

Graham Cormode (45)
University of Warwick, UK

Jurek Czyzowicz (137)
Université du Québec en Outaouais, Gatineau, Québec, Canada

Ugo Dal Lago (111)
University of Bologna, Italy; INRIA Sophia Antipolis, France

Mina Dalirrooyfard (46, 47)
MIT, Cambridge, MA, USA

Víctor Dalmau (29)
Department of Information and Communication Technologies, Universitat Pompeu Fabra, Barcelona, Spain

Jacques Dark (45)
University of Warwick, UK

Anuj Dawar (112)
University of Cambridge, UK

Joel D. Day (109)
Department of Computer Science, Loughborough University, UK

Ronald de Wolf (16)
QuSoft, CWI and University of Amsterdam, The Netherlands

Argyrios Deligkas (138)
Department of Computer Science, University of Liverpool, Liverpool, UK; Leverhulme Research Centre for Functional Materials Design, Liverpool, UK

Holger Dell (113)
Cluster of Excellence (MMCI), Saarland Informatics Campus (SIC), Saarbrücken, Germany

Michael Dinitz (40)
Johns Hopkins University, Baltimore, MD, USA

Stefan Dobrev (139)
Institute of Mathematics, Slovak Academy of Sciences, Bratislava, Slovakia

Dani Dorfman (114)
Blavatnik School of Computer Science, Tel Aviv University, Israel

Ran Duan (48)
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China

Guillaume Ducoffe (49)
National Institute for Research and Development in Informatics, Romania; The Research Institute of the University of Bucharest ICUB, Romania; University of Bucharest, Romania

Josep Díaz (50)
Computer Science Department, Universitat Politècnica de Catalunya, Barcelona

Julian Dörfler (51)
Saarland University, Saarbrücken, Germany

Talya Eden (52)
Tel Aviv University, Tel Aviv, Israel

Murray Elder (110)
University of Technology Sydney, Ultimo NSW 2007, Australia

Yuval Emek (140)
Faculty of Industrial Engineering and Management, Technion - IIT, Haifa, Israel

Alina Ene (53, 54)
Department of Computer Science, Boston University, MA, USA

Massimo Equi (55)
Department of Computer Science, University of Helsinki, Finland

Thomas Erlebach (141)
Department of Informatics, University of Leicester, Leicester, England

Kousha Etessami (115)
School of Informatics, University of Edinburgh, UK

Nova Fandina (20)
Department of Computer Science, Hebrew University of Jerusalem, Israel

Angelo Fanelli (133)
CNRS (UMR-6211), Caen, France

John Fearnley (56, 138)
University of Liverpool, UK

Michal Feldman (1)
Blavatnik School of Computer Science, Tel-Aviv University, Israel

Miron Ficak (57)
Theoretical Computer Science Department, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland

Diego Figueira (104)
CNRS & LaBRI, Talence, France

Yuval Filmus (35, 58)
Department of Computer Science, Technion Israel Institute of Technology, Haifa, Israel

Pamela Fleischmann (109)
Department of Computer Science, Kiel University, Germany

Fedor V. Fomin (11, 59, 60)
University of Bergen, Bergen, Norway

Marie Fortin (116)
LSV, CNRS & ENS Paris-Saclay, Université Paris-Saclay, France

Casper Benjamin Freksen (10)
Computer Science Department, Aarhus University, Denmark

Tobias Friedrich (61)
Algorithm Engineering Group, Hasso Plattner Institute, University of Potsdam, Germany

Andreas Galanis (22)
Department of Computer Science, University of Oxford, UK

Ankit Garg (62)
Microsoft Research India, Bangalore, India

Naveen Garg (63)
Computer Science and Engineering Department, Indian Institute of Technology, Delhi, India

Francesco Gavazzo (111)
IMDEA Software Institute, Spain

Dmitry Gavinsky (64)
Institute of Mathematics, Czech Academy of Sciences, 115 67 Žitna 25, Praha 1, Czech Republic

Paweł Gawrychowski (21)
Institute of Computer Science, University of Wrocław, Poland

Oliver Gebhard (43)
Goethe University, Frankfurt, Germany

Loukas Georgiadis (7)
University of Ioannina, Greece

Konstantinos Georgiou (137)
Department of Mathematics, Ryerson University, Toronto, Ontario, Canada

Mohsen Ghaffari (142)
ETH Zurich, Switzerland

Giorgio Ghelli (101)
Dipartimento di Informatica, Università di Pisa, Italy

Alexandru Gheorghiu (6)
Department of Computing and Mathematical Sciences, California Institute of Technology, USA; School of Informatics, University of Edinburgh, UK

Panos Giannopoulos (9)
giCenter, Department of Computer Science, City University of London, EC1V 0HB, London, UK

Seth Gilbert (149)
Department of Computer Science, National University of Singapore

András Gilyén (33, 99)
QuSoft/CWI, The Netherlands

Leslie Ann Goldberg (22)
Department of Computer Science, University of Oxford, UK

Paul W. Goldberg (65)
Department of Computer Science, University of Oxford, United Kingdom

Petr A. Golovach (59)
Department of Informatics, University of Bergen, Norway

Alexander Golovnev (66)
Harvard University, Cambridge, USA

Spencer Gordon (56)
California Institute of Technology, Pasadena, CA, USA

Alex B. Grilo (28)
CWI, Amsterdam, The Netherlands; QuSoft, Amsterdam, The Netherlands

Martin Grohe (2, 117)
RWTH Aachen University, Aachen, Germany

Roberto Grossi (55)
Dipartimento di Informatica, Università di Pisa, Italy

Erich Grädel (112)
RWTH Aachen University, Germany

Anupam Gupta (42, 63, 67)
Carnegie Mellon University, Pittsburgh, PA, USA

Nikhil Gupta (62)
Department of Computer Science and Automation, Indian Institute of Science, India

Siddharth Gupta (143)
Ben-Gurion University of the Negev, Israel

Guru Guruganesh (67)
Google Research, United States

Venkatesan Guruswami (68, 92)
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

Bernhard Haeupler (144)
Carnegie Mellon University, Pittsburgh, PA, USA

Max Hahn-Klimroth (43)
Goethe University, Frankfurt, Germany

Lianna Hambardzumyan (58)
School of Computer Science, McGill University, Montreal, QC, Canada

Yassine Hamoudi (69)
Université de Paris, IRIF, CNRS, F-75013 Paris, France

Samuel Haney (70)
Duke University, Durham, NC, USA

Hamed Hatami (58)
School of Computer Science, McGill University, Montreal, QC, Canada

Pooya Hatami (58)
Department of Computer Science, UT Austin, Austin, TX, USA

Falko Hegerfeld (71)
Humboldt-Universität zu Berlin, Germany

Monika Henzinger (13)
University of Vienna, Austria

Alexandros Hollender (65)
Department of Computer Science, University of Oxford, United Kingdom

Chih-Duo Hong (103)
Department of Computer Science, University of Oxford, UK

Mehran Hosseini (118)
Department of Computer Science, University of Oxford, UK

Pavel Hrubeš (72)
Institute of Mathematics of ASCR, Prague

Zhiyi Huang (73)
The University of Hong Kong

Christian Ikenmeyer (51)
Max Planck Institute for Software Systems, Saarbrücken, Germany

Rahul Ilango (66)
Rutgers University, New Brunswick, USA

Sungjin Im (145)
University of California at Merced, USA

Russell Impagliazzo (66)
University of California San Diego, USA

Giuseppe F. Italiano (7)
LUISS University, Rome, Italy

Klaus Jansen (74, 75)
Department of Computer Science, Christian-Albrechts-Universität, Kiel, Germany

Emmanuel Jeandel (108)
Université de Lorraine, CNRS, Inria, LORIA, F 54000 Nancy, France

Stacey Jeffery (33)
QuSoft/CWI, The Netherlands

Ce Jin (48, 76)
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China

Lingfei Jin (68)
Shanghai Key Laboratory of Intelligent Information Processing, School of Computer Science, Fudan University, Shanghai, China; Shanghai Institute of Intelligent Electronics & Systems, Shanghai, China; Shanghai Bolckchain Engineering Research Center, Fudan University, Shanghai 200433, China

Zhengzhong Jin (37)
Department of Computer Science, Johns Hopkins University, USA

Valentine Kabanets (39, 66)
Simon Fraser University, Burnaby, Canada

Amir Kalev (27)
Joint Center for Quantum Information and Computer Science, University of Maryland, USA

Lior Kamma (10)
Computer Science Department, Aarhus University, Denmark

Frank Kammer (141)
THM, University of Applied Sciences Mittelhessen, Giessen, Germany

Haim Kaplan (114)
Blavatnik School of Computer Science, Tel Aviv University, Israel

Elham Kashefi (6)
School of Informatics, University of Edinburgh, UK; CNRS LIP6, Université Pierre et Marie Curie, Paris, France

Petteri Kaski (26)
Department of Computer Science, Aalto University, Finland

Neeraj Kayal (62)
Microsoft Research India, Bangalore, India

Sandra Kiefer (106, 117)
Department of Computer Science, RWTH Aachen University, Germany

Stefan Kiefer (102, 119)
University of Oxford, UK

Ryan Killick (137)
School of Computer Science, Carleton University, Ottawa, Ontario, Canada

Philipp Kindermann (23)
Lehrstuhl für Informatik I, Universität Würzburg, Germany

Lefteris Kirousis (50)
Department of Mathematics, National and Kapodistrian University of Athens; Computer Science Department, Universitat Politècnica de Catalunya, Barcelona

Tomasz Kociumaka (109)
Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel; Institute of Informatics, University of Warsaw, Poland

Sofia Kokonezi (50)
Department of Mathematics, National and Kapodistrian University of Athens

Vladimir Kolmogorov (77)
Institute of Science and Technology Austria, Klosterneuburg, Austria

Antonina Kolokolova (66)
Memorial University of Newfoundland, St. John's, Canada

Christian Konrad (45)
University of Bristol, UK

Amos Korman (146)
Université de Paris, IRIF, CNRS, F-75013 Paris, France

Sajin Koroth (35)
Department of Computer Science, University of Haifa, Haifa, Israel

Adrian Kosowski (143)
Inria, Paris, France

Dariusz R. Kowalski (147)
Department of Computer Science, University of Liverpool, UK; SWPS University of Social Sciences and Humanities, Warsaw, Poland

Marcin Kozik (57)
Theoretical Computer Science Department, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland

Evangelos Kranakis (137)
School of Computer Science, Carleton University, Ottawa, Ontario, Canada

Stefan Kratsch (71)
Humboldt-Universität zu Berlin, Germany

Robert Krauthgamer (7)
Weizmann Institute of Science, Israel

Danny Krizanc (137)
Department of Mathematics & Comp. Sci., Wesleyan University, Middletown, CT, USA

Fabian Kuhn (144)
University of Freiburg, Germany

Amit Kumar (42, 63)
IIT Delhi, India

Mrinal Kumar (78)
University of Toronto, Canada

Orna Kupferman (100)
School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel

Adam Kurpisz (79)
ETH Zürich, Department of Mathematics, Rämistrasse 101, 8092 Zürich, Switzerland

William Kuszmaul (80)
Massachusetts Institute of Technology, Cambridge, USA

Shay Kutten (140)
Faculty of Industrial Engineering and Management, Technion - IIT, Haifa, Israel

Manuel Lafond (137)
Department of Computer Science, Université de Sherbrooke, Sherbrooke, Québec, Canada

Benoît Larose (29)
LACIM, Université du Québec a Montréal, Montréal, Canada

Kasper Green Larsen (10)
Computer Science Department, Aarhus University, Denmark

Alexandra Lassota (75)
Department of Computer Science, Kiel University, Kiel, Germany

Ron Lavi (140)
Faculty of Industrial Engineering and Management, Technion - IIT, Haifa, Israel

Xuan-Bach Le (103)
Department of Computer Science, University of Oxford, UK

Euiwoong Lee (42)
New York University, NY, USA

Troy Lee (16, 64)
Centre for Quantum Software and Information, School of Software, Faculty of Engineering and Information Technology, University of Technology Sydney, Australia

Stefano Leonardi (136)
Sapienza University of Rome, Italy

Nathan Lhote (106)
Institute of Informatics, University of Warsaw, Poland

Jason Li (41, 42)
Carnegie Mellon University, Pittsburgh, PA, USA

Tongyang Li (27)
Joint Center for Quantum Information and Computer Science, University of Maryland, USA

Xin Li (37)
Department of Computer Science, Johns Hopkins University, USA

Mehraneh Liaee (70)
Northeastern University, Boston, MA, USA

Anthony W. Lin (103)
Technische Universität Kaiserslautern, Germany

Bingkai Lin (81)
National Institute of Informatics, Tokyo, Japan; Nanjing University, Nanjing, China

Cedric Yen-Yu Lin (27)
Joint Center for Quantum Information and Computer Science, University of Maryland, USA

John Livieratos (50)
Department of Mathematics, National and Kapodistrian University of Athens

Philipp Loick (43)
Goethe University, Frankfurt, Germany

Daniel Lokshtanov (11, 24, 59, 60)
University of California Santa Barbara, Santa Barbara, California

Zhenjian Lu (39)
School of Computing Science, Simon Fraser University, Burnaby, BC, Canada

Austin Luchsinger (31)
Department of Computer Science, University of Texas - Rio Grande Valley, USA

Kelin Luo (141)
School of Management, Xi'an Jiaotong University, Xianning West Road, Xi'an, China

Christof Löding (120)
RWTH Aachen University, Ahornstr. 55, 52074 Aachen, Germany

Maarten Löffler (9)
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Bruce M. Maggs (70)
Duke University, Durham, NC, USA; Akamai Technologies, Cambridge, MA, USA

Frédéric Magniez (69)
Université de Paris, IRIF, CNRS, F-75013 Paris, France

Tal Malkin (15)
Columbia University, New York City, NY, USA

Frederik Mallmann-Trenn (148)
MIT, CSAIL, Cambridge, MA, US

Nikhil S. Mande (30)
Georgetown University, Washington, DC, USA

Florin Manea (109)
Department of Computer Science, Kiel University, Germany

Yuchen Mao (38)
Department of Computer Science and Engineering, HKUST, Hong Kong

Emanuel Martinov (115)
School of Informatics, University of Edinburgh, UK

Anders Martinsson (144)
ETH Zurich, Switzerland

Jannik Matuschke (82)
Research Center for Operations Management, KU Leuven, Leuven, Belgium

Yannic Maus (148)
Department of Computer Science, Technion, Haifa, Israel,

Richard Mayr (119)
University of Edinburgh, UK

Ruta Mehta (56)
University of Illinois at Urbana-Champaign, IL, USA

Or Meir (35)
Department of Computer Science, University of Haifa, Haifa, Israel

Themistoklis Melissourgos (138)
Department of Computer Science, University of Liverpool, Liverpool, UK

Arturo I. Merino (83)
Dept. of Mathematical Engineering and CMM, Universidad de Chile & UMI-CNRS 2807, Santiago, Chile

Ian Mertz (84)
University of Toronto, Canada

George B. Mertzios (131)
Department of Computer Science, Durham University, UK

Stefan Milius (130)
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Eyal Mizrachi (85)
Computer Science Department, Technion, Haifa 32000, Israel

Antonio Molina Lovett (121)
University of Waterloo, Canada

Shay Moran (34)
Department of Computer Science, Princeton University, Princeton, USA

Benjamin Moseley (86, 145)
Carnegie Mellon University, Pittsburgh, PA, USA

William K. Moses Jr. (140)
Faculty of Industrial Engineering and Management, Technion - IIT, Haifa, Israel

Miguel A. Mosteiro (147)
Computer Science Department, Pace University, New York, NY, USA

Anca Muscholl (122)
LaBRI, University of Bordeaux, France

Nabil H. Mustafa (87)
Université Paris-Est, Laboratoire d'Informatique Gaspard-Monge, ESIEE Paris, France

Dimitrios Myrisiotis (39)
Department of Computing, Imperial College London, London, UK

Veli Mäkinen (55)
Department of Computer Science, University of Helsinki, Finland

Joseph (Seffi) Naor (88)
Technion, Haifa, Israel

Lata Narayanan (137, 139)
Department of Comp. Sci. and Software Eng., Concordia University, Montreal, Québec, Canada

Sivaramakrishnan Natarajan Ramamoorthy (72)
Paul G. Allen School of Computer Science & Engineering, University of Washington, USA

Ofer Neiman (20)
Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Huy L. Nguyen (53, 54)
College of Computer and Information Science, Northeastern University, Boston, MA, USA

Lê Thành Dũng Nguyễn (123)
LIPN, UMR 7030 CNRS, Université Paris 13, Sorbonne Paris Cité, France

Sotiris Nikoletseas (131)
Computer Engineering & Informatics Department, University of Patras, and CTI, Greece

Reino Niskanen (103)
Department of Computer Science, University of Oxford, UK

Negev Shekel Nosatzki (15)
Columbia University, New York City, NY, USA

Petr Novotný (102)
Masaryk University, Brno, Czech Republic

Shunhao Oh (149)
Department of Computer Science, National University of Singapore

Igor Carboni Oliveira (32)
Department of Computer Science, University of Oxford, UK

Rafael Oliveira (78)
University of Toronto, Canada

Miroslav Olšák (57)
Department of Algebra, Charles University, Prague, Czech Republic

Jaroslav Opatrny (137, 139)
Department of Comp. Sci. and Software Eng., Concordia University, Montreal, Québec, Canada

Joël Ouaknine (44, 118)
The Max Planck Institute for Software Systems, Saarbrücken, Germany; Department of Computer Science, University of Oxford, United Kingdom

Dominik Pajak (148)
Faculty of Fundamental Problems of Technology, Wroclaw University of Science and Technology, Poland; Tooploox, Wroclaw, Poland

Wied Pakusa (112)
RWTH Aachen University, Germany

Konstantinos Panagiotou (90)
LMU München, Germany

Debmalya Panigrahi (70)
Duke University, Durham, NC, USA

Denis Pankratov (139)
Department of CSSE, Concordia University, Montreal, Canada

Fahad Panolan (60)
University of Bergen, Norway

Greta Panova (51)
University of Southern Californa, Los Angeles, CA, USA; University of Pennsylvania, Philadelphia, PA, USA

Manaswi Paraashar (16)
Indian Statistical Institute, Kolkata, India

Nikos Parotsidis (7)
University of Copenhagen, Denmark

Merav Parter (89)
Weizmann IS, Rehovot, Israel

Matija Pasch (90)
LMU München, Germany

Binghui Peng (67)
Tsinghua University, China

Simon Perdrix (108)
Université de Lorraine, CNRS, Inria, LORIA, F 54000 Nancy, France

Joseph G. Peters (134)
School of Computing Science, Simon Fraser University, Canada

Kalina Petrova (144)
ETH Zurich, Switzerland

Pascal Pfister (144)
ETH Zurich, Switzerland

Matthieu Picantin (124)
IRIF UMR 8243 CNRS & Univ Paris Diderot, 75013 Paris, France

Jean-Éric Pin (125)
IRIF, Université Paris Denis Diderot, CNRS - Case 7014 - F-75205 Paris Cedex 13, France

Anton Pirogov (120)
RWTH Aachen University, Ahornstr. 55, 52074 Aachen, Germany

Nadia Pisanti (21)
Department of Computer Science, University of Pisa, Italy; ERABLE Team, INRIA, France

Solon P. Pissis (21)
CWI, Amsterdam, The Netherlands

Toniann Pitassi (35, 84)
Department of Computer Science, University of Toronto, Canada

Thomas Place (126)
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400, Talence and IUF, France

Pierre Pradic (123)
ENS de Lyon, Université de Lyon, LIP, France; University of Warsaw, Faculty of Mathematics, Informatics and Mechanics, Poland

Eric Price (36)
The University of Texas at Austin, USA

Kirk Pruhs (145)
University of Pittsburgh, PA, USA

Gabriele Puppis (122)
CNRS, LaBRI, Bordeaux, France

Manish Purohit (145)
Google, Mountain View, CA, USA

Guillermo A. Pérez (102)
University of Antwerp, Belgium

Mikaël Rabie (135)
IRIF, Université de Paris, France; Aalto University, Finland

Akbar Rafiey (91)
Department of Computing Science, Simon Fraser University, Burnaby, Canada

Arash Rafiey (91)
Indiana State University, Terre Haute, IN, USA; Simon Fraser University, Burnaby, Canada

Rajmohan Rajaraman (70)
Northeastern University, Boston, MA, USA

Anup Rao (72)
Paul G. Allen School of Computer Science & Engineering, University of Washington, USA

Christoforos Raptopoulos (131)
Computer Engineering & Informatics Department, University of Patras, and CTI, Greece

Martin Raszyk (127)
Department of Computer Science, ETH Zürich, Universitätstrasse 6, 8092, Switzerland

Christophe Reutenauer (125)
Mathématiques, Université du Québec à Montréal, CP 8888, succ. Centre Ville, Canada H3C 3P8

Pierre-Alain Reynier (128)
Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France

Andrii Riazanov (92)
Computer Science Department, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, USA, 15213

Thomas Robinson (40)
Ben Gurion University of the Negev, Beersheva, Israel

Yoav Rodeh (146)
Ort Braude College, Karmiel, Israel

Liam Roditty (13)
Bar Ilan University, Ramat Gan, Israel

Lars Rohwedder (74, 75)
Department of Computer Science, Christian-Albrechts-Universität, Kiel, Germany

Miguel Romero (104)
Department of Computer Science, University of Oxford, Oxford, UK

Dana Ron (52)
Tel Aviv University, Tel Aviv, Israel

Will Rosenbaum (52)
Max Planck Institute for Informatics, Saarbrücken, Germany

Giovanna Rosone (21)
Department of Computer Science, University of Pisa, Italy

Günter Rote (9)
Institut für Informatik, Freie Universität Berlin, Takustraße 9, 14195 Berlin, Germany

Marc Roth (113)
Cluster of Excellence (MMCI), Saarland Informatics Campus (SIC), Saarbrücken, Germany

Ralf Rothenberger (61)
Algorithm Engineering Group, Hasso Plattner Institute, University of Potsdam, Germany

Chandan Saha (62)
Department of Computer Science and Automation, Indian Institute of Science, India

Andrej Sajenko (141)
THM, University of Applied Sciences Mittelhessen, Giessen, Germany

Miklos Santha (64)
CNRS, IRIF, Université de Paris, 75205 Paris, France; Centre for Quantum Technologies, National University of Singapore, Singapore 117543; MajuLab, UMI 3654, Singapore

Thiago Santos (91)
Indiana State University, Terre Haute, IN, USA

Swagato Sanyal (64)
Indian Institute of Technology Kharagpur, India

Ramprasad Saptharishi (78)
Tata Institute of Fundamental Research

Carlo Sartiani (101)
DIMIE, Università della Basilicata - Potenza, Italy

Thomas Sauerwald (93)
Department of Computer Science and Technology, University of Cambridge, United Kingdom

Saket Saurabh (11, 24, 59, 60)
Institute of Mathematical Sciences, HBNI and UMI ReLaX Chennai, India; University of Bergen, Bergen, Norway

Rahul Savani (56)
University of Liverpool, UK

Ali Sayyadi (142)
Sharif University of Technology, Iran

Christian Scheideler (150)
Paderborn University, Germany

Daniel Schmand (151)
Goethe University Frankfurt, Germany

Markus L. Schmid (109)
Trier University, Germany

Ulrike Schmidt-Kraepelin (82)
Institute of Software Engineering and Theoretical Computer Science, TU Berlin, Berlin, Germany

Sylvain Schmitz (129)
LSV, ENS Paris Saclay & CNRS, Université Paris-Saclay, France; IUF, France

Jason Schoeters (134)
LaBRI, Université de Bordeaux, CNRS, Bordeaux INP, France

Marc Schröder (151)
RWTH Aachen University, Germany

Roy Schwartz (85)
Computer Science Department, Technion, Haifa 32000, Israel

Robert Schweller (31)
Department of Computer Science, University of Texas - Rio Grande Valley, USA

Pavel Semukhin (44)
Department of Computer Science, University of Oxford, United Kingdom

Alexander Setzer (150)
Paderborn University, Germany

Jeffrey Shallit (121)
University of Waterloo, Canada

Sunil Shende (137)
Department of Computer Science, Rutgers University, Camden, NJ, USA

Mahsa Shirmohammadi (102, 119)
CNRS, Paris, France; IRIF, Paris, France

Alexandra Silva (107)
University College London, UK

Sahil Singla (63)
Princeton University and Institute for Advanced Study, USA

Alexander Skopalik (151)
University of Twente, Netherlands

Shay Solomon (17)
School of Electrical Engineering, Tel Aviv University, Israel

José A. Soto (83)
Dept. of Mathematical Engineering and CMM, Universidad de Chile & UMI-CNRS 2807, Santiago, Chile

Paul G. Spirakis (131, 138)
Department of Computer Science, University of Liverpool, UK; Computer Engineering & Informatics Department, University of Patras, Greece

Joachim Spoerhase (85)
Department of Computer Science, Aalto University, Espoo, Finland

Jakob T. Spooner (141)
Department of Informatics, University of Leicester, Leicester, England

Szymon Stankiewicz (57)
Theoretical Computer Science Department, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland

Clifford Stein (14)
Columbia University, New York City, NY, USA

Alistair Stewart (115)
Department of Computer Science, University of Southern California, Los Angeles, CA, USA

Xiaoming Sun (94)
CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China; University of Chinese Academy of Sciences, Beijing, China

Ravi Sundaram (70)
Northeastern University, Boston, MA, USA

Ola Svensson (3)
EPFL, Lausanne, Switzerland

Krysta M. Svore (27)
Station Q, Quantum Architectures and Computation Group, Microsoft Research, USA

Avishay Tal (66)
Stanford University, USA

Prafullkumar Tale (11)
Institute of Mathematical Sciences, HBNI, Chennai, India

Justin Thaler (30)
Georgetown University, Washington, DC, USA

Mikkel Thorup (95)
Department of Computer Science, University of Copenhagen, Denmark

Alexandru I. Tomescu (55)
Department of Computer Science, University of Helsinki, Finland

Patrick Totzke (119)
University of Liverpool, UK

Ohad Trabelsi (7)
Weizmann Institute of Science, Israel

Dmitriy Traytel (127)
Department of Computer Science, ETH Zürich, Universitätstrasse 6, 8092, Switzerland

Christos Tzamos (19)
University of Wisconsin-Madison, USA

Seeun William Umboh (88)
The University of Sydney, Australia

Sumedha Uniyal (85)
Department of Computer Science, Aalto University, Espoo, Finland

Henning Urbat (130)
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Przemysław Uznański (7)
University of Wrocław, Poland

Frits Vaandrager (4)
Department of Software Science, Radboud University, The Netherlands

Joran van Apeldoorn (99)
QuSoft, CWI, The Netherlands

José Verschae (82)
Institute of Engineering Sciences, Universidad de O'Higgins, Rancagua, Chile

Laurent Viennot (143)
Inria, Paris, France

Didier Villevalois (128)
Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France

Renaud Vilmart (108)
Université de Lorraine, CNRS, Inria, LORIA, F 54000 Nancy, France

Nikhil Vyas (46, 47)
MIT, Cambridge, MA, USA

David Wajc (67)
Carnegie Mellon University, Pittsburgh, PA, USA

Manfred K. Warmuth (34)
Computer Science Department, University of California, Santa Cruz, USA

Thomas Watson (96)
University of Memphis, Memphis, TN, USA

Yuanhao Wei (84)
Carnegie Mellon University, Pittsburgh, PA, USA

Nicole Wein (13, 46, 47)
MIT, Cambridge, MA, USA

Philip Wellnitz (113)
Max Planck Institute for Informatics, Saarland Informatics Campus (SIC), Saarbrücken, Germany

Colin White (18)
Carnegie Mellon University, Pittsburgh, PA, USA

Ryan Williams (25, 26)
Department of Electrical Engineering and Computer Science & CSAIL, MIT, Cambridge, MA, USA

Virginia Vassilevska Williams (13, 46, 47)
MIT, Cambridge, MA, USA

David P. Williamson (88)
Cornell University, Ithaca, NY, USA

Daniel Wolleb-Graf (7)
ETH Zürich, Switzerland

David P. Woodruff (18, 94, 97)
Carnegie Mellon University, Pittsburgh, PA, USA

James Worrell (44, 118)
Department of Computer Science, University of Oxford, United Kingdom

Hongxun Wu (48)
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China

Ke Wu (37)
Department of Computer Science, Johns Hopkins University, USA

Xiaodi Wu (27)
Joint Center for Quantum Information and Computer Science, University of Maryland, USA

Tim Wylie (31)
Department of Computer Science, University of Texas - Rio Grande Valley, USA

Chaoping Xing (68, 98)
School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore

Yinzhan Xu (46)
MIT, Cambridge, MA, USA

Guang Yang (94, 97)
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China; Conflux, Beijing, China

Mihalis Yannakakis (5, 115)
Department of Computer Science, Columbia University, New York City, NY, USA

Amir Yehudayoff (72)
Department of Mathematics, Technion-IIT, Haifa, Israel

Eylon Yogev (89)
Technion, Haifa, Israel

Akira Yoshimizu (111)
INRIA Sophia Antipolis, France

Yuancheng Yu (46)
MIT, Cambridge, MA, USA

Chen Yuan (98)
Centrum Wiskunde & Informatica, Amsterdam, The Netherlands

Viktor Zamaraev (131)
Department of Computer Science, Durham University, UK

Or Zamir (95)
Blavatnik School of Computer Science, Tel Aviv University, Israel

Luca Zanetti (93)
Department of Computer Science and Technology, University of Cambridge, United Kingdom

Meirav Zehavi (24, 59, 60)
Ben-Gurion University, Beersheba, Israel

Marc Zeitoun (126)
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400, Talence, France

Jialin Zhang (94)
CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China; University of Chinese Academy of Sciences, Beijing, China

Peilin Zhong (14)
Columbia University, New York City, NY, USA

Xue Zhu (73)
The University of Hong Kong

David Zuckerman (58)
Department of Computer Science, UT Austin, Austin, TX, USA

Uri Zwick (95, 114)
Blavatnik School of Computer Science, Tel Aviv University, Israel

Jakub Łącki (136)
Google Research, New York, USA

Daniel Štefankovič (22)
Department of Computer Science, University of Rochester, Rochester, NY, USA

Auction Design under Interdependent Values

Michal Feldman
Blavatnik School of Computer Science, Tel-Aviv University, Israel
michal.feldman@cs.tau.ac.il

Abstract

We study combinatorial auctions with interdependent valuations. In such settings, every agent has a private signal, and every agent has a valuation function that depends on the private signals of all the agents. Interdependent valuations capture settings where agents lack information to determine their own valuations. Examples include auctions for artwork or oil drilling rights. For single item auctions and assume some restrictive conditions (the so-called single-crossing condition), full welfare can be achieved. However, in general, there are strong impossibility results on welfare maximization in the interdependent setting. This is in contrast to settings where agents are aware of their own valuations, where the optimal welfare can always be obtained by an incentive compatible mechanism.

Motivated by these impossibility results, we study welfare maximization for interdependent valuations through the lens of approximation. We introduce two valuation properties that enable positive results. The first is a relaxed, parameterized version of single crossing; the second is a submodularity condition over the signals. We obtain a host of approximation guarantees under these two notions for various scenarios.

Related publications: [1, 2]

2012 ACM Subject Classification Theory of computation → Algorithmic game theory and mechanism design

Keywords and phrases Combinatorial auctions, Interdependent values, Welfare approximation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.1

Category Invited Talk

References

1 Alon Eden, Michal Feldman, Amos Fiat, and Kira Goldner. Interdependent Values without Single-Crossing. In Éva Tardos, Edith Elkind, and Rakesh Vohra, editors, *Proceedings of the 2018 ACM Conference on Economics and Computation, Ithaca, NY, USA, June 18-22, 2018*, page 369. ACM, 2018. doi:10.1145/3219166.3219173.

2 Alon Eden, Michal Feldman, Amos Fiat, Kira Goldner, and Anna R. Karlin. Combinatorial Auctions with Interdependent Valuations: SOS to the Rescue. In *Proceedings of the 2019 ACM Conference on Economics and Computation, Phoenix, AZ, June 24-28, 2019*. ACM, 2019.

Symmetry and Similarity

Martin Grohe
RWTH Aachen University, Lehrstuhl Informatik 7, Ahornstr. 55, 52074 Aachen, Germany
http://www.lics.rwth-aachen.de/~grohe
grohe@informatik.rwth-aachen.de

Abstract

Deciding if two graphs are isomorphic, or equivalently, computing the symmetries of a graph, is a fundamental algorithmic problem. It has many interesting applications, and it is one of the few natural problems in the class NP whose complexity status is still unresolved. Three years ago, Babai (STOC 2016) gave a quasi-polynomial time isomorphism algorithm. Despite of this breakthrough, the question for a polynomial algorithm remains wide open.

Related to the isomorphism problem is the problem of determining the similarity between graphs. Variations of this problems are known as robust graph isomorphism or graph matching (the latter in the machine learning and computer vision literature). This problem is significantly harder than the isomorphism problem, both from a complexity theoretical and from a practical point of view, but for many applications it is the more relevant problem.

My talk will be a survey of recent progress on the isomorphism and on the similarity problem. I will focus on generic algorithmic strategies (as opposed to algorithms tailored towards specific graph classes) that have proved to be useful and interesting in various context, both theoretical and practical.

2012 ACM Subject Classification Mathematics of computing → Graph theory; Theory of computation → Graph algorithms analysis

Keywords and phrases Graph Isomorphism, Graph Similarity, Graph Matching

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.2

Category Invited Talk

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 2; pp. 2:1–2:1

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

Approximately Good and Modern Matchings

Ola Svensson
EPFL, Lausanne, Switzerland
https://theory.epfl.ch/osven/
ola.svensson@epfl.ch

Abstract

The matching problem is one of our favorite benchmark problems. Work on it has contributed to the development of many core concepts of computer science, including the equation of efficiency with polynomial time computation in the groundbreaking work by Edmonds in 1965.

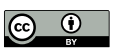
However, half a century later, we still do not have full understanding of the complexity of the matching problem in several models of computation such as parallel, online, and streaming algorithms. In this talk we survey some of the major challenges and report some recent progress.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Algorithms, Matchings, Computational Complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.3

Category Invited Talk



46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 3; pp. 3:1–3:1

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

Automata Learning and Galois Connections

Frits Vaandrager
Department of Software Science, Radboud University, The Netherlands
http://www.cs.ru.nl/~fvaan/
F.Vaandrager@cs.ru.nl

Abstract

Automata learning is emerging as an effective technique for obtaining state machine models of software and hardware systems. I will present an overview of recent work in which we used active automata learning to find standard violations and security vulnerabilities in implementations of network protocols such as TCP and SSH. Also, I will discuss applications of automata learning to support refactoring of legacy control software and identifying job patterns in manufacturing systems. As a guiding theme in my presentation, I will show how Galois connections (adjunctions) help us to scale the application of learning algorithms to practical problems.

2012 ACM Subject Classification Theory of computation → Active learning; Theory of computation → Regular languages; Security and privacy → Logic and verification; Software and its engineering → Model-driven software engineering; Software and its engineering → Software testing and debugging

Keywords and phrases Automaton Learning, Model Learning, Protocol Verification, Applications of Automata Learning, Galois Connections

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.4

Category Invited Talk

Funding NWO TOP project 612.001.852 Grey-box learning of Interfaces for Refactoring Legacy Software (GIRLS)

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 4; pp. 4:1–4:1

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

Fixed Point Computation Problems and Facets of Complexity

Mihalis Yannakakis
Department of Computer Science, Columbia University, 455 Computer Science Building,
1214 Amsterdam Avenue, New York, NY 10027, USA
mihalis@cs.columbia.edu

Abstract

Many problems from a wide variety of areas can be formulated mathematically as the problem of computing a fixed point of a suitable given multivariate function. Examples include a variety of problems from game theory, economics, optimization, stochastic analysis, verification, and others. In some problems there is a unique fixed point (for example if the function is a contraction); in others there may be multiple fixed points and any one of them is an acceptable solution; while in other cases the desired object is a specific fixed point (for example the least fixed point or greatest fixed point of a monotone function). In this talk we will discuss several types of fixed point computation problems, their complexity, and some of the common themes that have emerged: classes of problems for which there are efficient algorithms, and other classes for which there seem to be serious obstacles.

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

Keywords and phrases Fixed Point, Polynomial Time Algorithm, Computational Complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.5

Category Invited Talk

Funding *Mihalis Yannakakis*: Supported by NSF Grants CCF-1703925, CCF-1763970.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 5; pp. 5:1–5:1

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

Complexity-Theoretic Limitations on Blind Delegated Quantum Computation

Scott Aaronson
Department of Computer Science, University of Texas at Austin, USA
aaronson@cs.utexas.edu

Alexandru Cojocaru
School of Informatics, University of Edinburgh, UK
a.cojocaru@sms.ed.ac.uk

Alexandru Gheorghiu[1]
Department of Computing and Mathematical Sciences, California Institute of Technology, USA
School of Informatics, University of Edinburgh, UK
andrugh@caltech.edu

Elham Kashefi
School of Informatics, University of Edinburgh, UK
CNRS LIP6, Université Pierre et Marie Curie, Paris, France
ekashefi@inf.ed.ac.uk

Abstract

Blind delegation protocols allow a client to delegate a computation to a server so that the server learns nothing about the input to the computation apart from its size. For the specific case of *quantum computation* we know, from work over the past decade, that blind delegation protocols can achieve information-theoretic security (provided the client and the server exchange some amount of quantum information). In this paper we prove, provided certain complexity-theoretic conjectures are true, that the power of *information-theoretically secure* blind delegation protocols for quantum computation (ITS-BQC protocols) is in a number of ways constrained.

In the first part of our paper we provide some indication that ITS-BQC protocols for delegating polynomial-time quantum computations in which the client and the server interact only classically are unlikely to exist. We first show that having such a protocol in which the client and the server exchange $\mathsf{O}(\mathsf{n}^\mathsf{d})$ bits of communication, implies that $\mathsf{BQP} \subset \mathsf{MA}/\mathsf{O}(\mathsf{n}^\mathsf{d})$. We conjecture that this containment is unlikely by proving that there exists an oracle relative to which $\mathsf{BQP} \not\subset \mathsf{MA}/\mathsf{O}(\mathsf{n}^\mathsf{d})$. We then show that if an ITS-BQC protocol exists in which the client and the server interact only classically and which allows the client to delegate quantum sampling problems to the server (such as BosonSampling) then there exist non-uniform circuits of size $2^{\mathsf{n}-\Omega(\mathsf{n}/\log(\mathsf{n}))}$, making polynomially-sized queries to an NP^NP oracle, for computing the permanent of an $n \times n$ matrix.

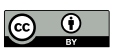
The second part of our paper concerns ITS-BQC protocols in which the client and the server engage in one round of quantum communication and then exchange polynomially many classical messages. First, we provide a complexity-theoretic upper bound on the types of functions that could be delegated in such a protocol by showing that they must be contained in $\mathsf{QCMA}/\mathsf{qpoly} \cap \mathsf{coQCMA}/\mathsf{qpoly}$. Then, we show that having such a protocol for delegating NP-hard functions implies $\mathsf{coNP}^{\mathsf{NP}^{\mathsf{NP}}} \subseteq \mathsf{NP}^{\mathsf{NP}^{\mathsf{PromiseQMA}}}$.

2012 ACM Subject Classification Theory of computation → Quantum computation theory; Theory of computation → Quantum complexity theory

Keywords and phrases Quantum cryptography, Complexity theory, Delegated quantum computation, Computing on encrypted data

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.6

[1] Corresponding author.



46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 6; pp. 6:1–6:13

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

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1704.08482`.

Funding *Scott Aaronson*: Vannevar Bush Fellowship from the US Department of Defense.
Alexandru Cojocaru: EPSRC grants EP/N003829/1, EP/M013243/1.
Alexandru Gheorghiu: MURI Grant FA9550-18-1-0161 and the IQIM, an NSF Physics Frontiers Center (NSF Grant PHY-1125565) with support of the Gordon and Betty Moore Foundation (GBMF-12500028).
Elham Kashefi: EPSRC grants EP/N003829/1, EP/M013243/1.

Acknowledgements We would like to thank the following people for useful discussions and comments: Petros Wallden, Matty J Hoban, Kousha Etessami, Marc Kaplan, Ronald de Wolf, Urmila Mahadev, Umesh Vazirani, Chris Heunen, Thomas Vidick, Ashley Montanaro, Tina Zhang and Pia Kullik. A.G. is in particular grateful to Petros Wallden and Matty J Hoban for their patience and for their help in clarifying several technical issues.

1 Introduction

An important area of research in modern cryptography is that of performing *computations on encrypted data*. The general idea is that a client wants to compute some function f on some input x, but lacks the computational power to do this in a reasonable amount of time. Luckily, the client has access to a computationally powerful server (cloud, cluster etc) which can compute $f(x)$ quickly. However, because the computation might involve sensitive or classified information, or the server could be compromised remotely, we would like the input x to be hidden from the server at all times. The client can simply encrypt x, but this raises the question: how can the server compute $f(x)$ if it doesn't know x? The general problem of computing on encrypted data was first considered by Rivest, Adleman and Dertouzos [52]. Since then, instances of this problem have appeared in many areas of modern research including those of electronic voting, machine learning on encrypted data, program obfuscation and others [22, 32, 11, 28, 37, 41].

It was shown in 2009, when Gentry produced the first *fully homomorphic encryption scheme*, that performing classical computations on encrypted data is possible [29]. In homomorphic encryption the client has a pair of efficient algorithms (Enc, Dec), which respectively perform encryption and decryption, and which satisfy the property $Dec(f, x, Eval(f, Enc(x))) = f(x)$, for any function f from some set $\mathcal{C}$. In other words, the server evaluates f on the encrypted input $Enc(x)$ using $Eval$ and returns this to the client which can then decrypt it to $f(x)$. Of course, the server should not be able to infer information about x from $Enc(x)$, a condition which is typically expressed through the criterion of *semantic security* [40]. If the set $\mathcal{C}$ contains all polynomial-sized circuits then the scheme becomes a fully homomorphic encryption scheme, commonly abbreviated FHE. All known FHE schemes are secure under *cryptographic assumptions*.

Computing on encrypted data becomes particularly interesting when the server is a *quantum computer*. This is because efficient quantum algorithms have been found for various problems which are believed to be intractable for classical computers. In fact, it has been shown that if a classical computer and a quantum computer are both given black-box or oracle access to certain functions, then the quantum computer exponentially outperforms the classical computer [13, 55, 23, 5]. Classical clients would therefore be highly motivated to delegate problems to quantum computers. However, ensuring the privacy of their inputs is challenging. In particular, we'd have to solve the following problems:

- Devise an encryption scheme which is secure against quantum computers and does not leak information to the server about the client's input.
- Ensure that the encryption scheme allows the client to recover the output of the computation from the result provided by the quantum server.
- Ensure that the protocol is efficient for the client. Ideally, the number of rounds of interaction between the client and the server as well as the client's local computations, should scale at most polynomially with the size of the input.

In spite of these stringent requirements, protocols that achieve these properties already exist and are known collectively as *delegated blind quantum computing schemes* [26]. In such protocols, a probabilistic polynomial-time client is able to delegate polynomial-time quantum computations to a server in such a way that the client's input (apart from an upper bound on its size) is kept hidden from the server in an *information-theoretic* sense. All of the above schemes require the client and the server to share at least one round of quantum communication. *Universal Blind Quantum Computation* (UBQC), shown schematically in Figure 1, is an example of such a protocol [19].

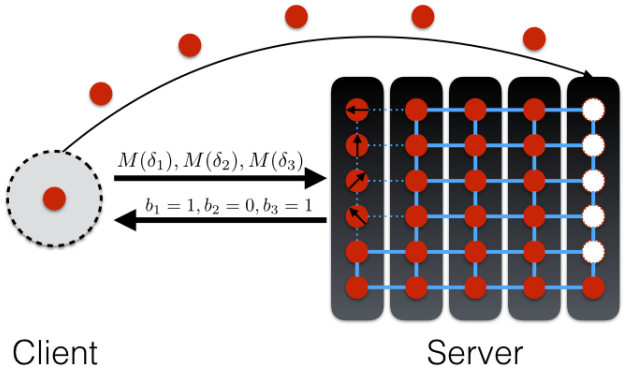


Figure 1 Universal Blind Quantum Computation [19]. In UBQC, a classical client augmented with the ability to prepare single-qubit states sends these qubits to the server along with instructions on how to entangle and measure them in order to perform a computation. The $M(\delta_i)$ indicate measurement instructions and the b_i indicate the server's responses for these instructions (if he follows the protocol, these responses would represent the outcomes of the measurements that the client instructed him to perform).

The first blind delegation protocol was devised by Childs in [21], and since then these protocols have been improved and extended in various works [47, 31, 44, 27, 45, 46, 39, 38]. UBQC and related protocols require the client and the server to exchange only one quantum message, while the rest of the communication is classical [19, 9, 18]. This quantum message (which is sent by the client to the server) consists of a tensor product of single-qubit states. As such, the only quantum capability the client needs is the ability to prepare single-qubit states. In this paper, we explore two questions pertaining to blind delegation protocols:

(1) Is there a scheme for blind quantum computing that is information-theoretically secure, and that requires only classical communication between client and server?

(2) For schemes in which the client and the server are allowed one round of quantum communication, which functions can the client delegate to the server while maintaining information-theoretic security? In particular, could the client delegate the evaluation of NP-hard functions?

We provide some indication, based on complexity-theoretic conjectures, that the answer to the first question is no. In other words, provided these complexity-theoretic conjectures hold, a classical client running in polynomial time and communicating only classically with a

server cannot delegate arbitrary polynomial-time quantum computations to that server while keeping its input hidden in an information-theoretic sense. Importantly, our result does not contradict recent results on quantum fully homomorphic encryption with a classical client [43, 15], since those schemes are based on cryptographic assumptions: *we are interested only in information-theoretic security.*

In answer to the second question, we provide a complexity-theoretic upper bound on the types of functions that can be evaluated by UBQC-type protocols (i.e. protocols in which the client can send one quantum message to the server[2]). We show that, under plausible complexity-theoretic assumptions, this upper bound prevents the client from delegating NP-hard functions to the server. Thus, allowing for quantum communication between the client and the server expands the set of functions that the client can delegate to the server to include BQP, but not enough so as to include NP as well.

1.1 Main results

We phrase our results formally using the concept of a *generalised encryption scheme* (GES) introduced by Abadi, Feigenbaum and Killian [8]. Roughly speaking, a GES is a protocol between a probabilistic polynomial-time classical client and a computationally unbounded server for computing on encrypted data. The client sends the server a description of some function[3] $f : \{0,1\}^n \to \{0,1\}$. Using some polynomial-time algorithm denoted E, the client encrypts its input x, and sends $E(x)$ to the server. The server and the client then interact for a number of rounds which is polynomial in the length of x. Finally, using a polynomial-time decryption algorithm denoted D, the client decrypts the server's responses and obtains $f(x)$ with probability $1/2 + 1/\operatorname{poly}(n)$. Importantly throughout the protocol, the server learns no more than the length of x. Because the server is computationally unbounded, the scheme requires information-theoretic security. Abadi et al. gave a complexity theoretic upper bound on the types of functions that admit such a scheme. They showed that any function f that the client could delegate in a GES must be contained in the class $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$.

The GES framework allows us to restate the questions we address in this paper as follows:

(1) Can we design a GES for delegating BQP functions? Note that, by the Abadi et al. result, this is the same as asking whether $\mathsf{BQP} \subset \mathsf{NP/poly} \cap \mathsf{coNP/poly}$. We will consider two variants on the GES framework: one which allows the client to delegate sampling problems to the server, and one in which the total communication between client and server is bounded by $\mathsf{O}(\mathsf{n}^\mathsf{d})$, for some constant $d > 0$. For the former, we show that having such a scheme for quantum sampling problems, like BosonSampling, implies that circuits exist which can compute the permanent of a matrix more efficiently than we believe is possible. For the latter, having a GES with bounded communication for polynomial-time quantum computation implies that $\mathsf{BQP} \subset \mathsf{MA/O}(\mathsf{n}^\mathsf{d})$, and we argue that this containment is unlikely by providing an oracle separation between these classes.

(2) If we change the GES framework to allow one round of quantum communication between the client and the server, what functions can the client delegate to the server? We answer this question by "quantising" the Abadi et al. result and showing that such

[2] In fact our result concerns protocols in which the client and the server start with *one round* of quantum communication, followed by polynomially-many rounds of classical communication. In other words, not only is there one quantum message from the client to the server, but the server is also allowed to respond with a quantum message.

[3] Unless otherwise specified, we restrict our attention to decision problems. This is why the function f has the codomain $\{0,1\}$.

functions would be contained in $\mathsf{QCMA/qpoly} \cap \mathsf{coQCMA/qpoly}$ (a quantum analogue of $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$). We also show that $\mathsf{QCMA/qpoly} \cap \mathsf{coQCMA/qpoly}$ is unlikely to contain NP-hard functions.

The complete proofs for our results can be found in the full version of our paper [7].

1.1.1 Generalised encryption scheme for BQP decision problems

As we have mentioned, for the case of decision problems, Abadi et al. showed that the class of problems that a client can delegate to a server using the GES framework is contained in $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$. They also observed that if NP-hard functions could be delegated by the client using a GES, then $\mathsf{NP} \subset \mathsf{NP/poly} \cap \mathsf{coNP/poly}$, and, in particular, $\mathsf{NP} \subset \mathsf{coNP/poly}$. Yap showed that having such a containment leads to a collapse of the polynomial hierarchy at the third level [58]. In other words, it seems unlikely that NP-hard problems would admit a GES.

What about BQP-hard functions? The Abadi et al. result implies that having a GES for BQP-hard functions leads to $\mathsf{BQP} \subset \mathsf{NP/poly} \cap \mathsf{coNP/poly}$. While we would like to argue, similarly, that such a containment leads to a collapse of the polynomial hierarchy, even $\mathsf{BQP} = \mathsf{P}$ isn't known to lead to such a collapse. We instead consider a modified GES in which the total communication between the client and the server is upper bounded by a polynomial of *fixed* degree, $d > 0$, in the size of the input[4]. In that case, it can be shown that $\mathsf{BQP} \subset \mathsf{MA/O}(\mathsf{n^d}) \cap \mathsf{coMA/O}(\mathsf{n^d})$. We argue that this containment is unlikely based on the following result:

▶ **Theorem 1.** *For each $d \in \mathbb{N}$, there exists an oracle O_d such that BQP^{O_d} is not contained in $(\mathsf{MA/O}(\mathsf{n^d}))^{O_d}$.*

Essentially, the theorem shows that relative to an oracle O_d, there are problems that can be solved by a polynomial-time quantum algorithm, but which a classical client cannot delegate to a server in a GES with bounded communication. Since the oracle is parameterised by d, we are in fact defining a family of oracles. The specific problem on which the oracle O_d is based is a version of *Simon's problem* [55]. Simon's problem is the following: for an input of size n, and given oracle access to a function $g : \{0,1\}^n \to \{0,1\}^n$ that is guaranteed to be either an injective function, or a 2-to-1 and periodic function[5], the task is to decide which is the case. Simon provided a polynomial-time quantum algorithm for solving this problem, thus showing that it belongs to BQP (relative to the function oracle). For the case in which one should accept when the function is 2-to-1, the problem can be shown to be outside of MA (relative to the function oracle). As such, Simon's problem provides an oracle separation between BQP and MA.

In Simon's original construction, the oracle function is the same for all inputs of size n. Note that, this version of the problem can be solved with one bit of advice: for all inputs of size n, the advice bit simply specifies whether the function is 1-to-1 or 2-to-1 and periodic. Therefore such a setup would not be useful in our case. For this reason, in our proof of Theorem 1, the function that the oracle provides access to is input-dependent. The problem we define, relative to this oracle, is again to decide whether the function is 1-to-1 or the function is 2-to-1 and periodic. However, we can show that, by considering a sufficiently

[4] Note that we impose no such restriction on the running time of the client.
[5] In other words, there exists a period $s \in \{0,1\}^n$, $s \neq 0^n$, such that for all $x, y \in \{0,1\}^n$, $x \neq y$, it is the case that $g(x) = g(y)$ iff. $x = s \oplus y$.

large domain for these functions – in other words, by letting $g : \{0,1\}^{n^D} \to \{0,1\}^{n^D}$ for some $D > d$ – the problem is not contained in $(\mathsf{MA}/\mathsf{O}(\mathsf{n}^\mathsf{d}))^{O_d}$, but is nevertheless contained in BQP. The proof uses a diagonalisation argument and can be found in the full version of our paper [7].

Unfortunately, the same oracle cannot be used to separate BQP from $\mathsf{NP/poly}$. This is because D is a function of d; to prove a separation with respect to $\mathsf{NP/poly}$, where the length of the advice string can be any polynomial, we would have to find an oracle that works *for all* possible values of d. It would be interesting to see whether the oracle that Raz and Tal [51] recently used to prove a separation between BQP and PH could also be used in order to separate BQP from $\mathsf{NP/poly}$, or even from $\mathsf{PH/poly}$. We leave this as an open problem.

One can argue that oracle results do not constitute compelling evidence on the relationships between complexity classes. For example, it has been known for a while that there exist oracles O_1, O_2 such that $\mathsf{P}^{O_1} \neq \mathsf{NP}^{O_1}$ but $\mathsf{P}^{O_2} = \mathsf{NP}^{O_2}$, and that, while $\mathsf{IP} = \mathsf{PSPACE}$, there is an oracle such that $\mathsf{IP}^O \neq \mathsf{PSPACE}^O$. Nonetheless, oracles allow us to study the query complexity of problems in different models of computation. In fact, there are situations in practice where computer programs are restricted to making black-box calls to functions in order to determine their properties [36]. Apart from this, oracle results have also inspired a number of important developments in algorithms and complexity theory[6]. For more arguments concerning the usefulness of oracle results, see Section 1.3 of [2].

1.1.2 Generalised encryption scheme for BQP sampling problems

We consider what would happen if we have a generalised encryption scheme which allowed a client to delegate a sampling problem, such as BosonSampling, to the server. BosonSampling, defined by Aaronson and Arkhipov in [6], is essentially the problem of simulating the statistics of photons (bosons) passing through a linear optics network. One starts with a configuration of identical photons in known locations (referred to as *modes*). The photons then pass through the linear optics network, which consists of optical elements (beamsplitters and phase shifters). Finally, one performs a measurement to determine the new locations of the photons in the *output modes* of the system. The reason this is referred to as a sampling problem is because we have a probability distribution over the different configurations of photons in the output modes. In *exact* BosonSampling, which is the problem we consider, the task is to produce a sample from that probability distribution. Aaronson and Arkhipov showed that the probability of observing a particular configuration of photons is proportional to the squared permanent of a matrix that can be obtained efficiently from the description of the optical network. They also showed that no polynomial-time probabilistic algorithm can sample from this distribution, unless the polynomial hierarchy collapses at the third level [6]. As such, while a quantum computer could simulate the optical network and sample from the target distribution in polynomial time, it seems unlikely that classical computers could do the same.

Sampling problems, like BosonSampling, are of interest because of their potential use in demonstrating *quantum computational supremacy* [35]. This entails having a quantum device perform a computational task that no classical computer would be able to reproduce *efficiently*. Sampling problems are natural candidates for this task for two main reasons. Firstly, many of the quantum sampling problems that have been considered could in principle be performed on a small-scale quantum computer having up to (or on the order of) 100

[6] A notable example is the fact that Simon's oracle separation between BPP and BQP led to Shor's algorithm for factoring and computing the discrete logarithm [54]

qubits and not requiring fault-tolerance [50]. Secondly, it has been shown that having a polynomial-time classical algorithm that can sample from the distribution of the quantum sampling problem leads to a collapse of the polynomial hierarchy. Contrast this to tasks such as factoring for which the existence of an efficient classical algorithm is not known to lead to any "disastrous" complexity-theoretic consequences.

Given that sampling problems are primarily considered in the context of demonstrating quantum computational supremacy, one could ask why a client would like to delegate such a problem to the server via a GES. Firstly, all existing schemes for blind delegated quantum computation allow the client to delegate both decision and sampling problems [26]. It is therefore natural to ask whether such a scheme, involving only classical communication, can also exist. Secondly, there is currently no known way for a classical client to efficiently certify whether the server has sampled from the correct distribution (at least with an information-theoretic guarantee). In fact, in certain cases the client would require exponentially many samples from the server in order to perform this verification [34]. However, if a GES for quantum sampling problems existed, the client might be able to leverage it in order to perform the certification, in the same way that many delegated quantum computation protocols leverage blindess to achieve verifiability [30]. Finally, due to the equivalence between sampling and searching shown in [3], our result holds if we substitute sampling problems with search problems.

In a GES for exact BosonSampling, the client's input would be a description of a linear optics network[7]. The client would like to delegate to the server the task of sampling from the BosonSampling distribution associated with this network, while keeping the description of the network hidden from the server. In other words, upon interacting with the server and decrypting its responses, the client should obtain a sample from the BosonSampling distribution. At the same time, the server learns at most an upper bound on the size of the network. We show the following:

▶ **Theorem 2.** *If exact* BosonSampling *admits a GES, then for any matrix* $X \in \{-1, 0, 1\}^{n\times n}$*, there exist circuits of size* $2^{n-\Omega\left(\frac{n}{\log n}\right)}$*, making polynomially-sized queries to an* $\mathsf{NP}^{\mathsf{NP}}$ *oracle, for computing the permanent of* X*.*

Computing the permanent of a matrix is a problem known to be #P-hard. By Toda's theorem, this means that if computing the permanent were possible at any level of the polynomial hierarchy, the hierarchy would collapse at that level [56]. Moreover, the best known algorithm for computing the permanent, by Björklund, has a run-time of $2^{n-\Omega\left(\sqrt{n/\log(n)}\right)}$ [14]. Prior to that, the leading algorithm for computing the permanent was Ryser's algorithm, developed over 50 years ago, which requires $O(n2^n)$ arithmetic operations [53]. We conjecture that the circuits of Theorem 2 do not exist and, thus, that there can be no GES for BosonSampling.

1.1.3 Quantum generalised encryption scheme

While having a GES for delegating BQP computations seems unlikely, we know that giving the client some minimal quantum capabilities removes this limitation: schemes such as UBQC exist which allow for the information-theoretically secure blind delegation of quantum

[7] In principle, one could also specify the configuration of the photons in the input modes as part of the client's input. Equivalently, however, one can always initialise the input modes to some fixed initial state, and produce whichever starting state is in fact desired by altering the linear optics network.

computations. In the spirit of the Abadi et al. result, it is natural to consider *quantum generalised encryption schemes* (or QGES), in which the client is no longer classical, and investigate the complexity-theoretic upper bounds on functions that admit such a protocol. For the QGES, we are still assuming information-theoretic security and that the encryption scheme leaks at most the size of the input. However, unlike the GES, the client is now assumed to be a quantum computer performing polynomial-time computations[8]. Additionally, the client and the server perform one round of quantum communication at the beginning of the protocol. The rest of the communication is classical.

We impose one other restriction on the QGES, known as *offline-ness*. Roughly speaking, an offline protocol is one in which the client does not need to commit to any particular input (of a given size), after having sent the quantum message to the server. The quantum message only depends on the size of the input. We note that offline-ness is a property which UBQC and all other currently known blind quantum computing protocols share. From a practical perspective, this presents the client with the option of sending the first quantum message to the server and deciding at a later time on which input the server should perform the computation. One could imagine, for instance, that the client and the server have access to a quantum channel for a limited amount of time. In practice, such a situation can occur if the communication between the parties is mediated by a satellite, as is the case with satellite-based quantum-key distribution [42]. In this case, the satellite is in the line of sight of the two parties for only a few minutes at a time. Our result is the following:

▶ **Theorem 3.** *The class of functions that a client can delegate to a server in an offline QGES is contained in* $\mathsf{QCMA/qpoly} \cap \mathsf{coQCMA/qpoly}$.

Note that the class $\mathsf{QCMA/qpoly} \cap \mathsf{coQCMA/qpoly}$ can be seen as a quantum analogue of the class $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$ which we encounter in the GES case. We therefore view Theorem 3 as a "quantisation" of the Abadi et al. bound on the power of generalised encryption schemes.

Again, in the spirit of the Abadi et al. result, one can ask whether NP-complete functions are contained in $\mathsf{QCMA/qpoly} \cap \mathsf{coQCMA/qpoly}$. In other words: does giving quantum capabilities to the client increase the class of functions that it can securely delegate so that this class contains NP? We give an indication that the answer is no:

▶ **Theorem 4.** $\mathsf{NP} \subset \mathsf{QCMA/qpoly} \cap \mathsf{coQCMA/qpoly}$ *implies* $\mathsf{coNP}^{\mathsf{NP}^{\mathsf{NP}}} \subseteq \mathsf{NP}^{\mathsf{NP}^{\mathsf{PromiseQMA}}}$.

Note that if PromiseQMA in the above expression were replaced with NP, this would imply a collapse of the polynomial hierarchy at the third level. Our result is as close to a collapse of the polynomial hierarchy as one can reasonably hope to get, given a quantum hypothesis. Hence, while a QGES does allow the client to delegate BQP computations, it seems to be no more useful than the regular GES for delegating NP-hard functions.

One could ask why we would even be interested in delegating NP-hard problems to a quantum computer, given that we do not expect quantum computers to be able to solve such problems in polynomial time [1]. First of all, from a theoretical perspective, note that in the QGES formalism we are not limiting the server to polynomial-time quantum computations, but instead assuming that it has unbounded computational power. Therefore, the way to view this result is not as "how can a client blindly delegate the evaluation of NP-hard functions to a quantum computer?" but as "can quantum communication help in blindly delegating the evaluation of NP-hard functions to an unbounded server?".

From a practical perspective, while we do not believe that quantum computers can solve NP-complete problems in polynomial time, they could, in principle, solve such problems quadratically faster than classical computers, thanks to Grover's algorithm [33]. Even

[8] It should be noted that our upper bound on the power of QGES schemes also holds if the client is restricted to BPP computations (as is the case in UBQC), since $\mathsf{BPP} \subseteq \mathsf{BQP}$.

though the speedup of Grover's algorithm is only quadratic, from (say) 2^n to $2^{n/2}$, our result is only concerned with the length of the computation performed on the client side, and therefore applies to Grover's algorithm just as it would to a quantum algorithm achieving exponential speedup. In fact, as is mentioned in [4], there are NP-complete problems for which quantum computers provide a superpolynomial speedup, at least with respect to the best known classical algorithms. Our no-go theorem indicates that clients cannot exploit such speedups by delegating the computation to the server, even when allowing some quantum communication, if we also want to keep their inputs hidden in an information-theoretic sense.

Proofs of these results can be found in the full version of our paper [7].

1.2 Related work

As mentioned, the problem of computing on encrypted data was first considered by Rivest, Adleman and Dertouzos [52], which then led to the development of *homomorphic encryption* and eventually to fully homomorphic encryption with Gentry's scheme [29]. Since then there have been many other FHE protocols relying on more standard cryptographic assumptions and having more practical requirements [17, 16, 57].

While FHE is similar to the GES in many respects, there are also significant differences. For starters, FHE protocols have only one round of interaction between the client and the server, whereas a GES allows for polynomially many rounds. Additionally, the GES assumes the server is computationally unbounded and hence requires information-theoretic security. In contrast, FHE relies on computational security. More precisely FHE schemes have semantic security against polynomial-time (quantum) algorithms [29].

The problem of *quantum* computing on encrypted data was introduced by Childs [21] and Arrighi and Salvail [12]. Further development eventually led to UBQC [19, 9] and a scheme of Broadbent [18]. The latter was followed by the construction of the first schemes for quantum fully homomorphic encryption (QFHE) [20, 24]. For a review of blind quantum computing and related protocols see [26].

In the QFHE schemes of [20, 24], the server is a polynomial-time quantum computer and the client has some quantum capabilities of its own, although it is not able to perform universal quantum computations. Both the size of the exchanged messages and the number of operations of the client are polynomial in the size of the input. More recently, QFHE schemes have been proposed in which the client is completely classical [43, 15]. Similar to FHE, these protocols rely on computational assumptions for security [10] and involve one round of back and forth interaction between the client and the server. QFHE with information-theoretic security (and a computationally unbounded server) has been considered by Yu et al. in [59], where it is shown that it is impossible to have such a scheme for arbitrary unitary operations (or even arbitrary reversible classical computations). This result was later reproved by Newmann and Shi using quantum random-access codes [49]. In relation to our work, QFHE with information-theoretic security can be viewed as a one-round QGES in which the server responds with a quantum message. The complexity-theoretic upper bound we prove for QGES computable functions would then apply to QFHE as well (provided that in QFHE we only leak the size of the input to the server), since our proof allows a quantum message from the server just as well as a classical message.

The possibility of a classical client delegating a blind computation to a quantum server was considered by Morimae and Koshiba [48]. They showed that such a protocol in which the client leaks no information about its input to the server and there is only one round of interaction leads to $\mathsf{BQP} \subseteq \mathsf{NP}$, considered an unlikely containment. We consider the more general setting of a GES for BQP functions, where the number of rounds can be polynomial

in the size of the input and we allow the encryption to leak the size of the input. In fact, the question of whether a GES, as defined in Abadi et al. [8], can exist for quantum computations was raised before by Dunjko and Kashefi [25].

1.3 Future work

As we remarked in Section 1.1, in the case of decision problems, the existence of a GES with bounded communication, for polynomial-time quantum computations, leads to the inclusion $\mathsf{BQP} \subset \mathsf{MA}/\mathsf{O}(n^d)$. We argue that this containment is unlikely based on the existence of an oracle separating the two complexity classes. A natural extension of this result would be to prove an oracle separation between BQP and $\mathsf{NP}/\mathsf{poly}$. This would provide more compelling evidence that a GES for quantum computations cannot exist.

In the case of sampling problems, we showed that a GES for BosonSampling implies the existence of circuits of size $2^{n-\Omega\left(\frac{n}{\log n}\right)}$, making polynomially-sized queries to an $\mathsf{NP}^{\mathsf{NP}}$ oracle, for computing matrix permanents. Can this result be strengthened so as to provide circuits for computing matrix permanents that would be ruled out by the *strong exponential-time hypothesis*? Alternatively, could one use other quantum sampling problems (such as random circuit sampling or IQP problems [35]) to show that having a GES for such a problem leads to a collapse of the polynomial hierarchy?

We also defined the QGES, which extends the GES by allowing the client to send one quantum message to the server, and gave an upper bound for the set of functions that can be delegated using an offline version of such a scheme. The immediate question one could ask is: what upper bound can we give for an online QGES? A related question is: what upper bound can we give for a QGES that allows all of the communication between the client and the server to be quantum? The difficulty in answering both of these questions is that the offline property of the QGES is what allowed us to relate the set of functions that can be delegated to advice classes. Without this property, it seems that a different approach would be needed to provide a complexity-theoretic upper bound.

Another direction that can be explored has to do with the size of the quantum communication between the client and the server. In a QGES in which the client's quantum message is logarithmic or poly-logarithmic in the size of the input (while the classical communication is still polynomial), is it still possible to delegate BQP functions to the server? Of course, this question only makes sense if we assume that the client is not able to perform BQP computations itself.

References

1. Scott Aaronson. Limits on efficient computation in the physical world. *arXiv preprint*, 2004. arXiv:quant-ph/0412143.
2. Scott Aaronson. BQP and the polynomial hierarchy. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 141–150, New York, NY, USA, 2010. ACM. doi:10.1145/1806689.1806711.
3. Scott Aaronson. The Equivalence of Sampling and Searching. In *Proceedings of the 6th International Conference on Computer Science: Theory and Applications*, CSR'11, pages 1–14, Berlin, Heidelberg, 2011. Springer-Verlag. URL: http://dl.acm.org/citation.cfm?id=2017990.2017991.
4. Scott Aaronson. $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$, 2017. URL: https://www.scottaaronson.com/papers/pnp.pdf.
5. Scott Aaronson and Andris Ambainis. Forrelation: A Problem That Optimally Separates Quantum from Classical Computing. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 307–316, New York, NY, USA, 2015. ACM. doi:10.1145/2746539.2746547.

6 Scott Aaronson and Alex Arkhipov. The Computational Complexity of Linear Optics. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC '11, pages 333–342, New York, NY, USA, 2011. ACM. doi:10.1145/1993636.1993682.

7 Scott Aaronson, Alexandru Cojocaru, Alexandru Gheorghiu, and Elham Kashefi. Complexity-theoretic limitations on blind delegated quantum computation. *arXiv preprint*, 2017. arXiv:1704.08482.

8 M. Abadi, J. Feigenbaum, and J. Kilian. On Hiding Information from an Oracle. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 195–203, New York, NY, USA, 1987. ACM. doi:10.1145/28395.28417.

9 Dorit Aharonov, Michael Ben-Or, and Elad Eban. Interactive Proofs For Quantum Computations. In *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 453–469, 2010. URL: http://conference.itcs.tsinghua.edu.cn/ICS2010/content/papers/35.html.

10 Gorjan Alagic, Anne Broadbent, Bill Fefferman, Tommaso Gagliardoni, Christian Schaffner, and Michael St. Jules. *Computational Security of Quantum Encryption*, pages 47–71. Springer International Publishing, Cham, 2016. doi:10.1007/978-3-319-49175-2_3.

11 Joël Alwen, Manuel Barbosa, Pooya Farshim, Rosario Gennaro, S. Dov Gordon, Stefano Tessaro, and David A. Wilson. *On the Relationship between Functional Encryption, Obfuscation, and Fully Homomorphic Encryption*, pages 65–84. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. doi:10.1007/978-3-642-45239-0_5.

12 Pablo Arrighi and Louis Salvail. Blind quantum computation. *International Journal of Quantum Information*, 04(05):883–898, 2006. doi:10.1142/S0219749906002171.

13 Ethan Bernstein and Umesh Vazirani. Quantum Complexity Theory. *SIAM J. Comput.*, 26(5):1411–1473, October 1997. doi:10.1137/S0097539796300921.

14 Andreas Björklund. Below All Subsets for Some Permutational Counting Problems . In Rasmus Pagh, editor, *15th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2016)*, volume 53 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:11, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.SWAT.2016.17.

15 Zvika Brakerski. Quantum FHE (Almost) As Secure as Classical. Cryptology ePrint Archive, Report 2018/338, 2018. URL: https://eprint.iacr.org/2018/338.

16 Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) Fully Homomorphic Encryption Without Bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, New York, NY, USA, 2012. ACM. doi:10.1145/2090236.2090262.

17 Zvika Brakerski and Vinod Vaikuntanathan. Efficient Fully Homomorphic Encryption from (Standard) LWE. In *Proceedings of the 2011 IEEE 52Nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 97–106, Washington, DC, USA, 2011. IEEE Computer Society. doi:10.1109/FOCS.2011.12.

18 Anne Broadbent. Delegating private quantum computations. *Canadian Journal of Physics*, 93(9):941–946, 2015. doi:10.1139/cjp-2015-0030.

19 Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. In *Proceedings of the 50th Annual Symposium on Foundations of Computer Science*, FOCS '09, pages 517–526. IEEE Computer Society, 2009. doi:10.1109/FOCS.2009.36.

20 Anne Broadbent and Stacey Jeffery. Quantum Homomorphic Encryption for Circuits of Low T-gate Complexity. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 609–629, 2015. doi:10.1007/978-3-662-48000-7_30.

21 Andrew M. Childs. Secure Assisted Quantum Computation. *Quantum Info. Comput.*, 5(6):456–466, September 2005. URL: http://dl.acm.org/citation.cfm?id=2011670.2011674.

22 Ivan Damgård, Jens Groth, and Gorm Salomonsen. *The Theory and Implementation of an Electronic Voting System*, pages 77–99. Springer US, Boston, MA, 2003. doi:10.1007/978-1-4615-0239-5_6.

23 J Niel De Beaudrap, Richard Cleve, John Watrous, et al. Sharp quantum versus classical query complexity separations. *Algorithmica*, 34(4):449–461, 2002.

24 Yfke Dulek, Christian Schaffner, and Florian Speelman. *Quantum Homomorphic Encryption for Polynomial-Sized Circuits*, pages 3–32. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. doi:10.1007/978-3-662-53015-3_1.

25 Vedran Dunjko and Elham Kashefi. Blind quantum computing with two almost identical states, 2016. arXiv:1604.01586.

26 Joseph F Fitzsimons. Private quantum computation: an introduction to blind quantum computing and related protocols. *npj Quantum Information*, 3(1):23, 2017.

27 Joseph F Fitzsimons and Elham Kashefi. Unconditionally verifiable blind quantum computation. *Physical Review A*, 96(1):012303, 2017.

28 Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate Indistinguishability Obfuscation and Functional Encryption for All Circuits. In *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, FOCS '13, pages 40–49, Washington, DC, USA, 2013. IEEE Computer Society. doi:10.1109/FOCS.2013.13.

29 Craig Gentry. Fully Homomorphic Encryption Using Ideal Lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM. doi:10.1145/1536414.1536440.

30 Alexandru Gheorghiu, Theodoros Kapourniotis, and Elham Kashefi. Verification of quantum computation: An overview of existing approaches. *Theory of computing systems*, pages 1–94, 2018.

31 Vittorio Giovannetti, Lorenzo Maccone, Tomoyuki Morimae, and Terry G. Rudolph. Efficient Universal Blind Quantum Computation. *Phys. Rev. Lett.*, 111:230501, December 2013. doi:10.1103/PhysRevLett.111.230501.

32 Thore Graepel, Kristin Lauter, and Michael Naehrig. ML Confidential: Machine Learning on Encrypted Data. In *Proceedings of the 15th International Conference on Information Security and Cryptology*, ICISC'12, pages 1–21, Berlin, Heidelberg, 2013. Springer-Verlag. doi:10.1007/978-3-642-37682-5_1.

33 Lov K. Grover. A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM. doi:10.1145/237814.237866.

34 Dominik Hangleiter, Martin Kliesch, Jens Eisert, and Christian Gogolin. Sample complexity of device-independently certified" quantum supremacy". *arXiv preprint*, 2018. arXiv:1812.01023.

35 Aram W Harrow and Ashley Montanaro. Quantum computational supremacy. *Nature*, 549(7671):203, 2017.

36 Thomas Jansen. On the Black-Box Complexity of Example Functions: The Real Jump Function. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, FOGA '15, pages 16–24, New York, NY, USA, 2015. ACM. doi:10.1145/2725494.2725507.

37 Arjan Jeckmans, Andreas Peter, and Pieter Hartel. *Efficient Privacy-Enhanced Familiarity-Based Recommender System*, pages 400–417. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. doi:10.1007/978-3-642-40203-6_23.

38 Elham Kashefi, Luka Music, and Petros Wallden. The Quantum Cut-and-Choose Technique and Quantum Two-Party Computation, 2017. arXiv:1703.03754.

39 Elham Kashefi and Petros Wallden. Garbled quantum computation. *Cryptography*, 1(1):6, 2017.

40 Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2014.

41 Kristin E. Lauter. Practical Applications of Homomorphic Encryption. In *Proceedings of the 2012 ACM Workshop on Cloud Computing Security Workshop*, CCSW '12, pages 57–58, New York, NY, USA, 2012. ACM. doi:10.1145/2381913.2381924.

42 Sheng-Kai Liao, Wen-Qi Cai, Wei-Yue Liu, Liang Zhang, Yang Li, Ji-Gang Ren, Juan Yin, Qi Shen, Yuan Cao, Zheng-Ping Li, et al. Satellite-to-ground quantum key distribution. *Nature*, 549(7670):43, 2017.

43 Urmila Mahadev. Classical homomorphic encryption for quantum circuits. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 332–338. IEEE, 2018.

44 Atul Mantri, Tommaso F Demarie, and Joseph F Fitzsimons. Universality of quantum computation with cluster states and (X, Y)-plane measurements. *Scientific reports*, 7:42861, 2017.

45 Atul Mantri, Carlos A. Pérez-Delgado, and Joseph F. Fitzsimons. Optimal Blind Quantum Computation. *Phys. Rev. Lett.*, 111:230502, December 2013. doi:10.1103/PhysRevLett.111.230502.

46 Tomoyuki Morimae, Vedran Dunjko, and Elham Kashefi. Ground State Blind Quantum Computation on AKLT State. *Quantum Info. Comput.*, 15(3-4):200–234, March 2015. URL: http://dl.acm.org/citation.cfm?id=2871393.2871395.

47 Tomoyuki Morimae and Keisuke Fujii. Blind quantum computation protocol in which Alice only makes measurements. *Phys. Rev. A*, 87:050301, May 2013. doi:10.1103/PhysRevA.87.050301.

48 Tomoyuki Morimae and Takeshi Koshiba. Impossibility of perfectly-secure delegated quantum computing for classical client, 2014. arXiv:1407.1636.

49 Michael Newman and Yaoyun Shi. Limitations on Transversal Computation through Quantum Homomorphic Encryption. *Quantum Information and Computation*, 18:0927–0948, 2018.

50 John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, 2018.

51 Ran Raz and Avishay Tal. Oracle Separation of BQP and PH. *eccc preprint TR18-107*, 2018. Eprint:https://eccc.weizmann.ac.il/report/2018/107/.

52 Ronald L Rivest, Len Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, 4(11):169–180, 1978.

53 Herbert John Ryser. *Combinatorial mathematics*, volume 14. JSTOR, 1963.

54 Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Review*, 41(2):303–332, 1999. doi:10.1137/S0036144598347011.

55 Daniel R. Simon. On the Power of Quantum Computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997. doi:10.1137/S0097539796298637.

56 Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991. doi:10.1137/0220053.

57 Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully Homomorphic Encryption over the Integers. In *Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'10, pages 24–43, Berlin, Heidelberg, 2010. Springer-Verlag. doi:10.1007/978-3-642-13190-5_2.

58 Chee K. Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, 26(3):287–300, 1983. doi:10.1016/0304-3975(83)90020-8.

59 Li Yu, Carlos A. Pérez-Delgado, and Joseph F. Fitzsimons. Limitations on information-theoretically-secure quantum homomorphic encryption. *Phys. Rev. A*, 90:050303, November 2014. doi:10.1103/PhysRevA.90.050303.

Faster Algorithms for All-Pairs Bounded Min-Cuts

Amir Abboud
IBM Almaden Research Center, California, USA
amir.abboud@ibm.com

Loukas Georgiadis
University of Ioannina, Greece
loukas@cs.uoi.gr

Giuseppe F. Italiano
LUISS University, Rome, Italy
gitaliano@luiss.it

Robert Krauthgamer
Weizmann Institute of Science, Israel
robert.krauthgamer@weizmann.ac.il

Nikos Parotsidis
University of Copenhagen, Denmark
nipa@di.ku.dk

Ohad Trabelsi
Weizmann Institute of Science, Israel
ohad.trabelsi@weizmann.ac.il

Przemysław Uznański
University of Wrocław, Poland
puznanski@cs.uni.wroc.pl

Daniel Wolleb-Graf
ETH Zürich, Switzerland
daniel.graf@inf.ethz.ch

Abstract

The All-Pairs Min-Cut problem (aka All-Pairs Max-Flow) asks to compute a minimum s-t cut (or just its value) for all pairs of vertices s, t. We study this problem in *directed graphs* with *unit* edge/vertex capacities (corresponding to edge/vertex connectivity). Our focus is on the *k-bounded* case, where the algorithm has to find all pairs with min-cut value less than k, and report only those. The most basic case $k = 1$ is the Transitive Closure (TC) problem, which can be solved in graphs with n vertices and m edges in time $O(mn)$ combinatorially, and in time $O(n^\omega)$ where $\omega < 2.38$ is the matrix-multiplication exponent. These time bounds are conjectured to be optimal.

We present new algorithms and conditional lower bounds that advance the frontier for larger k, as follows:

- A randomized algorithm for *vertex capacities* that runs in time $O((nk)^\omega)$. This is only a factor k^ω away from the TC bound, and nearly matches it for all $k = n^{o(1)}$.
- Two deterministic algorithms for *edge capacities* (which is more general) that work in DAGs and further reports a minimum cut for each pair. The first algorithm is combinatorial (does not involve matrix multiplication) and runs in time $O(2^{O(k^2)} \cdot mn)$. The second algorithm can be faster on dense DAGs and runs in time $O((k \log n)^{4^{k+o(k)}} \cdot n^\omega)$. Previously, Georgiadis et al. [ICALP 2017], could match the TC bound (up to $n^{o(1)}$ factors) only when $k = 2$, and now our two algorithms match it for all $k = o(\sqrt{\log n})$ and $k = o(\log \log n)$.
- The first super-cubic lower bound of $n^{\omega-1-o(1)}k^2$ time under the 4-Clique conjecture, which holds even in the simplest case of DAGs with unit vertex capacities. It improves on the previous (SETH-based) lower bounds even in the unbounded setting $k = n$. For combinatorial algorithms, our reduction implies an $n^{2-o(1)}k^2$ conditional lower bound. Thus, we identify new settings where the complexity of the problem is (conditionally) higher than that of TC.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 7; pp. 7:1–7:15

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

Our three sets of results are obtained via different techniques. The first one adapts the network coding method of Cheung, Lau, and Leung [SICOMP 2013] to vertex-capacitated digraphs. The second set exploits new insights on the structure of latest cuts together with suitable algebraic tools. The lower bounds arise from a novel reduction of a different structure than the SETH-based constructions.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Network flows

Keywords and phrases All-pairs min-cut, k-reachability, network coding, Directed graphs, fine-grained complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.7

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at arXiv:1807.05803.

Funding *Robert Krauthgamer*: Work supported in part by ONR Award N00014-18-1-2364, Israel Science Foundation grant #1086/18, a Minerva Foundation grant, and a Google Faculty Research Award.
Nikos Parotsidis: The author is supported by Grant Number 16582, Basic Algorithms Research Copenhagen (BARC), from the VILLUM Foundation.
Ohad Trabelsi: Work partly done at IBM Almaden Research Center, USA.

Acknowledgements We thank Paweł Gawrychowski, Mohsen Ghaffari, Atri Rudra and Peter Widmayer for the valuable discussions on this problem.

1 Introduction

Connectivity-related problems are some of the most well-studied problems in graph theory and algorithms, and have been thoroughly investigated in the literature. Given a directed graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges,[1] perhaps the most fundamental such problem is to compute a *minimum s-t cut*, i.e., a set of edges E' of minimum-cardinality such that t is not reachable from s in $G \setminus E'$. This minimum s-t cut problem is well-known to be equivalent to maximum s-t flow, as they have the exact same value [17]. Currently, the fastest algorithms for this problem run in time $\tilde{O}(m\sqrt{n}\log^{O(1)} U)$ [24] and $\tilde{O}(m^{10/7}U^{1/7})$ (faster for sparse graphs) [26], where U is the maximum edge capacity (aka weight).[2]

The central problem of study in this paper is All-Pairs Min-Cut (also known as All-Pairs Max-Flow), where the input is a digraph $G = (V, E)$ and the goal is to compute the minimum s-t cut value for all $s, t \in V$. All our graphs will have *unit* edge/vertex capacities, in which case the value of the minimum s-t cut is just the maximum number of disjoint paths from s to t (aka edge/vertex connectivity), by [28]. We will consider a few variants: vertex capacities vs. edge capacities,[3] reporting only the value vs. the cut itself (a witness), or a general digraph vs. a directed acyclic graph (DAG). For all these variants, we will be interested in the *k-bounded* version (aka *bounded min-cuts*, hence the title of the paper) where the algorithm needs to find which minimum s-t cuts have value less than a given parameter k, and report

[1] We sometimes use arcs when referring to directed edges, or use nodes instead of vertices.
[2] The notation $\tilde{O}(\cdot)$ hides polylogarithmic factors.
[3] The folklore reduction where each vertex v is replaced by two vertices connected by an edge $v_{in} \to v_{out}$ shows that in all our problems, vertex capacities are no harder (and perhaps easier) than edge capacities. Notice that this is only true for directed graphs.

only those. Put differently, the goal is to compute, for every $s, t \in V$, the minimum between k and the actual minimum s-t cut value. Nonetheless, some of our results (the lower bounds) are of interest even without this restriction.

The time complexity of these problems should be compared against the fundamental special case that lies at their core – the Transitive Closure problem (aka All-Pairs Reachability), which is known to be time-equivalent to Boolean Matrix Multiplication, and in some sense, to Triangle Detection [34]. This is the case $k = 1$, and it can be solved in time $O(\min\{mn/\log n, n^\omega\})$, where $\omega < 2.38$ is the matrix-multiplication exponent [14, 23, 33, 11]; the latter term is asymptotically better for dense graphs, but it is not *combinatorial*.[4] This time bound is conjectured to be optimal for Transitive Closure, which can be viewed as a conditional lower bound for All-Pairs Min-Cut; but can we achieve this time bound algorithmically, or is All-Pairs Min-Cut a harder problem?

The naive strategy for solving All-Pairs Min-Cut is to execute a minimum s-t cut algorithm $O(n^2)$ times, with total running time $\tilde{O}(n^2 m^{10/7})$ [26] or $\tilde{O}(n^{2.5}m)$ [24]. For not-too-dense graphs, there is a faster randomized algorithm of Cheung, Lau, and Leung [13] that runs in time $O(m^\omega)$. For smaller k, some better bounds are known. First, observe that a minimum s-t cut can be found via k iterations of the Ford-Fulkerson algorithm [17] in time $O(km)$, which gives a total bound of $O(n^2mk)$. Another randomized algorithm of [13] runs in better time $O(mnk^{\omega-1})$ but it works only in DAGs. Notice that the latter bound matches the running time of Transitive Closure if the graphs are sparse enough. For the case $k = 2$, Georgiadis et al. [18] achieved the same running time as Transitive Closure up to sub-polynomial factor $n^{o(1)}$ in all settings, by devising two deterministic algorithms, whose running times are $\tilde{O}(mn)$ and $\tilde{O}(n^\omega)$.

Other than the lower bound from Transitive Closure, the main previously known result is from [21], which showed that under the Strong Exponential Time Hypothesis (SETH),[5] All-Pairs Min-Cut requires, up to sub-polynomial factors, time $\Omega(mn)$ in unit edge capacitated digraphs of any edge density, and even in the simpler case of (unit) vertex capacities and of DAGs. As a function of k their lower bound becomes $\Omega(n^{2-o(1)}k)$ [21]. Combining the two, we have a conditional lower bound of $(n^2k + n^\omega)^{1-o(1)}$.

Related Work. There are many other results related to the edge capacitated All-Pairs Min-Cut problem, let us mention a few. Other than DAGs, the problem has also been considered in the special cases of planar digraphs [6, 22], sparse digraphs and digraphs with bounded treewidth [6].

In *undirected* graphs, the problem was studied extensively following the seminal work of Gomory and Hu [19] in 1961, which introduced a representation of All-Pairs Min-Cuts via a weighted tree, commonly called a Gomory-Hu tree, and further showed how to compute it using $n - 1$ executions of maximum s-t flow. Bhalgat et al. [8] designed an algorithm that computes a Gomory-Hu tree in unit edge capacitated undirected graphs in $\tilde{O}(mn)$ time, and this upper bound was recently improved [4]. The case of bounded min-cuts (small k) in undirected graphs was studied by Hariharan et al. [20], motivated in part by applications in practical scenarios. The fastest running time for this problem is $\tilde{O}(mk)$ [31], achieved by combining results from [20] and [8]. On the negative side, there is an $n^{3-o(1)}$ lower bound for All-Pairs Min-Cut in sparse *capacitated* digraphs [21], and very recently, a similar lower bound was shown for *undirected* graphs with vertex capacities [4].

[4] Combinatorial is an informal term to describe algorithms that do not rely on fast matrix-multiplication algorithms, which are infamous for being impractical. See [1, 3] for further discussions.

[5] These lower bounds hold even under the weaker assumption that the 3-Orthogonal Vectors problem requires $n^{3-o(1)}$ time.

1.1 Our Contribution

The goal of this work is to reduce the gaps in our understanding of the All-Pairs Min-Cut problem (see Table 1 for a list of known and new results). In particular, we are motivated by three high-level questions. First, how large can k be while keeping the time complexity the same as Transitive Closure? Second, could the problem be solved in cubic time (or faster) in all settings? Currently no $\Omega(n^{3+\varepsilon})$ lower bound is known even in the hardest settings of the problem (capacitated, dense, general graphs). And third, can the actual cuts (witnesses) be reported in the same amount of time it takes to only report their values? Some of the previous techniques, such as those of [13], cannot do that.

New Algorithms. Our first result is a randomized algorithm that solves the k-bounded version of All-Pairs Min-Cut in a digraph with *unit vertex capacities* in time $O((nk)^\omega)$. This upper bound is only a factor k^ω away from that of Transitive Closure, and thus matches it up to polynomial factors for any $k = n^{o(1)}$. Moreover, any poly(n)-factor improvement over our upper bound would imply a breakthrough for Transitive Closure (and many other problems). Our algorithm builds on the network-coding method of [13], and in effect adapts this method to the easier setting of vertex capacities, to achieve a better running time than what is known for unit edge capacities. This algorithm is actually more general: Given a digraph $G = (V, E)$ with unit vertex capacities, two subsets $S, T \subseteq V$ and $k > 0$, it computes for all $s \in S, t \in T$ the minimum s-t cut value if this value is less than k, all in time $O((n + (|S| + |T|)k)^\omega + |S||T|k^\omega)$. Due to space constraints, we overview these results in Section 3.1, with full details in the full version.

Three weaknesses of this algorithm and the ones by Cheung et al. [13] are that they do not return the actual cuts, they are randomized, and they are not combinatorial. Our next set of algorithmic results deals with these issues. More specifically, we present two deterministic algorithms for DAGs with unit edge (or vertex) capacities that compute, for every $s, t \in V$, an actual minimum s-t cut if its value is less than k. The first algorithm is *combinatorial* (i.e., it does not involve matrix multiplication) and runs in time $O(2^{O(k^2)} \cdot mn)$. The second algorithm can be faster on dense DAGs and runs in time $O((k \log n)^{4^k + o(k)} \cdot n^\omega)$. These algorithms extend the results of Georgiadis et al. [18], which matched the running time of Transitive Closure up to $n^{o(1)}$ factors, from just $k = 2$ to any $k = o(\sqrt{\log n})$ (in the first case) and $k = o(\log \log n)$ (in the second case). We give an overview of these algorithms in Section 3.2, and the formal results are in the full version.

New Lower Bounds. Finally, we present conditional lower bounds for our problem, the k-bounded version of All-Pairs Min-Cut. As a result, we identify new settings where the problem is harder than Transitive Closure, and provide the first evidence that the problem cannot be solved in cubic time. Technically, the main novelty here is a reduction from the 4-Clique problem. It implies lower bounds that apply to the basic setting of DAGs with unit vertex capacities, and therefore immediately apply also to more general settings, such as edge capacities, capacitated inputs, and general digraphs, and they in fact improve over previous lower bounds [5, 21] in all these settings.[6] We prove the following theorem in Section 4.

▶ **Theorem 1.1.** *If for some fixed $\varepsilon > 0$ and any $k \in [n^{1/2}, n]$, the k-bounded version of All-Pairs Min-Cut can be solved on DAGs with unit vertex capacities in time $O((n^{\omega-1}k^2)^{1-\varepsilon})$, then 4-Clique can be solved in time $O(n^{\omega+1-\delta})$ for some $\delta = \delta(\varepsilon) > 0$.*

[6] It is unclear if our new reduction can be combined with the ideas in [4] to improve the lower bounds in the seemingly easier case of undirected graphs with vertex capacities.

Moreover, if for some fixed $\varepsilon > 0$ and any $k \in [n^{1/2}, n]$ that version of All-Pairs Min-Cut can be solved combinatorially in time $O((n^2k^2)^{1-\varepsilon})$, then 4-Clique can be solved combinatorially in time $O(n^{4-\delta})$ for some $\delta = \delta(\varepsilon) > 0$.

To appreciate the new bounds, consider first the case $k = n$, which is equivalent to not restricting k. The previous lower bound, under SETH, is $n^{3-o(1)}$ and ours is larger by a factor of $n^{\omega-2}$. For combinatorial algorithms, our lower bound is $n^{4-o(1)}$, which is essentially the largest possible lower bound one can prove without a major breakthrough in fine-grained complexity. This is because the naive algorithm for All-Pairs Min-Cuts is to invoke an algorithm for Max-Flow $O(n^2)$ times, hence a lower bound larger than $\Omega(n^4)$ for our problem would imply the first non-trivial lower bound for minimum s-t cut. The latter is perhaps the biggest open question in fine-grained complexity, and in fact many experts believe that near-linear time algorithms for minimum s-t cut do exist, and can even be considered "combinatorial" in the sense that they do not involve the infamous inefficiencies of fast matrix multiplication. If such algorithms for minimum s-t cut do exist, then our lower bound is tight.

Our lower bound shows that as k exceeds $n^{1/2-o(1)}$, the time complexity of k-bounded of All-Pairs Min-Cut exceeds that of Transitive Closure by polynomial factors. The lower bound is super-cubic whenever $k \geq n^{2-\omega/2+\varepsilon}$.

Table 1 Summary of new and known results. Unless mentioned otherwise, all upper and lower bounds hold both for unit edge capacities and for unit vertex capacitities.

Time		Input	Output	Reference
$O(mn), \tilde{O}(n^\omega)$	deterministic	digraphs	cuts, only $k = 2$	[18]
$O(n^2mk)$	deterministic	digraphs	cuts	[17]
$O(m^\omega)$	randomized	digraphs	cut values	[13]
$O(mnk^{\omega-1})$	randomized	DAGs	cut values	[13]
$O((nk)^\omega)$	randomized, vertex capacities	digraphs	cut values	full version
$2^{O(k^2)}mn$	deterministic	DAGs	cuts	full version
$(k \log n)^{4^k+o(k)} \cdot n^\omega$	deterministic	DAGs	cuts	full version
$(mn + n^\omega)^{1-o(1)}$	based on Transitive Closure	DAGs	cut values	
$n^{2-o(1)}k$	based on SETH	DAGs	cut values	[21]
$n^{\omega-1-o(1)}k^2$	based on 4-Clique	DAGs	cut values	Theorem 1.1

2 Preliminaries

We start with some terminology and well-known results on graphs and cuts. Next we will briefly introduce the main algebraic tools that will be used throughout the paper. We note that although we are interested in solving the k-bounded All-Pairs Min-Cut problem, where we wish to find the all-pairs min-cuts of size at most $k-1$, for the sake of using simpler notation we compute the min-cuts of size at most k (instead of less than k) solving this way the $(k+1)$-bounded All-Pairs Min-Cut problem.

Directed graphs. The input of our problem consists of an integer $k \geq 1$ and a *directed graph*, digraph for short, $G = (V, A)$ with $n := |V|$ *vertices* and $m := |A|$ *arcs*. Every arc $a = (u, v) \in A$ consists of a tail $u \in V$ and a head $v \in V$. By $G[S]$, we denote the subgraph of G *induced* by the set of vertices S, formally $G[S] = (S, A \cap (S \times S))$. By $N^+(v)$, we denote

the *out-neighborhood* of v consisting of all the heads of the arcs leaving v. We denote by $\text{outdeg}(v)$ the number of outgoing arcs from v. All our results extend to multi-digraphs, where each pair of vertices can be connected with multiple (*parallel*) arcs. For parallel arcs, we always refer to each arc individually, as if each arc had a unique identifier. So whenever we refer to a set of arcs, we refer to the set of their unique identifiers, i.e., without collapsing parallel arcs, like in a multi-set.

Flows and cuts. We follow the notation used by Ford and Fulkerson [17]. Let $G = (V, A)$ be a digraph, where each arc a has a nonnegative capacity $c(a)$. For a pair of vertices s and t, an s-t flow of G is a function f on A such that $0 \leq f(a) \leq c(a)$, and for every vertex $v \neq s, t$ the incoming flow is equal to outgoing flow, i.e., $\sum_{(u,v)\in A} f(u, v) = \sum_{(v,u)\in A} f(v, u)$. If G has vertex capacities as well, then f must also satisfy $\sum_{(u,v)\in A} f(u, v) \leq c(v)$ for every $v \neq s, t$, where $c(v)$ is the capacity of v. The value of the flow is defined as $|f| = \sum_{(s,v)\in A} f(s, v)$. We denote the existence of a path from s to t by $s \rightsquigarrow t$ and by $s \not\rightsquigarrow t$ the lack of such a path. Any set $M \subseteq A$ is an *s-t-cut* if $s \not\rightsquigarrow t$ in $G \setminus M$. M is a *minimal* s-t-cut if no proper subset of M is s-t-cut. For an s-t-cut M, we say that its *source side* is $S_M = \{x \mid s \rightsquigarrow x \text{ in } G \setminus M\}$ and its *target side* is $T_M = \{x \mid x \rightsquigarrow t \text{ in } G \setminus M\}$. We also refer to the source side and the target side as *s-reachable* and *t-reaching*, respectively. An s-t k-cut is a minimal cut of size k. A set $\mathcal{M}$ of s-t cuts of size at most k is called a set of s-t $\leq k$-cuts. We can define vertex cuts analogously.

Order of cuts. An s-t cut M is *later* (respectively *earlier*) than an s-t cut M' if and only if $T_M \subseteq T_{M'}$ (resp. $S_M \subseteq S_{M'}$), and we denote it $M \geq M'$ (resp. $M \leq M'$). Note that those relations are not necessarily complementary if the cuts are not minimal (see Figure 1 for an example).

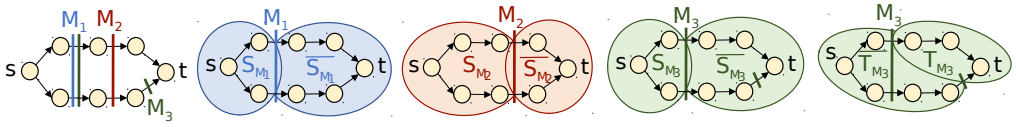


Figure 1 A digraph with three s-t-cuts M_1, M_2, M_3. While M_1 and M_2 are minimal, M_3 is not. Hence, the source side and target side differ only for M_3. This illustrates that the earlier and later orders might not be symmetric for non-minimal cuts. We have $M_3 < M_2$ yet $M_2 \not> M_3$ (and also $M_3 \leq M_2$ yet $M_2 \not\geq M_3$). Additionally, $M_1 \not< M_3$ yet $M_3 > M_1$ (yet both $M_1 \leq M_3$ and $M_3 \geq M_1$).

We make these inequalities strict (i.e., '$>$' or '$<$') whenever the inclusions are proper. We compare a cut M and an arc a by defining $a > M$ whenever both endpoints of a are in T_M. Additionally, $a \geq M$ includes the case where $a \in M$. Definitions of the relations '$\leq$' and '$<$' follow by symmetry. We refer to Figure 2 for illustrations.

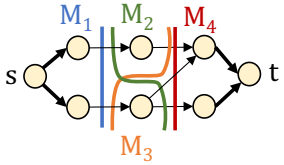


Figure 2 A digraph with several s-t cuts. Bold arcs represent parallel arcs which are too expensive to cut. M_1 is the earliest s-t min-cut and M_3 is the latest s-t min-cut. M_2 is later than M_1, but M_2 is not s-t-latest, as M_4 is later and not larger than M_2.

This partial order of cuts also allows us to define cuts that are extremal with respect to all other s-t cuts in the following sense:

▶ **Definition 2.1** (s-t-latest cuts [27]). *An s-t cut is s-t-*latest *(resp. s-t-*earliest*) if and only if there is no later (resp. earlier) s-t cut of smaller or equal size.*

Informally speaking, a cut is s-t-latest if we would have to cut through more arcs whenever we would like to cut off fewer vertices. This naturally extends the definition of an s-t-latest *min*-cut as used by Ford and Fulkerson [17, Section 5]. The notion of latest cuts has first been introduced by Marx [27] (under the name of *important* cuts) in the context of fixed-parameter tractable algorithms for multi(way) cut problems, but has been independently studied (or its equivalent formulations), [7, 29]. Since we need both earliest and latest cuts, we do not refer to latest cuts as important cuts. Additionally, we use the term *s-t-extremal* cuts to refer to the union of s-t-earliest and s-t-latest cuts.

3 Overview of Our Algorithmic Approach

3.1 Randomized Algorithms on General Graphs

We will now briefly recap the framework of Cheung et al. [13] as we will modify them later for our purposes. In this framework, edges are encoded as vectors, so that the vector of each edge $e = (u, v)$ is a randomized linear combination of the vectors correspond to edges incoming to u, the source of e. One can compute all these vectors for the whole graph, simultaneously, using some matrix manipulations. The bottleneck is that one has to invert a certain $m \times m$ matrix with an entry for each pair of edges. Just reading the matrix that is output by the inversion requires $\Omega(m^2)$ time, since most entries in the inverted matrix are expected to be nonzero even if the graph is sparse.

To overcome this barrier, while using the same framework, we define the encoding vectors on the nodes rather than the edges. We show that this is sufficient for the vertex-capacitated setting. Then, instead of inverting a large matrix, we need to compute the rank of certain submatrices which becomes the new bottleneck. When k is small enough, this turns out to lead to a significant speed up compared to the running time in [13].

3.2 Deterministic Algorithms with Witnesses on DAGs

Here we deal with the problem of computing certificates for the k-bounded All-Pairs Min-Cut problem. Our contribution here is twofold. We first prove some properties of the structure of the s-t-latest k-cuts and of the s-t-latest $\leq k$-cuts, which might be of independent interest. This gives us some crucial insights on the structure of the cuts, and allows us to develop an algorithmic framework which is used to solve the k-bounded All-Pairs Min-Cut problem. As a second contribution, we exploit our new algorithmic framework in two different ways, leading to two new algorithms which run in $O(mn^{1+o(1)})$ time for $k = o(\sqrt{\log n})$ and in $O(n^{\omega+o(1)})$ time for $k = o(\log \log n)$.

Let $G = (V, A)$ be a DAG. Consider some arbitrary pair of vertices s and t, and any s-t-cut M. For every intermediate vertex v, M must be either a s-v-cut, or a v-t-cut. The knowledge of all s-v and all v-t min-cuts does not allow us to convey enough information for computing an s-t min-cut of size at most k quickly, as illustrated in Figure 3.

However, we are able to compute an s-t min-cut by processing all the *s-v-earliest* cuts and all the *v-t-latest* cuts, of size at most k. We build our approach around this insight. We note that the characterization that we develop is particularly useful, as it has been shown that the number of all earliest/latest u-v $\leq k$-cuts can be upper bounded by $2^{O(k)}$, independently of the size of the graph.

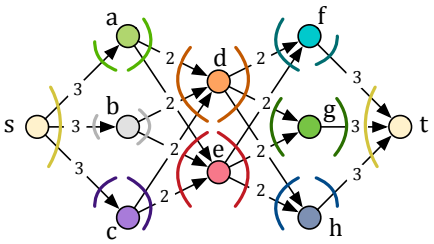


Figure 3 A digraph where each arc appears in at least one s-v or one v-t min-cut. The numbers on the arcs denote the number of parallel arcs. Note that neither of the two s-t min-cuts of size 9 (marked in yellow) are contained within the union of any two s-v or v-t min-cuts. Thus, finding all those min-cuts and trying to combine them in pairs in a divide-and-conquer-style approach is not sufficient to find an s-t min-cut.

For a more precise formulation on how to recover a min-cut (or extremal $\leq k$-cuts) from cuts to and from intermediate vertices, consider the following. Let A_1, A_2 be an *arc split*, that is a partition of the arc set A with the property that any path in G consists of a (possibly empty) sequence of arcs from A_1 followed by a (possibly empty) sequence of arcs from A_2. Assume that for each vertex v we know all the s-v-earliest $\leq k$-cuts in $G_1 = (V, A_1)$ and all the v-t-latest $\leq k$-cuts in $G_2 = (V, A_2)$. We show that a set of arcs M that contains as a subset one s-v-earliest $\leq k$-cut in G_1, or one v-t-latest $\leq k$-cut in G_2 for every v, is a s-t-cut. Moreover, we show that all the s-t-cuts of arcs with the above property include all the s-t-latest $\leq k$-cuts. Hence, in order to identify all s-t-latest $\leq k$ cuts, it is sufficient to identify all sets M with that property. We next describe how we use these structural properties to compute all s-t-extremal $\leq k$-cuts.

We formulate the following combinatorial problem over families of sets, which is independent of graphs and cuts, that we can use to compute all s-t-extremal $\leq k$-cuts. The input to our problem is c families of sets $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_c$, where each family $\mathcal{F}_i$ consists of at most K sets, and each set $F \in \mathcal{F}_i$ contains at most k elements from a universe U. The goal is to compute all minimal subsets $F^* \subset U, |F^*| \leq k$, for which there exists a set $F \in \mathcal{F}_i$ such that $F \subseteq F^*$, for all $1 \leq i \leq c$. We refer to this problem as Witness Superset. To create an instance (s, t, A_1, A_2) of the Witness Superset problem, we set $c = |V|$ and $\mathcal{F}_v$ to be all s-v-earliest $\leq k$-cuts in G_1 and all v-t-latest $\leq k$-cuts in G_2. Informally speaking, the solution to the instance (s, t, A_1, A_2) of the Witness Superset problem picks all sets of arcs that cover at least one earliest or one latest cut for every vertex. In a post-processing step, we filter the solution to the Witness Superset problem on the instance (s, t, A_1, A_2) in order to extract all the s-t-latest $\leq k$-cuts. We follow an analogous process to compute all the s-t-earliest $\leq k$-cuts.

Algorithmic framework. We next define a common algorithmic framework for solving the k-bounded All-Pairs Min-Cut problem, as follows. We pick a partition of the vertices V_1, V_2, such that there is no arc in $V_2 \times V_1$. Such a partition can be trivially computed from a topological order of the input DAG. Let $A_1, A_2, A_{1,2}$ be the sets of arcs in $G[V_1]$, in $G[V_2]$, and $A \cap (V_1 \times V_2)$.

- First, we recursively solve the problem in $G[V_1]$ and in $G[V_2]$. The recursion returns without doing any work whenever the graph is a singleton vertex.

- Second, for each pair of vertices (s,t), such that $s \in V_1$ has an outgoing arc from $A_{1,2}$ and $t \in V_2$, we solve the instance $(s,t,A_{1,2},A_2)$ of Witness Superset. Notice that the only non-empty earliest cuts in $(V,A_{1,2})$ for the pair (x,y) are the arcs $(x,y) \in A_{1,2}$.
- Finally, for each pair of vertices (s,t), such that $s \in V_1, t \in V_2$, we solve the instance $(s,t,A_1,A_{1,2} \cup A_2)$ of Witness Superset.

The Witness Superset problem can be solved naively as follows. Let $\mathcal{F}_v$ be the set of all s-v-earliest $\leq k$-cuts and all v-t-latest $\leq k$-cuts. Assume we have $\mathcal{F}_{v_1}, \mathcal{F}_{v_2}, \ldots, \mathcal{F}_{v_c}$, for all vertices $v_1, v_2, \ldots, v_c$ that are both reachable from s in $(V,A_{1,2})$ and that reach t in (V,A_2). Each of these sets contains $2^{O(k)}$ cuts. We can identify all sets M of arcs that contain at least one cut from each $\mathcal{F}_i$, in time $O(k \cdot (2^{O(k)})^c)$. This yields an algorithm with super-polynomial running time. However, we speed up this naive procedure by applying some judicious pruning, achieving a better running time of $O(c \cdot 2^{O(k^2)} \cdot \text{poly}(k))$, which is polynomial for $k = o(\sqrt{\log n})$. In the following, we sketch the two algorithms that we develop for solving efficiently the k-bounded All-Pairs Min-Cut problem.

Iterative division. For the first algorithm, we process the vertices in reverse topological order. When processing a vertex v, we define $V_1 = \{v\}$ and V_2 to be the set of vertices that appear after v in the topological order. Notice that V_1 has a trivial structure, and we already know all s-t-latest $\leq k$-cuts in $G[V_2]$. In this case, we present an algorithm for solving the instance $(v,t,A_{1,2},A_2)$ of the Witness Superset problem in time $O(2^{O(k^2)} \cdot c \cdot \text{poly}(k))$, where $c = |A_{1,2}|$ is the number of arcs leaving v. We invoke this algorithm for each v-w pair such that $w \in V_2$. For $k = o(\sqrt{\log n})$ this gives an algorithm that runs in time $O(\text{outdeg}(v) \cdot n^{1+o(1)})$ for processing v, and $O(mn^{1+o(1)})$ in total.

Recursive division. For the second algorithm, we recursively partition the set of vertices evenly into sets V_1 and V_2 at each level of the recursion. We first recursively solve the problem in $G[V_1]$ and in $G[V_2]$. Second, we solve the instances $(s,t,A_{1,2},A_2)$ and $(s,t,A_1,A_{1,2} \cup A_2)$ of Witness Superset for all pairs of vertices from $V_1 \times V_2$. Notice that the number of vertices that are both reachable from s in (V,A_1) and reach t in $(V,A_{1,2} \cup A_2)$ can be as high as $O(n)$. This implies that even constructing all $\Theta(n^2)$ instances of the Witness Superset problem, for all s,t, takes $\Omega(n^3)$ time. To overcome this barrier, we take advantage of the power of fast matrix multiplications by applying it into suitably defined matrices of binary codes (codewords). At a very high-level, this approach was used by Fischer and Meyer [16] in their $O(n^\omega)$ time algorithm for transitive closure in DAGs – there the binary codes where of size 1 indicating whether there exists an arc between two vertices.

Algebraic framework. In order to use coordinate-wise boolean matrix multiplication with the entries of the matrices being codewords we first encode all s-t-earliest and all s-t-latest $\leq k$-cuts using binary codes. The bitwise boolean multiplication of such matrices with binary codes in its entries allows a serial combination of both s-v cuts and v-t cuts based on AND operations, and thus allows us to construct a solution based on the OR operation of pairwise AND operations. We show that *superimposed codes* are suitable in our case, i.e., binary codes where sets are represented as bitwise-OR of codewords of objects, and small sets are guaranteed to be encoded uniquely. Superimposed codes provide a unique representation for sets of k elements from a universe of size $\text{poly}(n)$ with codewords of length $\text{poly}(k \log n)$. In this setting, the union of sets translates naturally to bitwise-OR of their codewords.

Tensor product of codes. To achieve our bounds, we compose several identical superimposed codes into a new binary code, so that encoding *set families* with it enables us to solve the corresponding instances of Witness Superset. Our composition has the cost of an exponential increase in the length of the code. Let $\mathcal{F} = F_1, \ldots, F_c$ be the set family that we wish to encode, and let $S_1, \ldots, S_c$ be their superimposed codes in the form of vectors. We construct a c-dimensional array M where $M[i_1, \ldots, i_c] = 1$ iff $S_j[i_j] = 1$, for each $1 \leq j \leq c$. In other words, the resulting code is the tensor product of all superimposed codes. This construction creates enough redundancy so that enough information on the structure of the set families is preserved. Furthermore, we can extract the encoded information from the bitwise-OR of several codewords. The resulting code is of length $O((k \log n)^{O(K)})$, where K is the upperbound on the allowed number of sets in each encoded set family. In our case $K \approx 4^k$, which results to only a logarithmic dependency on n at the price of a doubly-exponential dependency on k, thus making the problem tractable for small values of k.

From slices to Witness Superset. Finally, we show how the Witness Superset can be solved using tensor product of superimposed codes. Consider the notion of cutting the code of dimension K with an axis-parallel hyperplane of dimension $K - 1$. We call this resulting shorter codeword a *slice* of the original codeword. A slice of a tensor product is a tensor product of one dimension less, or an empty set, and a slice of a bitwise-OR of tensor products is as well a bitwise-OR of tensor products (of one dimension less). Thus, taking a slice of the bitwise-OR of the encoding of families of sets is equivalent to removing a particular set from some families and to dropping some other families completely and then encoding these remaining, reduced families. Thus, we can design a non-deterministic algorithm, which at each step of the recursion picks k slices, one slice for each element of the solution we want to output, and then recurses on the bitwise-OR of those slices, reducing the dimension by one in the process. This is always possible, since each element that belongs to a particular solution of Witness Superset satisfies one of the following: it either has a witnessing slice and thus it is preserved in the solution to the recursive call; or it is dense enough in the input so that it is a member of each solution and we can detect this situation from scanning the diagonal of the input codeword. This described nondeterministic approach is then made deterministic by simply considering every possible choice of k slices at each of the K steps of the recursion. This does not increase substantially the complexity of the decoding procedure, since $O(((K \cdot \text{poly}(k \log n))^k)^K)$ for $K \approx 4^k$ is still only doubly-exponential in k.

4 Reducing 4-Clique to All-Pairs Min-Cut

In this section we prove Theorem 1.1 by showing new reductions from the 4-Clique problem to k-bounded All-Pairs Min-Cut with unit vertex capacities. These reductions yield conditional lower bounds that are much higher than previous ones, which are based on SETH, in addition to always producing DAGs. Throughout this section, we will often use the term nodes for vertices.

▶ **Definition 4.1** (The 4-Clique Problem). *Given a 4-partite graph G, where $V(G) = A \cup B \cup C \cup D$ with $|A| = |B| = |C| = |D| = n$, decide whether there are four nodes $a \in A$, $b \in B$, $c \in C$, $d \in D$ that form a clique.*

This problem is equivalent to the standard formulation of 4-Clique (without the restriction to 4-partite graphs). The currently known running times are $O(n^{\omega+1})$ using matrix multiplication [15], and $O(n^4/\operatorname{polylog} n)$ combinatorially [35]. The k-Clique Conjecture

[3] hypothesizes that current clique algorithms are optimal. Usually when the k-Clique Conjecture is used, it is enough to assume that the current algorithms are optimal for every k that is a multiple of 3, where the known running times are $O(n^{\omega k/3})$ [30] and $O(n^k/\operatorname{polylog} n)$ combinatorially [32], see e.g. [2, 3, 10, 12, 25]. However, we will need the stronger assumption that one cannot improve the current algorithms for $k = 4$ by any polynomial factor. This stronger form was previously used by Bringmann, Grønlund, and Larsen [9].

4.1 Reduction to the Unbounded Case

We start with a reduction to the unbounded case (equivalent to $k = n$), that is, we reduce to All-Pairs Min-Cut with unit node capacities (abbreviated APMVC, for All-Pairs Minimum Vertex-Cut). Later (in Section 4.1) we will enhance the construction in order to bound k.

▶ **Lemma 4.2.** *Suppose APMVC on n-node DAGs with unit node capacities can be solved in time $T(n)$. Then 4-Clique on n-node graphs can be solved in time $O(T(n) + MM(n))$, where $MM(n)$ is the time to multiply two matrices from $\{0,1\}^{n\times n}$.*

To illustrate the usage of this lemma, observe that an $O(n^{3.99})$-time combinatorial algorithm for APMVC would imply a combinatorial algorithm with similar running time for 4-Clique.

Proof. Given a 4-partite graph G as input for the 4-Clique problem, the graph H is constructed as follows. The node set of H is the same as G, and we abuse notation and refer also to $V(H)$ as if it is partitioned into A,B,C, and D. Thinking of A as the set of sources and D as the set of sinks, the proof will focus on the number of node-disjoint paths from nodes $a \in A$ to nodes $d \in D$. The edges of H are defined in a more special way, see also Figure 4 for illustration.

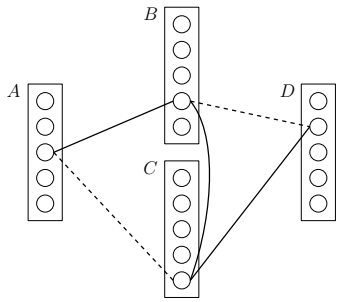


Figure 4 An illustration of H in the reduction. Solid lines between nodes represent the existence of an edge in the input graph G, and dashed lines represent the lack thereof.

- (A to B) For every $a \in A, b \in B$ such that $\{a, b\} \in E(G)$, add to $E(H)$ a directed edge (a, b).
- (B to C) For every $b \in B, c \in C$ such that $\{b, c\} \in E(G)$, add to $E(H)$ a directed edge (b, c).
- (C to D) For every $c \in C, d \in D$ such that $\{c, d\} \in E(G)$, add to $E(H)$ a directed edge (c, d).

The definition of the edges of H will continue shortly. So far, edges in H correspond to edges in G, and there is a (directed) path $a \to b \to c \to d$ if and only if the three (undirected) edges $\{a, b\}, \{b, c\}, \{c, d\}$ exist in G. In the rest of the construction, our goal is to make this 3-hop path contribute to the final $a \to d$ flow *if and only if* (a, b, c, d) is a 4-clique in G (i.e.,

all six edges exist, not only those three). Towards this end, additional edges are introduced, that make this 3-hop path useless in case $\{a, c\}$ or $\{b, d\}$ are not also edges in G. This allows "checking" for five of the six edges in the clique, rather than just three. The sixth edge is easy to "check".

- (A to C) For every $a \in A, c \in C$ such that $\{a, c\} \notin E(G)$, add to $E(H)$ a directed edge (a, c).
- (B to D) For every $b \in B, d \in D$ such that $\{b, d\} \notin E(G)$ in G, add to $E(H)$ a directed edge (b, d).

This completes the construction of H. Note that these additional edges imply that there is a path $a \to b \to d$ in H iff $\{a, b\} \in E(G)$ and $\{b, d\} \notin E(G)$, and similarly, there is a path $a \to c \to d$ in H iff $\{a, c\} \notin E(G)$ and $\{c, d\} \in E(G)$. Let us introduce notations to capture these paths. For nodes $a \in A, d \in D$ denote:

$$B'_{a,d} = \{b \in B \mid \{a, b\} \in E(G) \text{ and } \{b, d\} \notin E(G)\},$$
$$C'_{a,d} = \{c \in C \mid \{a, c\} \notin E(G) \text{ and } \{c, d\} \in E(G)\}.$$

We now argue that if an APMVC algorithm is run on H, enough information is received to be able to solve 4-Clique on G by spending only an additional post-processing stage of $O(n^3)$ time.

▷ Claim 4.3. Let $a \in A, d \in D$ be nodes with $\{a, d\} \in E(G)$. If the edge $\{a, d\}$ does not participate in a 4-clique in G, then the node connectivity from a to d in H, denoted $NC(a, d)$, is exactly

$$NC(a, d) = |B'_{a,d}| + |C'_{a,d}|,$$

and otherwise $NC(a, d)$ is strictly larger.

Proof of Claim 4.3. We start by observing that all paths from a to d in H have either two or three hops.

Assume now that there is a 4-clique (a, b^*, c^*, d) in G, and let us exhibit a set P of node-disjoint paths from a to d of size $|B'_{a,d}| + |C'_{a,d}| + 1$. For all nodes $b \in B'_{a,d}$, add to P the 2-hop path $a \to b \to d$. For all nodes $c \in C'_{a,d}$, add to P the 2-hop path $a \to c \to d$. So far, all these paths are clearly node-disjoint. Then, add the 3-hop path $a \to b^* \to c^* \to d$ to P. This path is node-disjoint from the rest because $b^* \notin B'_{a,d}$ (because $\{b^*, d\} \in E(G)$) and $c^* \notin C'_{a,d}$ (because $\{a, c^*\} \in E(G)$).

Next, assume that no nodes $b \in B, c \in C$ complete a 4-clique with a, d. Then for every set P of node-disjoint paths from a to d, there is a set P' of 2-hop node-disjoint paths from a to d that has the same size. To see this, let $a \to b \to c \to d$ be some 3-hop path in P. Since (a, b, c, d) is not a 4-clique in G and $\{a, d\}, \{a, b\}, \{b, c\}, \{c, d\}$ are edges in G, we conclude that either $\{a, c\} \notin E(G)$ or $\{b, d\} \notin E(G)$. If $\{a, c\} \notin E(G)$ then $a \to c$ is an edge in H and the 3-hop path can be replaced with the 2-hop path $a \to c \to d$ (by skipping b) and one is remained with a set of node-disjoint paths of the same size. Similarly, if $\{b, d\} \notin E(G)$ then $b \to d$ is an edge in H and the 3-hop path can be replaced with the 2-hop path $a \to b \to d$. This can be done for all 3-hop paths and result in P'. Finally, note that the number of 2-hop paths from a to d is exactly $|B'_{a,d}| + |C'_{a,d}|$, and this completes the proof of Claim 4.3. ◁

Computing the estimates. To complete the reduction, observe that the values $|B'_{a,d}|+|C'_{a,d}|$ can be computed for all pairs $a \in A, d \in D$ using two matrix multiplications. To compute the $|B'_{a,d}|$ values, multiply the two matrices M, M' which have entries from $\{0,1\}$, with $M_{a,b} = 1$ iff $\{a,b\} \in E(G) \cap A \times B$ and $M'_{b,d} = 1$ iff $\{b,d\} \notin E(G) \cap B \times D$. Observe that $|B'_{a,d}|$ is exactly $(M \cdot M')_{a,d}$. To compute $|C'_{a,d}|$, multiply M, M' over $\{0,1\}$ where $M_{a,c} = 1$ iff $\{a,c\} \notin E(G) \cap A \times C$ and $M'_{c,d} = 1$ iff $\{c,d\} \in E(G) \cap C \times D$.

After having these estimates and computing APMVC on H, it can be decided whether G contains a 4-clique in $O(n^2)$ time as follows. Go through all edges $\{a,d\} \in E(G) \cap A \times D$ and decide whether the edge participates in a 4-clique by comparing $|B'_{a,d}| + |C'_{a,d}|$ to the node connectivity $NC(a,d)$ in H. By the above claim, an edge $\{a,d\}$ with $NC(a,d) > |B'_{a,d}| + |C'_{a,d}|$ is found if and only if there is a 4-clique in G. The total running time is $O(T(n) + MM(n))$, which completes the proof of Lemma 4.2. ◀

4.2 Reduction to the *k*-Bounded Case

Next, we exploit a certain versatility of the reduction and adapt it to ask only about min-cut values (aka node connectivities) that are smaller than k. In other words, we will reduce to the k-bounded version of All-Pairs Min-Cut with unit node capacities (abbreviated kAPMVC, for k-bounded All-Pairs Minimum Vertex-Cut). Our lower bound improves on the $\Omega(n^\omega)$ conjectured lower bound for Transitive Closure as long as $k = \omega(n^{1/2})$.

▶ **Lemma 4.4.** *Suppose kAPMVC on n-node DAGs with unit node capacities can be solved in time $T(n,k)$. Then 4-Clique on n-node graphs can be solved in time $O(\frac{n^2}{k^2} \cdot T(n,k) + MM(n))$, where $MM(n)$ is the time to multiply two matrices from $\{0,1\}^{n \times n}$.*

Due to space constraints, the proof appears in the full version.

Proof of Theorem 1.1. Assume there is an algorithm that solves kAPMVC in time $O((n^{\omega-1}k^2)^{1-\varepsilon})$. Then by Lemma 4.4 there is an algorithm that solves 4-Clique in time $= O(\frac{n^2}{k^2} \cdot (n^{\omega-1}k^2)^{1-\varepsilon} + MM(n)) \leq O(n^{\omega+1-\varepsilon'})$, for some $\varepsilon' > 0$. The bound for combinatorial algorithms is achieved similarly. ◀

References

1. A. Abboud and V. V. Williams. Popular Conjectures Imply Strong Lower Bounds for Dynamic Problems. In *FOCS*, pages 434–443, October 2014. doi:10.1109/FOCS.2014.53.
2. Amir Abboud, Arturs Backurs, Karl Bringmann, and Marvin Künnemann. Fine-Grained Complexity of Analyzing Compressed Data: Quantifying Improvements over Decompress-and-Solve. In *FOCS*, pages 192–203, 2017. doi:10.1109/FOCS.2017.12.
3. Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the Current Clique Algorithms are Optimal, So is Valiant's Parser. In *FOCS*, pages 98–117, 2015. doi:10.1109/FOCS.2015.16.
4. Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. New Algorithms and Lower Bounds for All-Pairs Max-Flow in Undirected Graphs. *CoRR*, abs/1901.01412, 2019. arXiv:1901.01412.
5. Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching Triangles and Basing Hardness on an Extremely Popular Conjecture. *SIAM J. Comput.*, 47(3):1098–1122, 2018. doi:10.1137/15M1050987.
6. Srinivasa R Arikati, Shiva Chaudhuri, and Christos D Zaroliagis. All-Pairs Min-Cut in Sparse Networks. *Journal of Algorithms*, 29(1):82–110, 1998. doi:10.1006/jagm.1998.0961.
7. Attila Bernáth and Gyula Pap. Blocking unions of arborescences. *CoRR*, abs/1507.00868, 2015. arXiv:1507.00868.

8 Anand Bhalgat, Ramesh Hariharan, Telikepalli Kavitha, and Debmalya Panigrahi. An $\tilde{O}(mn)$ Gomory-Hu Tree Construction Algorithm for Unweighted Graphs. In *STOC*, pages 605–614, 2007. doi:10.1145/1250790.1250879.

9 Karl Bringmann, Allan Grønlund, and Kasper Green Larsen. A Dichotomy for Regular Expression Membership Testing. In *FOCS*, pages 307–318, 2017. doi:10.1109/FOCS.2017.36.

10 Karl Bringmann and Philip Wellnitz. Clique-Based Lower Bounds for Parsing Tree-Adjoining Grammars. In *CPM*, pages 12:1–12:14, 2017. doi:10.4230/LIPIcs.CPM.2017.12.

11 Timothy M. Chan. All-Pairs Shortest Paths with Real Weights in $O(n^3/\log n)$ Time. *Algorithmica*, 50(2):236–243, 2008. doi:10.1007/s00453-007-9062-1.

12 Yi-Jun Chang. Conditional Lower Bound for RNA Folding Problem. *CoRR*, abs/1511.04731, 2015. arXiv:1511.04731.

13 Ho Yee Cheung, Lap Chi Lau, and Kai Man Leung. Graph Connectivities, Network Coding, and Expander Graphs. *SIAM Journal on Computing*, 42(3):733–751, 2013. doi:10.1137/110844970.

14 D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990. doi:10.1016/S0747-7171(08)80013-2.

15 Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theor. Comput. Sci.*, 326(1-3):57–67, 2004. doi:10.1016/j.tcs.2004.05.009.

16 M. J. Fischer and A. R. Meyer. Boolean matrix multiplication and transitive closure. In *SWAT*, pages 129–131. IEEE, 1971. doi:10.1109/SWAT.1971.4.

17 Lester Randolph Ford, Jr. and Delbert Ray Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

18 Loukas Georgiadis, Daniel Graf, Giuseppe F. Italiano, Nikos Parotsidis, and Przemysław Uznański. All-Pairs 2-Reachability in $O(n^\omega \log n)$ Time. In *ICALP*, volume 80, pages 74:1–74:14, 2017. doi:10.4230/LIPIcs.ICALP.2017.74.

19 R. E. Gomory and T. C. Hu. Multi-Terminal Network Flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961. doi:10.1137/0109047.

20 Ramesh Hariharan, Telikepalli Kavitha, and Debmalya Panigrahi. Efficient Algorithms for Computing All Low *s*-*t* Edge Connectivities and Related Problems. In *SODA*, pages 127–136, 2007. URL: http://dl.acm.org/citation.cfm?id=1283383.1283398.

21 Robert Krauthgamer and Ohad Trabelsi. Conditional Lower Bounds for All-Pairs Max-Flow. *ACM Trans. Algorithms*, 14(4):42:1–42:15, 2018. doi:10.1145/3212510.

22 Jakub Łącki, Yahav Nussbaum, Piotr Sankowski, and Christian Wulff-Nilsen. Single Source – All Sinks Max Flows in Planar Digraphs. In *FOCS*, pages 599–608. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.66.

23 F. Le Gall. Powers of Tensors and Fast Matrix Multiplication. In *ISSAC*, pages 296–303, 2014. doi:10.1145/2608628.2608664.

24 Y. T. Lee and A. Sidford. Path Finding Methods for Linear Programming: Solving Linear Programs in $\widetilde{O}(\sqrt{rank})$ Iterations and Faster Algorithms for Maximum Flow. In *FOCS*, pages 424–433, 2014. doi:10.1109/FOCS.2014.52.

25 Andrea Lincoln, Virginia Vassilevska Williams, and R. Ryan Williams. Tight Hardness for Shortest Cycles and Paths in Sparse Graphs. In *SODA*, pages 1236–1252, 2018.

26 A. Mądry. Computing Maximum Flow with Augmenting Electrical Flows. In *FOCS*, pages 593–602, 2016. doi:10.1109/FOCS.2016.70.

27 Dániel Marx. Parameterized graph separation problems. *Theoretical Computer Science*, 351(3):394–406, 2006.

28 K. Menger. Zur allgemeinen Kurventheorie. *Fundamenta Mathematicae*, 10(1):96–115, 1927.

29 Hiroshi Nagamochi and Yoko Kamidoi. Minimum cost subpartitions in graphs. *Inf. Process. Lett.*, 102(2-3):79–84, 2007. doi:10.1016/j.ipl.2006.11.011.

30 Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.

31 Debmalya Panigrahi. Gomory-Hu trees. In Ming-Yang Kao, editor, *Encyclopedia of Algorithms*, pages 858–861. Springer, 2016. doi:10.1007/978-1-4939-2864-4.

32 Virginia Vassilevska. Efficient algorithms for clique problems. *Inf. Process. Lett.*, 109(4):254–257, 2009. doi:10.1016/j.ipl.2008.10.014.

33 V. Vassilevska Williams. Multiplying Matrices Faster Than Coppersmith-Winograd. In *STOC*, pages 887–898, 2012. doi:10.1145/2213977.2214056.

34 Virginia Vassilevska Williams and R. Ryan Williams. Subcubic Equivalences Between Path, Matrix, and Triangle Problems. *J. ACM*, 65(5):27:1–27:38, 2018. doi:10.1145/3186893.

35 Huacheng Yu. An improved combinatorial algorithm for Boolean matrix multiplication. *Inf. Comput.*, 261:240–247, 2018. doi:10.1016/j.ic.2018.02.006.

Fine-Grained Reductions and Quantum Speedups for Dynamic Programming

Amir Abboud
IBM Almaden Research Center, San Jose, California, USA
amir.abboud@ibm.com

Abstract

This paper points at a connection between certain (classical) fine-grained reductions and the question: Do quantum algorithms offer an advantage for problems whose (classical) best solution is via dynamic programming?

A remarkable recent result of Ambainis et al. [SODA 2019] indicates that the answer is positive for some fundamental problems such as Set-Cover and Travelling Salesman. They design a quantum $O^*(1.728^n)$ time algorithm whereas the dynamic programming $O^*(2^n)$ time algorithms are conjectured to be classically optimal. In this paper, fine-grained reductions are extracted from their algorithms giving the first lower bounds for problems in P that are based on the intriguing Set-Cover Conjecture (SeCoCo) of Cygan et al. [CCC 2010].

In particular, the SeCoCo implies:

- a super-linear $\Omega(n^{1.08})$ lower bound for 3-SUM on n integers,
- an $\Omega(n^{\frac{k}{c_k}-\varepsilon})$ lower bound for k-SUM on n integers and k-Clique on n-node graphs, for *any* integer $k \geq 3$, where $c_k \leq \log_2 k + 1.4427$.

While far from being tight, these lower bounds are significantly stronger than what is known to follow from the Strong Exponential Time Hypothesis (SETH); the well-known $n^{\Omega(k)}$ ETH-based lower bounds for k-Clique and k-SUM are vacuous when k is constant.

Going in the opposite direction, this paper observes that some "sequential" problems with previously known fine-grained reductions to a "parallelizable" core also enjoy quantum speedups over their classical dynamic programming solutions. Examples include RNA Folding and Least-Weight Subsequence.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Fine-Grained Complexity, Set-Cover, 3-SUM, k-Clique, k-SUM, Dynamic Programming, Quantum Algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.8

Category Track A: Algorithms, Complexity and Games

Funding We acknowledge the support of the Quantum Computing Sciences program of the U.S. Air Force, Office of Scientific Research, administered through Air Force Research Laboratory contract FA8750-18-C-0098.

Acknowledgements We thank Karl Bringmann and the anonymous reviewers for helpful feedback.

1 Introduction

An increasing amount of effort is being dedicated to the question: When and by how much can quantum algorithms beat classical ones? Perhaps the most successful approach for getting quantum speedups is using Grover's search [22], which offers a quadratic improvement over classical exhaustive search, which is the bottleneck in the best-known algorithms for a long list of problems. This list includes nearly all problems studied in fine-grained complexity such as SAT, Orthogonal Vectors, 3-SUM, All-Pairs Shortest Paths, and so on. In fact, none

of the popular conjectures in fine-grained complexity (e.g. SETH) remain plausible when Grover's quantum search is allowed (with one exception, to be discussed shortly). It gives magical capabilities such as the following theorem.

▶ **Theorem 1** (Quantum Minimum Finding [17]). *Let $a_1, \ldots, a_n$ be integers accessed by a procedure $\mathcal{P}$. There exists a quantum algorithm that finds $\min_{i=1}^{n}\{a_i\}$ with success probability at least $2/3$ using $O(\sqrt{n})$ applications of $\mathcal{P}$.*

But what about problems whose best known algorithm is via dynamic programming, not exhaustive search? Could those problems be resilient to quantum speedups? A famous open question in this context is whether we can solve sequence alignment problems such Edit Distance and Longest Common Subsequence in truly-subquadratic quantum time. Such an algorithm could lead to important progress in bioinformatics in the future. The main challenge is the sequential nature of dynamic programming, which prevents us from using a Grover-like approach; one first solves the problem on small instances and then combines them to solve larger and larger instances. None of the steps involve an expensive exhaustive search. Another example is the Set-Cover problem, which can be solved in $O^*(2^n)$ time[1] with classical dynamic programming [19], and is conjectured by Cygan et al. [15] to have an $\Omega((2-\delta)^n)$ lower bound, for all $\delta > 0$. This is the so-called Set-Cover Conjecture (SeCoCo). Would this popular conjecture remain plausible in a quantum world?

A recent breakthrough of Ambainis et al. [5] shows otherwise. The authors give quantum $O^*(1.728^n)$ time algorithms for Set-Cover and Traveling Salesman, as well as other exponential speedups for problems such as Graph Bandwidth and Feedback Arc Set. These are problems where the best known classical algorithm is via dynamic programming. (It is unclear if their techniques will lead to solving Edit Distance in truly subquadratic quantum time.)

Ambainis et al. find a way to use the quantum minimum finding theorem above to solve Set-Cover. This can be viewed as a *reduction* from Set-Cover to a "parallelizable" core, and for the final algorithm to be fast the reduction should be efficient, qualifying it as a *fine-grained reduction* from Set-Cover to some minimum-finding problem. The main observation here is that the parallelizable-nature of the latter problem allows for further reductions to the popular problems in fine-grained complexity such as 3-SUM and k-Clique. Previously, no interesting reductions from Set-Cover to natural problems in P were known, and this gives a partial resolution to an open question in fine-grained complexity (see Section 5.1 in [39] and the discussion at the end of this paper). The sequential nature of Set-Cover had been a barrier for reductions as well, and the ideas of Ambainis et al. overcome it. The next subsection states the results and elaborates on their importance to the landscape of fine-grained complexity.

Section 3 suggests that this connection between quantum speedups for sequential problems and fine-grained reductions could be interesting also in the other direction.

1.1 The Consequences to Fine-Grained Complexity

The goal of fine-grained complexity and algorithms is to achieve upper and lower bounds that are as tight as possible for the computational problems of interest. The lower bounds are obtained via reductions and are based on a small set of popular conjectures, such as the Strong Exponential Time Hypothesis (SETH)[2] [23, 24], regarding the hardness of certain

[1] The notation $O^*(\cdot)$ hides polynomial factors.

[2] ETH states that 3-SAT cannot be solved in $2^{o(n)}$ time. The stronger version, SETH, states that we cannot solve k-SAT in $O((2-\varepsilon)^n)$ time for all k.

core problems, e.g. CNF-SAT. By now, there is a very long and evergrowing list of lower bounds that are tight up to $n^{o(1)}$ factors. See the recent surveys [40, 37]. One of the main gaps in knowledge, when it comes to this framework, is regarding the connections between the conjectures. For example, the k-SUM and k-Clique conjectures were used to prove many lower bound results that SETH (or any other conjectures) seem incapable of proving. Many conjectures have their own "hardness-class" and the whole theory would be much better if these conjectures can be supported by SETH or other (but different) conjectures.

k-SUM and k-Clique

The k-SUM problem asks to find k among n given numbers that sum to zero. It can be solved in $O(n^{\lceil k/2 \rceil})$ time which is conjectured to be optimal up to $n^{o(1)}$ factors [3]. The $k = 3$ case is the 3-SUM conjecture which is an old and famous conjecture in computational geometry [20]. This conjecture, for an even k, is supported by the hypothesis that Subset-Sum cannot be solved in $O(2^{(1/2-\varepsilon)n})$ for some $\varepsilon > 0$. A reduction of Patrascu and Williams [33] gives an $n^{\Omega(k)}$ lower bound for k-SUM assuming the ETH, when k is super-constant.

The k-Clique problem asks to find k among n nodes that form a clique. It can be solved in $O(n^{\frac{\omega}{3} \cdot k})$ time[3] [32, 18], where ω is the matrix multiplication exponent [38, 21], and this is conjectured to be optimal (and that $\omega = 2 + o(1)$) [1]. This conjecture, for k divisible by 3, is supported by the hypothesis that Max-Cut cannot be solved exponentially faster than its current runtime, since the current-best algorithm by Williams reduces Max-Cut to k-Clique [36]. The k-SUM conjecture implies an $n^{\lceil k/2 \rceil - o(1)}$ lower bound for k-Clique [4], and the ETH implies an $n^{\Omega(k)}$ lower bound for super-constant k [12, 13].

The aforementioned ETH lower bounds do not imply anything for these two problems when k is constant. No nontrivial SETH-based lower bounds are known, for any constant k.

SeCoCo vs. SETH

In a seminal paper on fine-grained reductions among NP-complete problems, Cygan et al. [15] introduced SeCoCo and used it to prove tight lower bounds for problems with dynamic programming solutions. It states that Set-Cover on n elements cannot be solved in $(2 - \delta)^n$ time, even when the sets have constant size (more formally defined in Section 2). Since then SeCoCo has been utilized for lower bounds for other NP-hard problems [6, 9, 26, 28].

Currently, SETH and SeCoCo seem incomparable and each of them has some advantages over the other as a hardness assumption. However, it is likely that SeCoCo will turn out to be a strictly safer (weaker, more believable) conjecture; there is no known barrier for (tightly) reducing SAT to Set-Cover and, at least back in 2010, Cygan et al. conjectured that such a reduction exists. On the other hand, a (tight) reduction in the other way, from Set-Cover to SAT, would be much more surprising. Unless the reduction is unusual, it would allow one to use the trivial 2^n algorithm for SAT to solve Set-Cover in 2^n time in a way that is simpler (no dynamic programming) and better in some ways (e.g. parallelizable). Even with today's knowledge, the latter point demonstrates an advantage of SeCoCo as a basis for reductions: it gives barriers for (e.g.) solving the end problem without dynamic programming. Meanwhile, SETH enjoys other benefits: First, it is much more popular. Second, SAT has been more extensively studied than Set-Cover in many areas such as complexity theory and verification. And third, refuting it implies some (weak) circuit lower bounds; this does not make it more believable, but it does "raise the stakes" in a way that is not known for SeCoCo. The reader is referred to [15, 31, 2, 27] for other discussions on the matter.

[3] The bound is slightly more complicated bound when k is not divisible by 3.

New Results

The main result of this paper is an extraction of fine-grained reductions from the quantum algorithms of Ambainis et al. leading to the first nontrivial lower bounds in P that are based on the Set-Cover Conjecture. The lower bounds can be based on a weaker form of the conjecture (discussed in Section 2) where sets are allowed to have size up to $n^{o(1)}$. The full version of the paper shows the same lower bound for the k-Orthogonal Vectors problem, via a minor modification to the reductions.

▶ **Theorem 2.** *Let $k \geq 3$ and[4] $c_k = k \cdot H(k^{-1}) = \log_2(k-1) + k \cdot \log \frac{k}{k-1}$. If for some $\varepsilon > 0$, either*

- *the k-Sum problem on n integers, or*
- *the k-Clique problem on n node graphs and $n^{2-\varepsilon}$ edges,*

can be solved in $O(n^{\frac{k}{c_k}-\varepsilon})$ time, then the (weak) SeCoCo is false.

Recall that the conjectured lower bounds are $1/c_k = 1/2$ for k-SUM and $1/c_k = 2/3$ for k-Clique. Whereas here, as k grows, the function c_k approaches $\log_2 k + 1.4427$, which means that the lower bound is approximately $\Omega(n^{\frac{k}{\log k}})$. It is non-trivial for k-Clique for all $k \geq 9$, while for k-SUM (where the input size is smaller) it is meaningful already in the most famous case of $k = 3$.

▶ **Corollary 3.** *If 3-SUM on n integers can be solved in $O(n^{1/H(1/3)-\varepsilon}) = O(n^{1.089-\varepsilon})$ time, for some $\varepsilon > 0$, then the (weak) SeCoCo is false.*

Perhaps the most exciting aspect of these results is that one can finally have a concrete lower bound for k-Clique and k-SUM for constant k. For instance, it implies an n^{10} lower bound for 2^{11}-Clique and 2^{11}-Sum. These two problems are canonical in parameterized complexity as they can be reduced to many other problems while preserving the boundedness of the natural parameter. Thus, the SeCoCo conjecture implies concrete lower bounds for all those other problems as well, when the parameter is a fixed constant. It is still an important open question to prove such lower bounds with a fixed $c_k > 0$ that is independent of k.

Replacing SeCoCo with SETH is a big open question. It is likely to be possible; in particular, it would follow if a reduction from SAT to Set-Cover is found. Such a reduction was conjectured to exist by Cygan et al. [15]. Notably, this suggests a barrier for proving much higher lower bounds for 3-SUM. It is known that the nondeterministic version of SETH (NSETH) rules out a SETH lower bound for 3-SUM of $\Omega(n^{1.5+\varepsilon})$ [11], and by the conjecture of Cygan et al. this is also a barrier for SeCoCo-based lower bounds.

Further open questions are discussed at the end of the paper. Since the algorithms of Ambainis et al. [5] for Set-Cover and for TSP are very similar, the exact same lower bounds above can be based on a corresponding assumption about the hardness of TSP. The modified proofs are deferred to the full version of the paper.

Less-Fine-Grained Complexity: Linear vs. Super-Linear

Traditional complexity theory classifies problems into polynomial (efficient) vs. super-polynomial (inefficient). Due, in part, to the increase in the data sizes this classification is now viewed as often being too coarse. Fine-Grained complexity's goal is to get a more exact classification that will be more relevant in many scenarios. It is natural to view the

[4] Let $H(p)$ be the binary entropy function on $p \in [0,1]$. See Section 2.

first-order goal of such a theory as: classifying problems into near-linear time solvable ones, those that can be solved "efficiently" even in "Big-Data" settings, vs. the ones that require super-linear time to solve. The latter set of problems will have to be relaxed before they can be solved at large scales. The more valuable and pleasing exact classification can be viewed as the next-step after this *less-fine-grained* classification is achieved.

This less-fine-grained classification has been conditionally achieved for a long list of problems, and the results of this paper strengthen the foundations for some of these results. For example, many problems in computational geometry are 3-SUM-hard in the sense that there is a linear-time reduction from 3-SUM to them. If these problems can be solved in near-linear time, then 3-SUM can, which is conjectured to be impossible (by a much weaker version of the 3-SUM conjecture). But what are other justifications for these "lower bounds"? Corollary3 gives a new one: such near-linear time algorithms would refute SeCoCo.

1.2 Other Related Work

Another connection between fine-grained complexity and quantum computing was recently demonstrated by Chen and Wang [14]. The authors show that a fast and communication-efficient *quantum* protocol for a function f implies a fast classical approximate counting algorithm for a related pair counting problem. They instantiate this connection in order to show a new approximate counting algorithm for the #-Orthogonal-Vectors problem.

A very recent paper by Khadiev [25] suggests a new quantum dynamic programming approach for problems on DAGs. It will be interesting to see if fine-grained reductions can be extracted.

2 Fine-Grained Reductions

Preliminaries

The standard notation $[n] = \{1, \ldots, n\}$ will be used throughout the paper. The starting point of the reductions in this paper is Set-Cover.

▶ **Definition 4** (The Set-Cover Problem). *Given m sets over the universe $U = [n]$, return the minimum number of sets required to cover U.*

The Set-Cover Conjecture (SeCoCo) of Cygan et al. states that the problem requires $2^{n-o(n)}$ time, even when all sets have constant size. Krauthgamer and Trabelsi [27] demonstrate that the Log-Set-Cover Conjecture, where the sets can have size up to $O(\log n)$, is equally useful but potentially more believable. They show that refuting it implies a breakthrough algorithm for Directed Hamiltonicity and Directed n-Tree. For the lower bounds in this paper, an even more relaxed conjecture is sufficient[5], where the sets are allowed to be of any size $n^{o(1)}$. Note that this automatically bounds the number of sets by $m = 2^{o(n)}$ which will be negligible.

The Weak Set-Cover Conjecture. No algorithm can solve Set-Cover over the universe $[n]$ with sets of size $n^{o(1)}$ in time $O((2-\delta)^n)$ where $\delta > 0$.

The main idea in the reductions, which is the crux of the quantum algorithms in [5], is an unusual split-and-list where all subsets of $[n]$ of size up to $d = n/k$ are enumerated. The analysis will rely on the following approximation of binomial coefficients, bounding the number of such sets:

[5] Although the stronger assumption would allow to extend the lower bound for larger values of k beyond $n^{1-\Omega(1)}$.

▶ **Lemma 5** (Entropy Approximation). *For all* $1 \leq d \leq n/2$,

$$\binom{n}{\leq d} = \sum_{i=1}^{d} \binom{n}{d} \leq 2^{H(d/n)\cdot n},$$

where $H(\epsilon) = -(\epsilon \log_2 \epsilon + (1-\epsilon)\log_2(1-\epsilon))$ *is the binary entropy of* $0 \leq \epsilon \leq 1$.

Observe that $H(1/k) = \frac{\log_2 k + O(1)}{k}$. In the proofs, the following bounds will be used for any $1 \leq k \leq n^{\varepsilon}$ where $\varepsilon < 1$: $\binom{n}{\leq n/k} \leq 2^{H(1/k)\cdot n}$ and when $t = n^{o(1)}$ then $\binom{n}{\leq n/k+t} \leq 2^{(H(1/k)+o(1))\cdot n}$. The last inequality follows because $\binom{n}{\leq n/k+t} \leq n^{o(1)} \cdot \binom{n}{n/k+t} \leq n^{o(1)} \cdot \binom{n}{(1+o(1))\cdot n/k} \leq 2^{H((1+o(1))/k)\cdot n + o(n)} \leq 2^{(1+o(1))\cdot H(1/k)\cdot n + o(n)} \leq 2^{(H(1/k)+o(1))\cdot n}$.

Key Observation

Following the notation in Ambainis et al., for a set system $\mathcal{S}$ over a universe U we denote the size of the minimum set-cover by $f(U, \mathcal{S})$. By definition, it follows that for any k sets $U_1, \ldots, U_k$ with union equal to U and any $\mathcal{S}$:

$$\sum_{i=1} f(U_i, \mathcal{S}) \geq f(U, \mathcal{S})$$

The central claim for all the reductions in this paper (and the quantum algorithm of Ambainis et al.) states that the above will be an equality in some cases.

▷ Claim 6. If all sets in a set system $\mathcal{S}$ over a universe U have size at most t then for all integers $k \geq 2$ such that $t < n/k$ there exist k disjoint subsets $U_1, \ldots, U_k \subseteq U$ each of size between $n/k - t$ and $n/k + t$ such that their union is equal to U and:

$$\sum_{i=1}^{k} f(U_i, \mathcal{S}) \leq f(U, \mathcal{S}).$$

Proof. Let $\mathcal{C} \subseteq \mathcal{S}$ be a minimum set-cover of U of size $\ell = f(U, \mathcal{S})$. To define a partition of U we consider this process: Start from an empty set X and add to it elements from U in the following way. Pick an arbitrary ordering of the sets in $\mathcal{C} = \{S_1, \ldots, S_\ell\}$ and go through them in order, at the i^{th} step we add all elements of S_i to X, unless they were already present.

Two observations about this process: (i) Since $\mathcal{C}$ is a Set-Cover we will end up with $X = U$. And (ii) all sets in $\mathcal{S}$ have size at most t and therefore each step can add up to t elements to X.

Define the sets $U_1, \ldots, U_k$ as follows. U_1 contains all elements added to X up until the earliest step in which $|X|$ became n/k or greater. Due to observation (ii) this guarantees that U_1 contains at most $n/k + t - 1$ elements. U_2 contains "the next" about n/k elements, and so on. More formally, U_i contains all elements added to X after the step in which its size became $\geq i \cdot n/k$ and up until the step in which its size became $\geq (i+1) \cdot n/k$. Again, due to observation (ii) the size of each U_i is upper bounded by $n/k + t$ and lower bounded by $n/k - t$. The sets are disjoint by definition and their union is equal to U.

Finally, we prove the inequality by arguing that each U_i can be covered separately using only the sets from $\mathcal{C}$. This is because one can choose exactly the sets that are responsible for adding the elements to U_i in the process in order to cover U_i. This is a cover by construction and thus $f(U_i, \mathcal{S})$ is upper bounded by this number of sets. And since the sets responsible for each U_i are different, the total number we choose for all the U_i's is exactly $|\mathcal{C}|$. Thus, $\sum_{i=1}^{k} f(U_i, \mathcal{S}) \leq |\mathcal{C}| = f(U, \mathcal{S})$. ◁

Intermediate Problems

This now allows for two preliminary reductions from Set-Cover to a parameterizd version of Exact Cover and to a simpler version of the k-Orthogonal-Vectors problem.

▶ **Definition 7** (The k-Exact-Cover Problem). *Given k lists of sets, each containing N subsets of the same universe U, decide if there are k subsets, one from each list, that are an exact cover of U, that is, they are disjoint and their union is equal to U.*

The following reduction from Set-Cover to this problem is a combination of existing tricks with the core idea of the algorithm by Ambainis et al. which is the enumeration of all subsets of $[n]$ of size equal to n/k up to plus-or-minus the size of a set in the instance, and the preprocessing of their solutions. The correctness will follow from the above key claim.

▶ **Lemma 8.** *For all $2 \leq k \leq n^{\varepsilon}$ where $\varepsilon < 1$, the Set-Cover problem on n elements and m sets of size $n^{o(1)}$ can be reduced to $2^{o(n)}$ instances of the k-Exact-Cover problem on k lists of $N \leq 2^{H(1/k)\cdot n+o(n)}$ subsets over a universe U of size n. The reduction runs in time $2^{H(1/k)\cdot n+o(n)}$.*

Proof. First, reduce Set-Cover to the decision version: is there a set-cover of size exactly ℓ? This can be done simply with an overhead of n in the number of instances by trying all possible values for $1 \leq \ell \leq n$, and returning the smallest one with a positive answer. Thus, it is enough to only consider the decision version.

The goal is to look for a partition of U into k sets that can be covered separately at no extra cost, as promised by Claim 6. Let $U_1, \ldots, U_k$ be such sets, and let $\alpha' = (f(U_1, \mathcal{S}), \ldots, f(U_k, \mathcal{S}))$ be the k-tuple of the sizes of their minimum covers. Note that for all i: $f(U_i, \mathcal{S}) \leq |U_i| \leq n/k + t$ where $t = n^{o(1)}$ is an upper bound on the size of the given sets. As a second preliminary step, we enumerate all tuples α of k numbers in $[n/k + t]$ that sum to ℓ, in an attempt to guess α', and for each tuple there will be an instance of k-Exact-Cover. The total number of tuples is at most $(n/k + t)^k = 2^{o(n)}$. From now on fix a tuple $\alpha = (\alpha_1, \ldots, \alpha_k)$.

Enumerate all subsets of U that are of size up to $n/k + t$, call this collection $\mathcal{P}$. In a preprocessing step, compute the minimum set-cover $f(P, \mathcal{S})$ for all sets $P \in \mathcal{P}$. This can be done in $O(|\mathcal{P}| \cdot m)$ time: In the classical dynamic programming algorithm we compute $f(X, \mathcal{S})$ for increasingly larger sets via the formula $f(X, \mathcal{S}) = \min_{S \in \mathcal{S}}\{f(X \setminus S, \mathcal{S}) + 1\}$. Do the same, but do not process sets such that $|X| > n/k + t$. Thus, the total running time for this step, which is the most expensive in the reduction, is $O(\binom{n}{\leq n/k+t} \cdot m) \leq 2^{H(1/k)\cdot n+o(n)}$.

For all $i \in [k]$, the i^{th} list $\mathcal{L}_i$ in the k-Exact-Cover instance contains all sets $P \in \mathcal{P}$ whose minimum set-cover size matches the guess in α, that is,

$$\mathcal{L}_i = \{P \in \mathcal{P} \mid f(P, \mathcal{S}) = \alpha_i\}.$$

This completes the reduction: at least one of the k-Exact-Cover instances (for some tuple α) is a yes-instance, if and only if there is a set-cover of size ℓ.

For the correctness, observe that for any α and any k sets $X_1 \in \mathcal{L}_1, \ldots, X_k \in \mathcal{L}_k$ whose union is equal to U we have that $f(U, \mathcal{S}) \leq \sum_{i=1}^{k} f(X_i, \mathcal{S}) = \ell$. Thus, if one of the k-Exact-Cover instances is a yes, then $f(U, \mathcal{S}) \leq \ell$. For the other direction, assume that there is a set-cover of size ℓ and consider the partition $U_1, \ldots, U_k$ promised by Claim 6 and let $\alpha = \alpha'$ above be the tuple of minimum set-cover sizes for the k sets in that partition. The instance corresponding to this α is a yes-instance because each U_i will exist in $\mathcal{L}_i$ and their union is equal to U and, moreover, they are disjoint. ◀

The second intermediate problem superficially looks like the k-Orthogonal-Vectors problem (which is the canonical problem for SETH-based lower bounds) but it is in fact easier and is more closely related to k-Clique.

▶ **Definition 9** (The k-Pairwise-Disjoint-Sets Problem, or k-Pairwise-Orthogonal-Vectors)**.** *Given k lists of sets, each containing N subsets of the same universe U, decide if there are k subsets, one from each list, that are pairwise disjoint.*

The reduction is also by a combination of existing tricks with the key claim.

▶ **Lemma 10.** *For all $2 \leq k \leq n^{\varepsilon}$ where $\varepsilon < 1$, the Set-Cover problem on n elements and m sets of size $n^{o(1)}$ can be reduced to $2^{o(n)}$ instances of the k-Pairwise-Disjoint-Sets problem on k lists of $N \leq 2^{H(1/k)\cdot n+o(n)}$ subsets over a universe U of size n. The reduction runs in time $2^{(1/k+H(1/k))\cdot n+o(n)}$.*

Moreover, in all instances, each subset can be disjoint to at most $2^{(1-1/k)\cdot H(\frac{1}{k-1})\cdot n+o(n)}$ others.

The last sentence of the lemma will be helpful in reducing the sparsity of the k-Clique instances later on.

Proof. The proof is similar to the one of Lemma 8 with a bit more. The additional challenge here is that the end problem does not check that the union of the sets is equal to U, only that they are disjoint. The trick to handle this is yet another guessing step making sure that the sizes of the sets we are choosing sums up to n (before, we only checked that the sum of their *min covers* is equal to ℓ).

First, as before, we reduce to the decision version, and also enumerate k-tuples $\alpha \in [n/k + t]^k$ with sum ℓ, that try to guess the values $f(U_i, \mathcal{S})$ in the partition $U_1, \ldots, U_k$ promised by Claim 6. The additional step here is to also enumerate all k-tuples $\beta \in [n/k+t]^k$ with sum equal to $|U| = n$, that try to guess the values $|U_i|$. The number of β's is also upper bounded by $2^{o(n)}$. From now on fix both an $\alpha = (\alpha_1, \ldots, \alpha_k)$ and a $\beta = (\beta_1, \ldots, \beta_k)$.

Then, as before, we enumerate the collection $\mathcal{P}$ of all subsets of U of size up to $n/k + t$, and compute $f(P, \mathcal{S})$ for each $P \in \mathcal{P}$. These computations can be done with dynamic programming in a total of $2^{H(1/k)\cdot n+o(n)}$ time.

For all $i \in [k]$, the i^{th} list $\mathcal{L}_i$ in the k-Pairwise-Disjoint-Sets instance contains all sets $P \in \mathcal{P}$ whose minimum set-cover size matches the guess in α *and* their size matches the guess in β, that is,

$$\mathcal{L}_i = \{P \in \mathcal{P} \mid f(P, \mathcal{S}) = \alpha_i \text{ and } |U_i| = \beta_i\}.$$

This completes the reduction: at least one of the k-Pairwise-Disjoint-Sets instances (for some tuples α, β) is a yes-instance, if and only if there is a Set-Cover of size ℓ.

For the correctness, observe that for any α, β and any k sets $X_1 \in \mathcal{L}_1, \ldots, X_k \in \mathcal{L}_k$ that are disjoint, we have that $f(U, \mathcal{S}) \leq \sum_{i=1}^{k} f(X_i, \mathcal{S}) = \ell$ and $|X_1 \cup \cdots \cup X_k| = \sum_{i=1}^{k} |X_i| = \sum_i \beta_i = n$. The latter implies that the sets cover the entire U. Thus, if one of the k-Exact-Cover instances is a yes, then $f(U, \mathcal{S}) \leq \ell$. For the other direction, assume that there is a set-cover of size ℓ and consider the partition $U_1, \ldots, U_k$ promised by Claim 6 and let α be the tuple of minimum set-cover sizes for the k sets in that partition, and β be the tuple of their sizes. The instance corresponding to this α, β pair is a yes-instance because each U_i will exist in $\mathcal{L}_i$ and they are disjoint.

Finally, observe that each subset in our instances has size at least n/k, and therefore it can be disjoint to at most

$$\binom{n - n/k}{n/k + t} = 2^{(1-1/k)\cdot H(\frac{1}{k-1})\cdot n+o(n)}$$

other sets in every instance. ◀

2.1 The Reduction to k-SUM

The following version of k-SUM is convenient for reductions.

▶ **Definition 11** (The k-SUM Problem). *Given k lists of N numbers in $[M]$ and a target number $t \in [M]$, decide if there are k numbers, one from each list, that sum to t.*

This "list" or "colored" version is equivalent to the more natural version (where we only have one list and/or where the target is fixed to $t = 0$) up to a k factor in the number of numbers and a k factor in their size.

The reduction to k-SUM is by a simple encoding of the k-Exact-Cover problem.

▶ **Theorem 12.** *For all $2 \leq k \leq n^{\varepsilon}$ where $\varepsilon < 1$, the Set-Cover problem on n elements and m sets of size $n^{o(1)}$ can be reduced to $2^{o(n)}$ instances of the k-SUM problem on k lists of $N = 2^{H(1/k)\cdot n+o(n)}$ integers and a target, where the numbers are $n \cdot \lceil \log_2 k \rceil = O(k \log N)$ bits long. The reduction runs in time $2^{H(1/k)\cdot n+o(n)}$.*

Proof. First, reduce Set-Cover instance to $2^{o(n)}$ instance of k-Exact-Cover, as in Lemma 8, and then reduce each instance to an equivalent k-SUM instance. The k-Exact-Cover instances have k lists of $N \leq 2^{H(1/k)\cdot n+o(n)}$ subsets over the universe $U = [n]$.

The following is a natural mapping g from sets $X \subseteq [n]$ to $(n \cdot \log_2 k)$ bit integers: for all $i \in [n]$, the bit number $(i-1) \cdot \lceil \log_2 k \rceil + 1$ in $g(X)$ is set to 1 if $i \in X$ and to 0 otherwise. All other bits are set to 0, and these are just "buffers" of length $\log_2 k$ that prevent the sum of k numbers to carry over from one "interesting" location to another. Another way to think of the mapping is as writing a number in base k where the i^{th} digit is 1 if $i \in X$ and 0 otherwise.

Let $\mathcal{L}_i$ be the i-th list in the k-Exact-Cover, and we will map it into a list of integers L_i by encoding each set $X \in \mathcal{L}_i$ with the integer $g(X)$. The target sum t for the k-SUM instance is the number $g(U)$, or in other words, the number that is all 1 in base k. This completes the reduction.

For the correctness, one can check that for any k sets $X_1, \ldots, X_k$: $\sum_{i=1}^{k} g(X_i) = g(U)$ if and only if the X_i's are disjoint and $X_1 \cup \cdots \cup X_k = U$. ◀

This proves the k-SUM part of the main theorem, and the lower bound for 3-SUM.

2.2 The Reduction to k-Clique

A reduction from k-SUM on n numbers to $n^{o(1)}$ instances of k-Clique on n nodes is known [4], and it can be used directly to get the desired result. However, the instances generated by that reduction could be dense ($m = n^2$ edges), whereas the following direct reduction from Set-Cover gives the same result but on sparser graphs.

The following version of k-Clique is most convenient for reductions and is equivalent up to a factor k.

▶ **Definition 13** (The k-Clique Problem). *Given a k-partite graph with N nodes in each part and M edges, decide if there is a k-clique (with one node in each part).*

The reduction to k-Clique is almost immediate after one goes though the k-Pairwise-Disjoint-Sets problem.

▶ **Theorem 14.** *For all $2 \leq k \leq n^{\varepsilon}$ where $\varepsilon < 1$, the Set-Cover problem on n elements and m sets of size $n^{o(1)}$ can be reduced to $2^{o(n)}$ instances of the k-Clique problem on k partite graphs of $N = 2^{H(1/k)\cdot n+o(n)}$ nodes and $M = 2^{(H(1/k)+(1-1/k)\cdot H(\frac{1}{k-1}))\cdot n+o(n)}$ edges. The reduction runs in time $2^{H(1/k)\cdot n+o(n)}$.*

Proof. First, reduce Set-Cover instance to $2^{o(n)}$ instance of k-Pairwise-Disjoint-Sets, as in Lemma 10, and then reduce each instance to an equivalent k-Clique instance. The nodes correspond to the sets and two nodes are connected by an edge iff the corresponding sets are disjoint. A k-clique corresponds directly to k pairwise-disjoint sets, and the correctness of the reduction is immediate. The parameters in the statement follow from the parameters in Lemma 10, including the bound on the number of edges which is:

$$M = N \cdot 2^{(1-1/k)\cdot H(\frac{1}{k-1})\cdot n + o(n)}.$$ ◀

This proves the k-Clique part of the main theorem. A corollary of this reduction is an $\Omega(M^{1.01-\varepsilon})$ lower bound for 9-Clique on M edge graphs under the Set-Cover Conjecture.

3 Quantum Algorithms

This section goes in the other direction and discusses a few examples where previously known fine-grained reductions from a "sequential problem" to a "parallelizable core" can be used to obtain quantum speedups. While these arguments are not the simplest way to obtain such quantum speedup, they may still be interesting conceptually.

The first example is from the work of Künnemann et al. [29] who investigated the fine-grained complexity of *one-dimensional dynamic programming*. Their results center around instantiations of a generic Least-Weight Subsequence (LWS) problem where: Given an ordered sequence of n data items, with a succinctly represented function that assigns a weight to each pair, find a (non-contiguous) subsequence of the data points that minimizes the total weight of pairs adjacent in the subsequence. The problem can be instantiated by fixing a weight function, and it can model basic questions such as the coin change problem and finding the longest chain of nested boxes. The authors prove subquadratic-equivalences between these problems and a parallelizable core such as the $(\min, +)$-Convolution problem or vector domination. Thus, classically these problems are unlikely to be solvable in subquadratic time. However, since the parallelizable core problems are easy to solve in $O(n^{2-\varepsilon})$ time for some $\varepsilon > 0$ (and even linear time) for Grover's search, their reductions lead to $O(n^{2-\varepsilon})$ quantum time algorithms for coin change, finding the longest chain of nested boxes, and many other problems as well.

Another example is the RNA Folding problem from bioinformatics and the related problems of Language Edit Distance and Stochastic Context-Free Grammar Parsing. These problems can be solved in cubic time with dynamic programming, and a reduction *a la* Valiant's Parser [35] shows that each of these problems can be solved in the same time as the $(\min, +)$-matrix multiplication problem, or distance product, which is equivalent to All Pairs Shortest Paths and many other problems [41]. In fact, this reduction was recently used by Bringmann et al. [10] to improve the upper bound to $O(n^{2.8244})$ by utilizing a special structure in the $(\min, +)$-matrix multiplication instances produced by the reduction. This is the fastest classical algorithm to date for each of these problems. Can it be improved with a quantum algorithm? Yes, even without any special structure, the $(\min, +)$-matrix multiplication problem is parallelizable and can be solved in $O(n^{2.5})$ time, and via Valiant's reduction, so can these problems. A detailed exposition of these reductions can be found in [10], and related examples on how to incorporate such quantum matrix multiplication algorithm to solve combinatorial problems can be found in [30].

4 Open Questions

The results in this paper lead to the following thoughts.

- From a technical perspective, the reductions in this paper and the quantum algorithms of Ambainis et al. are via an unusual split-and-list approach: it is less efficient, increasing the search space size from 2^n to $2^{\frac{k}{H(1/k)}\cdot n}$, but it can simplify the structure of the space. Is this approach fruitful in other settings, not only when the problems are hard to parallelize? Can it be applied to SAT to get SETH-based lower bounds, that may not be tight, but could have a whole new flavor?
- The efficiency of the reductions does not exactly match that of the quantum algorithm for Set-Cover, since the algorithm benefits from more asymmetry in the splitting[6]. However, it is plausible that better quantum algorithms will lead to better reductions (and lower bounds), and vice versa. Intuitively, the best result one can hope for, without refuting conjectures, is to solve Set-Cover in $\sqrt{2^n}$ quantum time and extract an $n^{k/2}$ (classical) lower bound for k-SUM (which would be tight, for an even k) and for k-Clique (which would be tight if proven for sparse enough graphs). Perhaps this can be achieved by incorporating some of the ideas from the more technically involved works on Set-Cover, e.g. [8, 31, 7, 34], into the currently simple reduction/algorithm. This would be an exciting finding.
- Proving an $\Omega(n^{\varepsilon k})$ lower bound for k-Clique for a fixed $\varepsilon > 0$ (ideally $\varepsilon = 2/3$) under SETH (or even under SeCoCo) is still an important open question. The results here give a qualitatively similar results for k that is a small constant, but one would hope for more. This issue is often considered related to the frequently raised question: Does SETH (or SeCoCo) imply an $(1+\varepsilon)^n$ lower bound for 3-SAT or 100-SAT for any fixed $\varepsilon > 0$?
- The open question of whether SeCoCo implies lower bounds for problems in P was only answered in a limited sense here. Yes, interesting lower bounds for k-Clique and k-SUM can be derived, but the real intent behind the question is: Can SeCoCo give interesting lower bounds in P that other "established" or popular conjectures are incapable of proving? The results here do not qualify, but perhaps they will lead to such results. Can the structure of Set-Cover be utilized further in order to prove similar lower bounds for easier problems? A natural candidate is the $(\min,+)$-Convolution problem which is considered to be the easiest problem without a truly subquadratic time algorithm [16].
- In [5], the quantum speedup over classical dynamic programming for the Graph Bandwidth problem is more substantial than it is for Set-Cover and TSP. The best classical algorithm has $O^*(5^n)$ complexity while the quantum one runs in time $O^*(2.945^n)$, which is quite close to $\sqrt{5^n}$. This suggests that if one reduces the Graph Bandwidth problem to problems in P, e.g. 3-SUM, a tighter relationship could be established, leading to higher "conditional lower bounds". This is more challenging than reducing from Set-Cover due to the more complicated nature of their quantum algorithm.

References

1 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight Hardness Results for LCS and other Sequence Similarity Measures. In *Proc. of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 59–78, 2015.

[6] By following their algorithm more closely one can get a slightly higher lower bound for the following asymmetric version of k-SUM: Given $k/2$ lists of n_1 numbers and $k/2$ lists of n_2 numbers, where $n_1 = O(n_2^{\varepsilon})$ decide if there is a k-SUM.

2 Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. SETH-Based Lower Bounds for Subset Sum and Bicriteria Path. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 41–57, 2019.

3 Amir Abboud and Kevin Lewi. Exact Weight Subgraphs and the k-sum Conjecture. In *Proc. of the 40th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 1–12, 2013.

4 Amir Abboud, Kevin Lewi, and R. Ryan Williams. Losing Weight by Gaining Edges. In *Proc. of the 22th annual European Symposium on Algorithms (ESA)*, pages 1–12, 2014.

5 Andris Ambainis, Kaspars Balodis, Janis Iraids, Martins Kokainis, Krisjanis Prusis, and Jevgenijs Vihrovs. Quantum Speedups for Exponential-Time Dynamic Programming Algorithms. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1783–1793, 2019.

6 Andreas Björklund, Holger Dell, and Thore Husfeldt. The Parity of Set Systems Under Random Restrictions with Applications to Exponential Time Problems. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 231–242, 2015.

7 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *J. Comput. Syst. Sci.*, 87:119–139, 2017. doi:10.1016/j.jcss.2017.03.003.

8 Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set Partitioning via Inclusion-Exclusion. *SIAM J. Comput.*, 39(2):546–563, 2009. doi:10.1137/070683933.

9 Andreas Björklund, Petteri Kaski, and Lukasz Kowalik. Constrained Multilinear Detection and Generalized Graph Motifs. *Algorithmica*, 74(2):947–967, 2016. doi:10.1007/s00453-015-9981-1.

10 Karl Bringmann, Fabrizio Grandoni, Barna Saha, and Virginia Vassilevska Williams. Truly Subcubic Algorithms for Language Edit Distance and RNA-Folding via Fast Bounded-Difference Min-Plus Product. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 375–384, 2016.

11 Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proc. of the 7th ACM Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 261–270, 2016.

12 Jianer Chen, Benny Chor, Mike Fellows, Xiuzhen Huang, David W. Juedes, Iyad A. Kanj, and Ge Xia. Tight lower bounds for certain parameterized NP-hard problems. *Inf. Comput.*, 201(2):216–231, 2005.

13 Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.*, 72(8):1346–1367, 2006.

14 Lijie Chen and Ruosong Wang. Classical Algorithms from Quantum and Arthur-Merlin Communication Protocols. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 23:1–23:20, 2019.

15 Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as CNF-SAT. *ACM Transactions on Algorithms*, 12(3):41, 2016.

16 Marek Cygan, Marcin Mucha, Karol Węgrzycki, and Michał Włodarczyk. On problems equivalent to (min,+)-convolution. *ACM Transactions on Algorithms (TALG)*, 15(1):14, 2019.

17 Christoph Dürr and Peter Høyer. A Quantum Algorithm for Finding the Minimum. *CoRR*, quant-ph/9607014, 1996. arXiv:quant-ph/9607014.

18 Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1):57–67, 2004.

19 Fedor V Fomin, Dieter Kratsch, and Gerhard J Woeginger. Exact (exponential) algorithms for the dominating set problem. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 245–256. Springer, 2004.

20 Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry*, 5(3):165–185, 1995.

21 François Le Gall. Powers of tensors and fast matrix multiplication. In *International Symposium on Symbolic and Algebraic Computation, ISSAC '14, Kobe, Japan, July 23-25, 2014*, pages 296–303, 2014.

22 Lov K Grover. A fast quantum mechanical algorithm for database search. *arXiv preprint*, 1996. arXiv:quant-ph/9605043.

23 Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

24 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

25 Kamil Khadiev. Quantum Dynamic Programming Algorithm for DAGs. Applications for AND-OR DAG Evaluation and DAG's Diameter Search. *CoRR*, abs/1804.09950, 2018. arXiv:1804.09950.

26 Lukasz Kowalik and Juho Lauri. On finding rainbow and colorful paths. *Theor. Comput. Sci.*, 628:110–114, 2016. doi:10.1016/j.tcs.2016.03.017.

27 Robert Krauthgamer and Ohad Trabelsi. The Set Cover Conjecture and Subgraph Isomorphism with a Tree Pattern. *arXiv preprint*, 2017. arXiv:1711.08041.

28 R Krithika, Abhishek Sahu, and Prafullkumar Tale. Dynamic parameterized problems. *Algorithmica*, 80(9):2637–2655, 2018.

29 Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. On the Fine-Grained Complexity of One-Dimensional Dynamic Programming. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 21:1–21:15, 2017.

30 Aran Nayebi and Virginia Vassilevska Williams. Quantum algorithms for shortest paths problems in structured instances. *CoRR*, abs/1410.6220, 2014. arXiv:1410.6220.

31 Jesper Nederlof. Finding Large Set Covers Faster via the Representation Method. In *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, pages 69:1–69:15, 2016.

32 J. Nešetřil and S. Poljak. On the complexity of the subgraph problem. *Commentationes Math. Universitatis Carolinae*, 26(2):415–419, 1985.

33 Mihai Pătraşcu and Ryan Williams. On the possibility of faster SAT algorithms. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1065–1075. SIAM, 2010.

34 Ohad Trabelsi. Nearly Optimal Time Bounds for kPath in Hypergraphs. *CoRR*, abs/1803.04940, 2018. arXiv:1803.04940.

35 Leslie G. Valiant. General Context-Free Recognition in Less than Cubic Time. *J. Comput. Syst. Sci.*, 10(2):308–315, 1975. doi:10.1016/S0022-0000(75)80046-8.

36 R. Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2–3):357–365, 2005.

37 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity.

38 Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th symposium on Theory of Computing*, pages 887–898. ACM, 2012.

39 Virginia Vassilevska Williams. Hardness of Easy Problems: Basing Hardness on Popular Conjectures such as the Strong Exponential Time Hypothesis (Invited Talk). In *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, pages 17–29, 2015.

40 Virginia Vassilevska Williams. Some Open Problems in Fine-Grained Complexity. *SIGACT News*, 49(4):29–35, 2018. doi:10.1145/3300150.3300158.

41 Virginia Vassilevska Williams and R. Ryan Williams. Subcubic Equivalences Between Path, Matrix, and Triangle Problems. *J. ACM*, 65(5):27:1–27:38, 2018. doi:10.1145/3186893.

Geometric Multicut

Mikkel Abrahamsen
BARC, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark
miab@di.ku.dk

Panos Giannopoulos
giCenter, Department of Computer Science, City University of London, EC1V 0HB, London, UK
panos.giannopoulos@city.ac.uk

Maarten Löffler
Department of Information and Computing Sciences, Utrecht University, The Netherlands
m.loffler@uu.nl

Günter Rote
Institut für Informatik, Freie Universität Berlin, Takustraße 9, 14195 Berlin, Germany
rote@inf.fu-berlin.de

Abstract

We study the following separation problem: Given a collection of colored objects in the plane, compute a shortest "fence" F, i.e., a union of curves of minimum total length, that separates every two objects of different colors. Two objects are separated if F contains a simple closed curve that has one object in the interior and the other in the exterior. We refer to the problem as GEOMETRIC k-CUT, where k is the number of different colors, as it can be seen as a geometric analogue to the well-studied multicut problem on graphs. We first give an $O(n^4 \log^3 n)$-time algorithm that computes an optimal fence for the case where the input consists of polygons of two colors and n corners in total. We then show that the problem is NP-hard for the case of three colors. Finally, we give a $(2 - 4/3k)$-approximation algorithm.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Computational geometry

Keywords and phrases multicut, clustering, Steiner tree

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.9

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1902.04045`.

Funding *Mikkel Abrahamsen*: Supported by the Innovation Fund Denmark through the DABAI project. MA is also a part of BARC, Basic Algorithms Research Copenhagen, supported by the VILLUM Foundation grant 16582.
Maarten Löffler: Partially supported by the Netherlands Organisation for Scientific Research (NWO); 614.001.504.

Acknowledgements This work was initiated at the workshop on *Fixed-Parameter Computational Geometry* at the Lorentz Center in Leiden in May 2018. We thank the organizers and the Lorentz Center for a nice workshop and Michael Hoffmann for useful discussions during the workshop.

1 Introduction

Problem Definition. We are given k pairwise interior-disjoint, not necessarily connected, sets $B_1, B_2, \ldots, B_k$ in the plane. We want to find a covering of the plane $\mathbb{R}^2 = \bar{B}_1 \cup \bar{B}_2 \cup \cdots \cup \bar{B}_k$ such that the sets $\bar{B}_i$ are closed and interior-disjoint, $B_i \subseteq \bar{B}_i$ and the total length of the boundary $F = \bigcup_{i=1}^{k} \partial \bar{B}_i$ between the different sets $\bar{B}_i$ is minimized.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 9; pp. 9:1–9:15


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

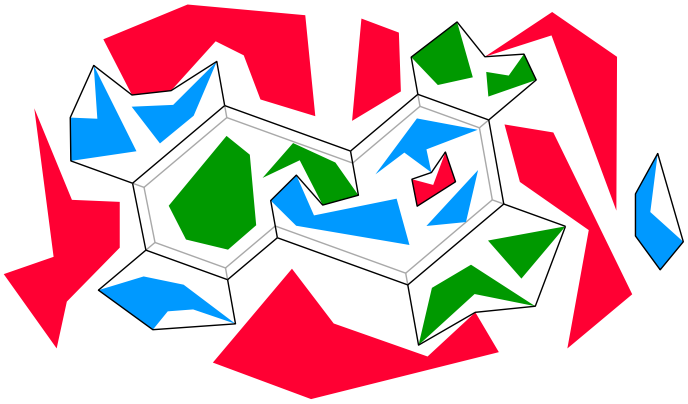


Figure 1 An instance of GEOMETRIC 3-CUT and an optimal fence in black. The fence contains a cycle that does not touch any object. The grey fence shows how the cycle can be shrunk without changing the total length of the fence.

We think of the k sets B_i as having k different *colors* and each set B_i as a union of simple geometric objects like circular disks and simple polygons. An example is shown in Figure 1. We call $\bar{B}_i$ the *territory* of color i. The "fence" F is the set of points that separates the territories. (Alternatively, F is the set of points belonging to more than one territory.) As we can see, a territory can have more than one connected component.

An alternative view of the problem concentrates on the *fence*: A fence is defined as a union of curves F such that each connected component of $\mathbb{R}^2 \setminus F$ intersects at most one set B_i. An interior-disjoint covering as defined above gives, by definition, such a fence. Likewise, a fence F induces such a covering, by assigning each connected component of $\mathbb{R}^2 \setminus F$ to an appropriate territory $\bar{B}_i$. The total length of a fence F is also called the *cost* of F and is denoted as $|F|$.

In our paper, we will focus on the case where the input consists of simple polygons (with disjoint interiors). We refer to this problem as *GEOMETRIC k-CUT*. Each input polygon is called an *object*. We use n to denote the total number of corners of the input polygons, counted with multiplicity.

Even in this simple setting, the problem poses both geometric and combinatorial difficulties. A set B_i can consist of disconnected pieces, and the combinatorial challenge is to choose which of the pieces should be grouped into the same component of $\bar{B}_i$. The geometric task is to construct a network of curves that surrounds the given groups of objects and thus separates the groups from each other. For $k = 2$ colors, optimal fences consist of geodesic curves around obstacles, which are well understood. As soon as the number k of colors exceeds 2, the geometry becomes more complicated, and the problem acquires traits of the geometric Steiner tree problem, as shown by the example in Figure 1.

The problem of enclosing a set of objects by a shortest system of fences has been considered with a single set B_1 by Abrahamsen et al. [1]. The task is to "enclose" the components of B_1 by a shortest system of fences. This can be formulated as a special case of our problem with $k = 2$ colors: We add an additional set B_2, far away from B_1 and large enough so that it is never optimal to enclose B_2. Thus, we have to enclose all components of B_1 and separate them from the unbounded region. In this setting, there will be no nested fences. Abrahamsen et al. gave an algorithm with running time $O(n \text{ polylog } n)$ for the case where the input consists of n unit disks.

Applications. Besides being a natural problem in its own right, the geometric multicut problem may well find applications in image processing and computer vision. As we describe in Section 3, a problem closely related to the case $k = 2$ has been studied from the perspective of image segmentation. Simplified slightly, we are given a picture with some pixels known to be black or white, and we have to choose colors for the remaining pixels so as to minimize the boundary between black and white regions. The problem for $k > 2$ is equally well-motivated in this context, although we have not found any explicit references to it (perhaps because of the NP-hardness that we will prove in this case).

Our Results. In Section 3, we show how to solve the case with $k = 2$ colors in time $O(n^4 \log^3 n)$. The algorithm works by reducing the problem to the multiple-source multiple-sink maximum flow problem in a planar graph. In Section 4, we show that already the case with $k = 3$ colors is NP-hard by a reduction from PLANAR POSITIVE 1-IN-3-SAT.

In Section 5, we discuss approximation algorithms. We first compare the optimal fence $F_{\mathcal{A}}$ consisting of line segments between corners of input polygons to the unrestricted optimal fence F^*. We show that $|F_{\mathcal{A}}| \leq 4/3 \cdot |F^*|$. After applying a $(3/2 - 1/k)$-approximation algorithm for the k-terminal multiway cut problem [6], we obtain a polynomial-time $(2 - \frac{4}{3k})$-approximation algorithm for GEOMETRIC k-CUT (Theorem 11).

Due to restricted space, many details and proofs have been removed and can be found in the full version [2].

2 Structure of Optimal Fences

▶ **Lemma 1.** *An optimal fence F^* is a union of (not necessarily disjoint) closed curves, disjoint from the interior of the objects. Furthermore, F^* is the union of straight line segments of positive length. Consider two non-collinear line segments $\ell_1, \ell_2 \subset F^*$ with a common endpoint p. If p is not a corner of an object, then exactly three line segments meet at p and form angles of $2\pi/3$.*

Proof. It is clear that an optimal fence F^* never enters the interior of an object.

We next show that F^* is the union of a set of closed curves. Suppose not. Let $F' \subset F^*$ be the union of all closed curves contained in F^* and let π be a connected component in $F^* \setminus F'$. Then π is the (not necessarily disjoint) union of a set of open curves, which do not contribute to the separation of any objects. Hence, $F^* \setminus \pi$ is a fence of smaller length than F^*, so F^* is not optimal.

In a similar way, one can consider the union L of all line segments of positive length contained in F^*, and if $F^* \setminus L$ is non-empty, a curve π in $F^* \setminus L$ can be replaced by a shortest path homotopic to it, which consists of a sequence of line segments. (See the proof of Lemma 13 in the full version.)

The last claimed property is shared with the Euclidean Steiner minimal tree on a set of points in the plane, and it can be proved in the same easy way by local optimality arguments, see for example Gilbert and Pollak [10]. ◀

As it can be seen in Figure 1, optimal fences may contain cycles that do not touch any object. As is also indicated in the figure, such a cycle can be shrunk until it eventually hits an object and is eliminated. This does not increase the length, so there is always an optimal fence with no cycle disjoint from all objects. See the full version for the details.

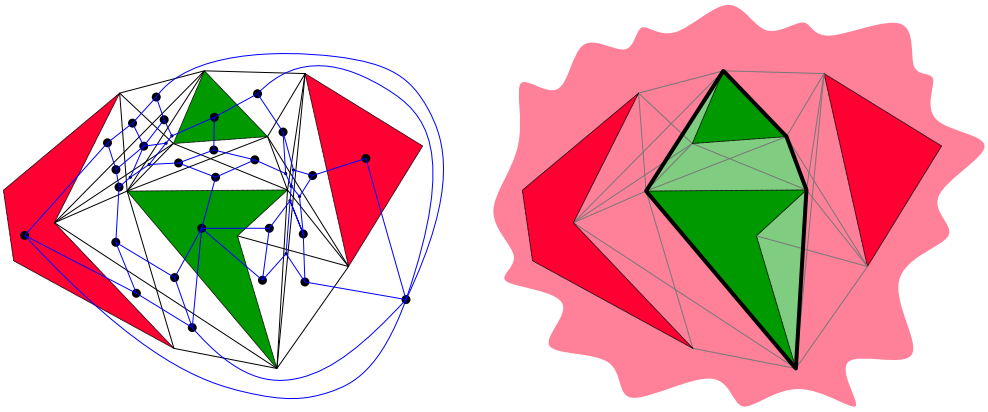


Figure 2 Left: The arrangement $\mathcal{A}$ induced by an instance of GEOMETRIC 2-CUT with two green and two red objects. The edges of the dual graph G are blue. Right: The optimal solution.

3 The Bicolored Case

In this section we consider the case of $k = 2$ different colors. Let N be the set of all corners of the objects. A line segment is said to be *free* if it is disjoint from the interior of every object. A vertex v of an optimal fence cannot have degree 3 or more unless $v \in N$, as otherwise two of the regions meeting at v would be part of the same territory and could be merged, thus reducing the length. We therefore get the following consequence of Lemma 1.

▶ **Lemma 2.** *An optimal fence consists of free line segments with endpoints in N.* ◀

Let S be the set of all free segments with endpoints in N. S includes all edges of the objects. Let $\mathcal{A}$ be the arrangement induced by S, see Figure 2. Consider an optimal fence F^* and the associated territories $\bar{B}_1$ and $\bar{B}_2$. Lemma 2 implies that F^* is contained in $\mathcal{A}$. Thus, each cell of $\mathcal{A}$ belongs entirely either to $\bar{B}_1$ or $\bar{B}_2$. The objects are cells of $\mathcal{A}$ whose classification (i.e., membership of $\bar{B}_1$ versus $\bar{B}_2$) is fixed. In order to find F^*, we need to select the territory that each of the other cells belongs to. Since $|S| = O(n^2)$, $\mathcal{A}$ has size $O(|S|^2) = O(n^4)$ and can be computed in $O(|\mathcal{A}|) = O(n^4)$ time [7]. For simplicity, we stick with the worst-case bounds. In practice, set S can be pruned by observing that the edges of an optimal fence must be *bitangents* that touch the objects in a certain way, because the curves of the fence are locally shortest.

Finding an optimal fence amounts to minimizing the boundary between $\bar{B}_1$ and $\bar{B}_2$. This can be formulated as a minimum-cut problem in the dual graph $G(V, E)$ of the arrangement $\mathcal{A}$. There is a node in V for each cell and a weighted edge in E for each pair of adjacent cells: the weight of the edge is the length of the cells' common boundary. Let $S_1, S_2 \subset V$ be the sets of cells that contain the objects of B_1, B_2, respectively. We need to find the minimum cut that separates S_1 from S_2. This can be obtained by finding the maximum flow in G from the sources S_1 to the sinks S_2, where the capacities are the weights. As G is a planar graph, we can use the algorithm by Borradaile et al. [5] with running time $O(|V| \log^3 |V|)$. The running time has since then been improved to $O(\frac{|V| \log^3 |V|}{\log^2 \log |V|})$ [9]. As $|V| = O(|S|^2) = O(n^4)$, we obtain the following theorem.

▶ **Theorem 3.** *GEOMETRIC 2-CUT can be solved in time $O(\frac{n^4 \log^3 n}{\log^2 \log n})$, where n is the total number of corners of the objects.*

A similar algorithm has been described before in a slightly different context: image segmentation [11], see also [5]. Here, we have a rectangular grid of pixels, each having a given gray-scale value. Some pixels are known to be either black or white. The remaining pixels have to be assigned either the black or the white color. Each pixel has edges to its (at most four) neighbors. The weights of these edges can be chosen in such a way that the minimum cut problem corresponds to minimizing a cost function consisting of two parts: One part, the *data component*, has a term for each pixel, and it measures the discrepancy between the gray-value of the pixel and the assigned value. The other part, the *smoothing component*, penalizes neighboring pixels with similar gray-values that are assigned different colors.

4 Hardness of the Tricolored Case

We show how to construct an instance I of GEOMETRIC 3-CUT from an instance Φ of PLANAR POSITIVE 1-IN-3-SAT. For ease of presentation, we first describe the reduction geometrically, allowing irrational coordinates. We prove that if Φ is satisfiable, then I has a fence of cost M^*, whereas if Φ is not satisfiable, then the cost is at least $M^* + 1/50$. We then argue that the corners can be slightly moved to make a new instance I' with rational coordinates while still being able to distinguish whether Φ is satisfiable or not, based on the cost of an optimal fence.

In order to make the proof as simple as possible, we introduce a new specialized problem COLORED TRIGRID POSITIVE 1-IN-3-SAT in the following.

4.1 Auxiliary NP-complete problems

▶ **Definition 4.** *In the POSITIVE 1-IN-3-SAT problem, we are given a collection Φ of clauses containing exactly three distinct variables (none of which are negated). The problem is to decide whether there exists an assignment of truth values to the variables of Φ such that exactly one variable in each clause is true.*

▶ **Definition 5.** *In the TRIGRID POSITIVE 1-IN-3-SAT problem, we are given an instance Φ of POSITIVE 1-IN-3-SAT together with a planar embedding of an associated graph $G(\Phi)$ with the following properties:*

- *$G(\Phi)$ is a subgraph of a regular triangular grid,*
- *for each variable x, there is a simple cycle v_x,*
- *for each clause $C = \{x, y, z\}$, there is a path c_C and three vertical paths $\ell_x^C, \ell_y^C, \ell_z^C$ with one endpoint at a vertex of c_C and one at a vertex of each of v_x, v_y, v_z,*
- *except for the described incidences, no edges share a vertex,*
- *all vertices have degree 2 or 3,*
- *any two adjacent edges form an angle of π or $2\pi/3$,*
- *the number of vertices is bounded by a quadratic function of the size of Φ.*

The problem is to decide whether Φ has a satisfying assignment (see Definition 4).

Mulzer and Rote [13] showed that another problem, PLANAR POSITIVE 1-IN-3-SAT, is NP-complete, which is similar but uses a slightly different embedding with axis-parallel segments. It trivially follows that TRIGRID POSITIVE 1-IN-3-SAT is also NP-complete, see Figure 3.

Consider an instance $(\Phi, G(\Phi))$ of TRIGRID POSITIVE 1-IN-3-SAT. There are some vertices of degree three on the cycles v_x corresponding to each variable x in Φ, and these we denote as *branch vertices* of $G(\Phi)$. There is also one vertex of degree three on the path c_C corresponding to each clause C in Φ, which we denote as a *clause vertex*. Except for branch and clause vertices, at most two edges meet at each vertex.

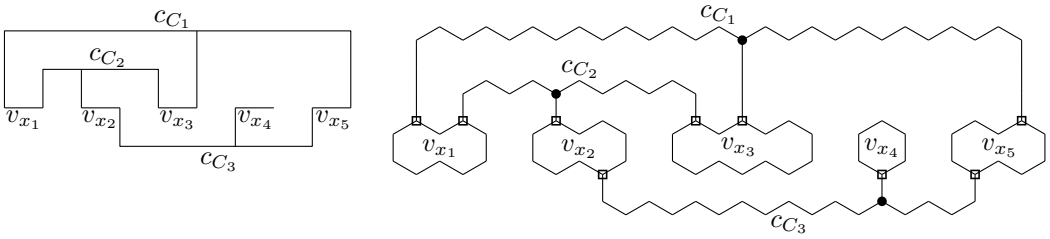


Figure 3 Left: An instance of PLANAR POSITIVE 1-IN-3-SAT for the formula $\Phi = C_1 \wedge C_2 \wedge C_3$ for $C_1 = x_1 \vee x_3 \vee x_5$, $C_2 = x_1 \vee x_2 \vee x_3$, and $C_3 = x_2 \vee x_4 \vee x_5$. Right: A corresponding instance of TRIGRID POSITIVE 1-IN-3-SAT. Clause vertices are drawn as dots and branch vertices as boxes.

Let $\mathcal{C}$ be the set of all clause vertices (considered as geometric points). Removing $\mathcal{C}$ from $G(\Phi)$ (considered as a subset of $\mathbb{R}^2$) splits $G(\Phi)$ into one connected component E_x for each variable x of Φ. The idea of our reduction to GEOMETRIC 3-CUT is to build a *channel* on top of E_x for each variable x. The channel has constant width $1/2$ and contains E_x in the center. The channel contains small *inner* objects and is bounded by larger *outer objects* of another color. There will be two equally good ways to separate the inner and outer objects, namely taking an individual fence around each inner object and taking long fences along the boundaries of the channel that enclose as many inner objects as possible. As it will turn out, any other way of separating the inner from the outer objects will require more fence. These two optimal fences play the roles of x being true and false, respectively.

At the clause vertices where three regions E_x, E_y, E_z meet, we make a clause gadget that connects the three channels corresponding to x, y, z. The objects in the clause gadget can be separated using the least amount of fence if and only if one of the channels is in the state corresponding to true and the other two are in the false state. Therefore, this corresponds to the clause in Φ being satisfied.

In order to make this idea work, we first assign every edge of $G(\Phi)$ an *inner* and an *outer* color among $\{\text{red}, \text{green}, \text{blue}\}$. These will be used as the colors of the inner and outer objects of the channel later on. We require the following of the coloring:

1. The inner and outer colors of any edge are distinct.
2. Any two adjacent collinear edges have the same inner or outer color.
3. Any two adjacent edges that meet at an angle of $2\pi/3$ at a non-clause vertex have the same inner and the same outer color.
4. The inner colors of the three edges meeting at a clause vertex are red, green, blue in clockwise order, while the outer colors of the same edges are blue, red, green, respectively.

We now introduce the problem COLORED TRIGRID POSITIVE 1-IN-3-SAT, which we will reduce to GEOMETRIC 3-CUT, see Figure 4. The problem is NP-complete, as shown in the full version.

▶ **Definition 6.** *In COLORED TRIGRID POSITIVE 1-IN-3-SAT, we are given an instance* $(\Phi, G(\Phi))$ *of TRIGRID POSITIVE 1-IN-3-SAT together with a coloring of the edges of* $G(\Phi)$ *satisfying the above requirements. We want to decide whether* Φ *has a satisfying assignment.*

4.2 Building a GEOMETRIC 3-SAT instance from tiles

Consider an instance $(\Phi, G(\Phi))$ of COLORED TRIGRID POSITIVE 1-IN-3-SAT that we will reduce to GEOMETRIC 3-CUT. We make the construction using hexagonal *tiles* of six different types, namely *straight*, *inner color change*, *outer color change*, *bend*, *branch*, and *clause* tiles. Each tile is a regular hexagon with side length $1/\sqrt{3}$ and hence has width 1. The tiles are rotated such that they have two horizontal edges.

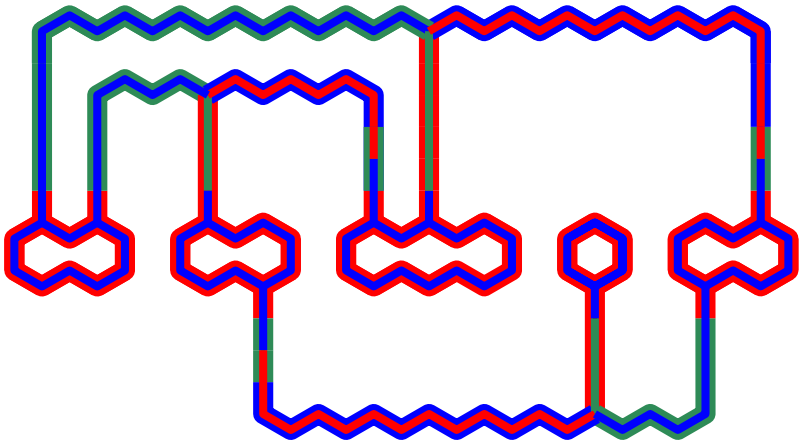


Figure 4 An instance of COLORED TRIGRID POSITIVE 1-IN-3-SAT based on the instance from Figure 3.

The tiles are placed so that each tile is centered at a vertex p of $G(\Phi)$. Let G_p be the part of $G(\Phi)$ within distance $1/2$ from p (recall that each edge of $G(\Phi)$ has length 1). Figure 5 shows the tiles and how they are placed according to the shape and colors of G_p.

In order to define the outer objects of a tile, we consider the straight skeleton offset [3, 4] of G_p at distance $1/4$. With the exception of the bend tile, this offset is the same as the Euclidean offset. By the *outer* and *inner region*, we mean the region of the tile outside, resp. inside, this offset. The outer objects cover the outer region, and every point is colored with the outer color of a closest edge in G_p. The inner region is empty except for the inner objects described in each case below. We suppose that $p = (0, 0)$.

The straight tile. If two collinear edges meet at p with the same inner and outer color, we use a straight tile. Suppose in this and the following two cases that G_p is the vertical line segment from $(0, -1/2)$ to $(0, 1/2)$ – tiles for edges of other slopes are obtained by rotation of the ones described here. There are four axis-parallel squares of the inner color of G_p with side length $1/8$ centered at $(\pm(1/4 - 1/16), \pm 1/4)$. This size is chosen so their total perimeter is 2, which is the length of the common boundary of the inner and outer regions.

The inner color change tile. If two collinear edges meet at p with different inner colors, we use an inner color change tile. There are again four squares colored in the inner color of the closest point in G_p. There are also four smaller axis-parallel squares with side length $1/28$ centered at $(\pm(1/4 - 1/56), \pm 1/56)$, likewise colored in the inner color of the closest point in G_p. The size of these small squares is chosen so that they can be individually enclosed using fences of total length $14 \cdot 1/28 = 1/2$, which is the width of the inner region.

The outer color change tile. If two collinear edges meet at p with different outer colors, we use an outer color change tile. There are four axis-parallel squares of the inner color of G_p with side length $3/32$. Their centers are $(\pm(1/4 - 3/64), \pm 1/4)$. The size of these squares is chosen so that their total perimeter is $2 - 1/2 = 3/2$.

The bend tile. If two non-collinear edges meet at p, we use a bend tile. Consider the case where G_p is the vertical line segment from p to $(0, 1/2)$ and the segment of length $1/2$ from p with direction $(\cos \pi/6, -\sin \pi/6)$. The other cases are obtained by a suitable rotation of this

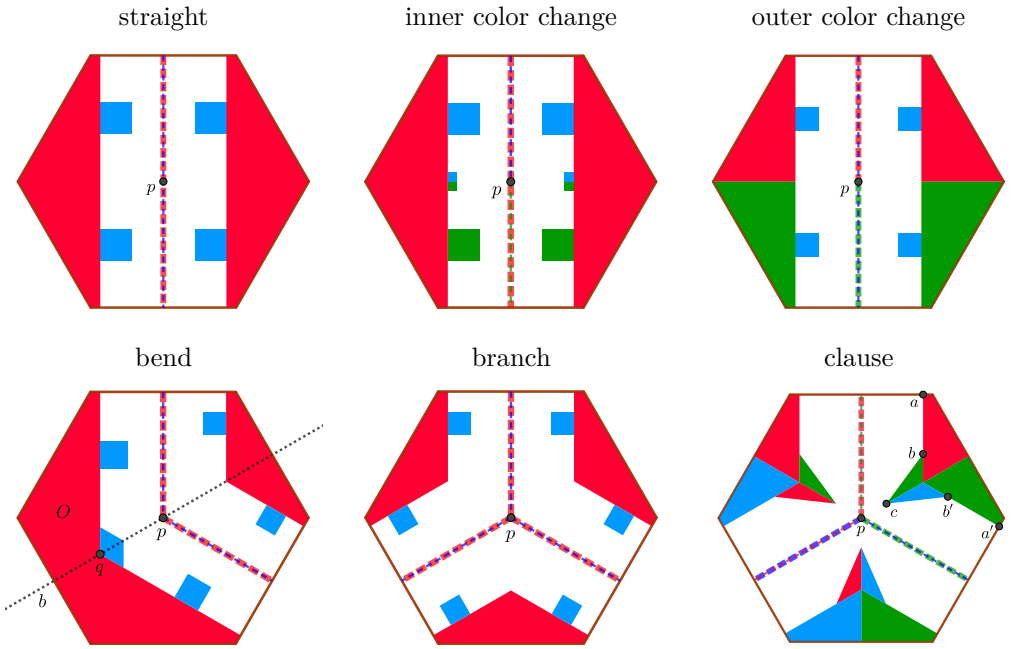


Figure 5 Different kinds of tiles used in the reduction to GEOMETRIC 3-CUT. The dashed colored segments show G_p and the inner and outer color of G_p. The tiles are colored accordingly. The points in the clause tile are defined so that $\|ab\| = \|a'b'\| = 6/25 = 0.24$ and $\|bc\| = \|b'c\| = 1/4 = 0.25$. Point c has coordinates $(x, x/\sqrt{3})$, where $x = \frac{13\sqrt{3}}{200} + 3/16 - \frac{\sqrt{-459+3900\sqrt{3}}}{400}$ is a solution to $10000x^2 + (-1300\sqrt{3} - 3750)x + 507 = 0$. The remaining points in the tile are given by rotations by angles $2\pi/3$ and $4\pi/3$ around p.

tile. There is an axis parallel square of side length $x = \frac{6+\sqrt{3}}{72}$ with center $(-(1/4 - x/2), 1/4)$ and another with side length $y = \frac{6-\sqrt{3}}{48}$ centered at $(1/4 - y/2, 3/8)$. The tile is symmetric with respect to the angular bisector b of G_p, and so the reflections of the described squares with respect to b are also inner objects. Note that there are two outer objects, one of which, O, has a concave corner q with exterior angle $2\pi/3$. We place a parallelogram with side length x, a corner at q, and two edges contained in the edges of O incident at q. It is easy to verify that the common boundary of the inner and outer regions has a total length of 2; the inner objects are chosen such that their total perimeter is also 2.

The branch tile. If p is a branch vertex, we use the branch tile. There are two cases: G_p either contains the vertical segment from p to $(0, 1/2)$ or that from p to $(0, -1/2)$. We specify the tile in the first case – the other can be obtained by a rotation of π. There are axis-parallel squares of side length $y = \frac{6-\sqrt{3}}{48}$ centered at $(\pm(1/4 - y/2), 3/8)$ and their rotations around p by angles $2\pi/3$ and $4\pi/3$. The common boundary of the inner and outer regions has a total length of $\frac{6-\sqrt{3}}{2}$, and the total perimeter of the inner objects is also $\frac{6-\sqrt{3}}{2}$.

The clause tile. If p is a clause vertex, we use the clause tile (defined in Figure 5). The other clause tiles are given by rotations of the described tile by angles $k\pi/3$ for $k = 1, \ldots, 5$.

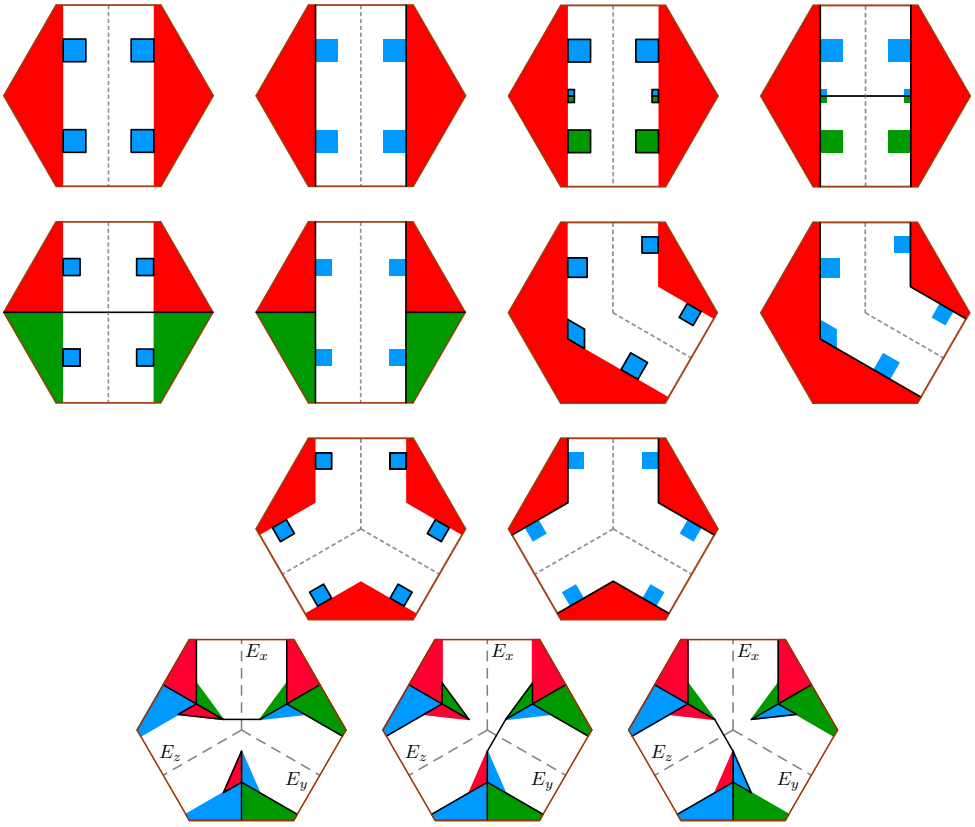


Figure 6 The optimal solutions to each type of tile. The edges in G_p are shown in dashed grey. We denote the left solution of each of the first five types of tiles as the *outer* solution and the other as the *inner* solution. For the clause tile, we define the solution as the *z-outer*, *x-outer*, and *y-outer* solution in order from left to right, respectively.

4.3 Solving the tiles

Let an instance I of GEOMETRIC 3-SAT be given together with an associated fence $\mathcal{F}$. Consider the restriction of I to a convex polygon P and the part of the fence $\mathcal{F} \cap P$ inside P. Note that $\mathcal{F} \cap P$ consists of (not necessarily disjoint) closed curves and open curves with endpoints on the boundary ∂P, such that no two objects in P of different color can be connected by a path $\pi \subset P$ unless π intersects $\mathcal{F}$. (An open curve is a subset of a larger closed curve of $\mathcal{F}$ that continues outside P.) We say that a set of closed and open curves in P with that property is a *solution* to $I \cap P$. In the following, we analyze the solutions to the tiles defined in Section 4.2 in order to characterize the solutions of minimum cost. We say that two closed curves (disjoint from the interiors of the objects) are *homotopic* if one can be continuously deformed into the other without entering the interiors of the objects. Two open curves with endpoints on the boundary of the tile are homotopic if they are subsets of two homotopic closed curves (that extend outside the tile).

The following lemma characterizes the optimal solutions to each type of tile. The statement is that if a solution is not too much more expensive than the solutions shown in Figure 6, then it will contain curves homotopic to each curve in one of the solutions in the figure. The proof is deferred to the full version.

▶ **Lemma 7.** *Figure 6 shows optimal solutions to each kind of tile. The cost in each case is: Straight tile:* 2*. Inner color change tile:* 5/2*. Outer color change tile:* $\left(\frac{2}{\sqrt{3}} - \frac{1}{2}\right) + 2 \approx 2.65$*. Bend tile:* 2*. Branch tile:* $\frac{6-\sqrt{3}}{2} \approx 2.13$*. Clause tile:* ≈ 3.51 *(the exact value is complicated due to the coordinates and of no use).*

If the cost of a solution $\mathcal{F}$ to a tile T exceeds the optimum by less than 1/50*, then $\mathcal{F}$ is homotopic to one of the optimal solutions $\mathcal{F}^*$ of T in the following sense: For each curve π^* in $\mathcal{F}^*$, there is a curve π in $\mathcal{F}$ homotopic to π^*. If π is closed, the distance from any point on π to the closest point on π^* is less than $\sqrt{(1/8 + 1/100)^2 - (1/8)^2} < 0.06$. If π is open and π^* has an endpoint f^*, there is a corresponding endpoint f of π with $\|f^* f\| < 1/10$.*

▶ **Theorem 8.** *The problem GEOMETRIC* 3*-CUT is NP-hard.*

Proof. Let an instance $(\Phi, G(\Phi))$ of COLORED TRIGRID POSITIVE 1-IN-3-SAT be given and construct the tiles on top of $G(\Phi)$ as described. Let $\mathcal{T}$ be the set of tiles and $\mathcal{A}$ the area that the tiles cover (i.e., $\mathcal{A}$ is a union of the hexagons). We will cover any holes in $\mathcal{A}$ with completely red tiles, and place red tiles all the way along the exterior boundary of $\mathcal{A}$. Let $\mathcal{R}$ be the set of these added red tiles and let I be the resulting instance of GEOMETRIC 3-CUT. It is now trivial how to place the fences in I everywhere except in the interior of $\mathcal{A}$.

Consider a fence $\mathcal{F}$ to the obtained instance with cost M. Let M^* be the sum of the cost of an optimal solution to each tile in $\mathcal{T}$ plus the cost of the fence that must be placed along the boundaries of the added red tiles in $\mathcal{R}$. We claim that if Φ is satisfiable, then a solution realizing the minimum M^* exists. Furthermore, if $M < M^* + 1/50$, then Φ is satisfiable.

Suppose that Φ is satisfiable and fix a satisfying assignment. Consider a clause tile where E_x, E_y, and E_z meet. Now, we choose the v-outer state, where $v \in \{x, y, z\}$ is the variable that is satisfied. For each non-clause tile that covers a part of E_w for a variable w of Φ, we choose the outer state if w is true and the inner otherwise. It is now easy to see that the curves form a fence of the desired cost.

On the other hand, suppose that $M < M^* + 1/50$. It follows that in each tile in $\mathcal{T}$, the cost exceeds the optimum by at most 1/50. Hence, the solution in each tile is homotopic to one of the optimal states as described in Lemma 7. We now claim that the states of all tiles representing one variable must agree on either the inner or outer state. Consider two adjacent tiles where one is in the inner state. There are open curves with endpoints on the shared edge of the two tiles with a distance of more than $1/2 - 2 \cdot 1/10 = 3/10$. The other tile cannot be in the outer state, because then there would have to be an extra open curve of length at least 3/10 to connect those endpoints. It follows that the other tile must also be in the inner state. Thus, both tiles are either in the inner or in the outer state, as desired.

We now describe how to obtain a satisfying assignment of Φ. Consider a clause tile where E_x, E_y, and E_z meet and suppose the tile is in the x-outer state. It follows from the above that each tile covering E_x is in the outer state or, in the case of the clause tile, in the x-outer state. Similarly, each non-clause tile covering only E_y (resp. E_z) is in the inner state and each clause tile covering a part of E_y (resp. E_z) is not in the y-outer (resp. z-outer) state. We now set x to true and y and z to false and do similarly with the other clause tiles, and it follows that we get a solution to Φ.

The proof that we can avoid the use of irrational corners is deferred to the full version. The basic idea is as follows. For each object O with corner v with an irrational coordinate, we choose a substitute $v' \in O$ with rational coordinates such that $\|vv'\| < \frac{1/50}{4n}$ and such that v' only requires polynomially many bits to represent. This results in a modified instance I', and we prove that I' has a solution of cost $M' := \frac{\lceil 100 M^* \rceil}{100}$ if and only if Φ is satisfiable. ◀

5 Approximation

The approach for $k = 2$ from Section 3 does not extend to $k \geq 3$ because Lemma 2 does not apply: The arrangement $\mathcal{A}$ (formed by the free segments between the corners N of the input objects) is no longer guaranteed to contain an optimal fence, see Figure 1. However, we can still search for an approximate solution in $\mathcal{A}$: We show that the optimal fence $F_{\mathcal{A}}$ contained in $\mathcal{A}$ has a cost which is at most 4/3 times higher than the true optimal fence $F^\star$ (Theorem 9). In the full version, we construct a corresponding lower-bound example with $|F_{\mathcal{A}}| > 1.15 \cdot |F^\star|$.

The graph-theoretic problem that we then have to solve in the weighted dual graph $G = (V, E)$ of $\mathcal{A}$ is the *colored multiterminal cut problem*: We have terminals of $k \geq 3$ different colors and want to make a cut that separates every pair of terminals of different colors. This problem is NP-hard, but we can use approximation algorithms, see Section 5.1.

▶ **Theorem 9.** $|F_{\mathcal{A}}| \leq 4/3 \cdot |F^\star|$.

Proof. From Section 2, we know that after cutting an optimal fence $F^\star$ at all points of N, the remaining components are Steiner minimal trees with leaves in N and internal *Steiner vertices* of degree 3, where three segments make angles of $2\pi/3$.

Consider such a Steiner tree T (Figure 7a). Since T is embedded in the plane, the leaves can be enumerated in cyclic order as $v_1, \ldots, v_m$. We will replace T by a connected system $\bar{T}$ of fences that connects the same set of leaves $v_1, \ldots, v_m$, but contains only segments from the arrangement $\mathcal{A}$. Furthermore, we prove that the total length of $\bar{T}$ is bounded as $|\bar{T}| \leq \frac{4}{3}|T|$. Thus, carrying out this replacement for every Steiner tree leads to the fence $F_{\mathcal{A}}$ of the desired cost. If T consists of a single segment, we define $\bar{T}$ to be the same segment, in which case trivially $|\bar{T}| \leq \frac{4}{3}|T|$. Assume therefore that T has at least one Steiner vertex.

Let T_{ij} be the path in T from v_i to v_j. For each pair $\{i, j\}$, we define the path $\bar{T}_{ij}$ as the shortest path with the properties that

a) $\bar{T}_{ij}$ has endpoints v_i and v_j, and
b) $\bar{T}_{ij}$ is *homotopic* to T_{ij}: this means that T_{ij} can be continuously deformed into $\bar{T}_{ij}$ while keeping the endpoints fixed at v_i and v_j, without entering the interiors of the objects.

It is clear that

c) $\bar{T}_{ij}$ is contained in the arrangement $\mathcal{A}$, and
d) $\bar{T}_{ij}$ is at most as long as T_{ij}.

We will construct $\bar{T}$ as the union of paths $\bar{T}_{ij}$ that are specified by a certain set S of leaf pairs $\{i, j\}$, and we will show that its total length is bounded $|\bar{T}| \leq \frac{4}{3}|T|$. The fact that $F_{\mathcal{A}}$ is a valid fence is ensured by our choice of the set S, which we will now discuss.

If we overlay all paths T_{ij} for $\{i, j\} \in S$, we get a multigraph $\widetilde{T}$, which has the same vertices as T and uses the edges of T, some of them multiple times. We require these three properties:

1. Every edge of T is used once or twice in $\widetilde{T}$.
2. Every Steiner vertex of T has even degree (4 or 6) in $\widetilde{T}$. (By contrast, the degree in T is always 3.)
3. Any two paths T_{ij} and $T_{i'j'}$ that have a point of T in common must *cross* in the following sense: If we assume, by relabeling if necessary, that $i < j$ and $i' < j'$, then $i \leq i' \leq j \leq j'$ or $i' \leq i \leq j' \leq j$.

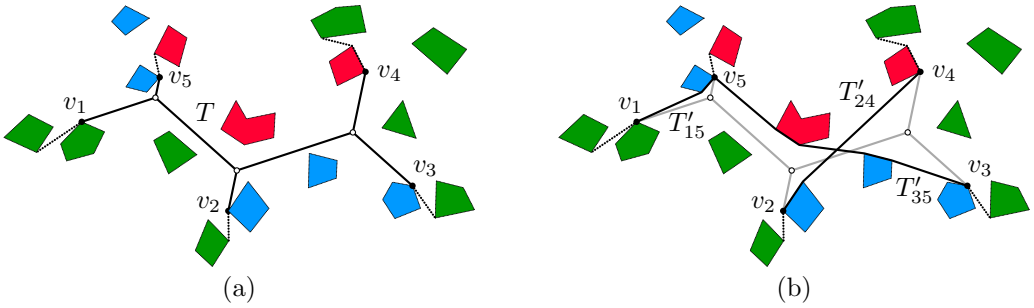


Figure 7 (a) a single Steiner tree T with 5 terminals $v_1, \ldots, v_5$, part of a larger fence system $F^\star$. Steiner vertices are white, leaves are black. (b) The transformed graph $\bar{T}$, formed as the union of three shortest homotopic paths $\bar{T}_{15}$, $\bar{T}_{24}$, and $\bar{T}_{35}$.

The last property is important to ensure that $\bar{T}$ is connected.

As we prove in the full version, Properties 1 and 3 imply that for any two leaves v_i and v_j (where the pair $\{i, j\}$ is not necessarily in S), the set $\bar{T}$ contains a path from v_i to v_j that is homotopic to the path T_{ij}. This means that after replacing T by $\bar{T}$ in $F^\star$, we get a system of fences F' that encloses and separates the same objects as $F^\star$, and thus we have indeed produced a valid fence.

To bound the length of $\bar{T}$, we bound each path $\bar{T}_{ij}$, $\{i, j\} \in S$, by the corresponding path T_{ij} in T. This upper estimate is simply the total length of T plus the length of the duplicated edges of T.

Our first task is to construct the multigraph $\widetilde{T}$. By Property 1, this boils down to selecting which edges of T to duplicate. In order to fulfill Property 2, we require that the degree of every inner vertex of $\widetilde{T}$ becomes even. (We show later that this is sufficient to ensure that the edges of $\widetilde{T}$ can be partitioned into paths T_{ij} subject to Property 3.)

▶ **Lemma 10.** *The edges that should be duplicated can be chosen such that their total length is at most* $|T|/3$.

Proof. For a particular tree, the optimum can be computed easily by dynamic programming, as follows. We root T at some arbitrary leaf. Consider a subtree U rooted at some vertex u of T such that u has one child v in U. We define U_1 and U_2 as the cost of the optimal set of duplicated edges in U, under the constraint that the multiplicity of the edge uv in $\widetilde{T}$ is 1 and 2, respectively.

By induction, we will establish that

$$2U_1 + U_2 \leq |U|. \tag{1}$$

This gives $\min\{U_1, U_2\} \leq |U|/3$ and proves the lemma, since this also holds for $U = T$. In the base case U has only one edge. Then $U_1 = 0$ and $U_2 = \|uv\| = |U|$, and (1) holds.

If U is larger, v has degree 3, and two subtrees L and R are attached there. If uv is not duplicated, then exactly one of the other edges incident to v has to be duplicated in order for v to get even degree in $\widetilde{T}$. On the other hand, if uv is duplicated, then either both or none of the other edges should be duplicated. Hence, we can compute U_1 and U_2 by the following recursion:

$$U_1 = \min\{L_1 + R_2, L_2 + R_1\} \tag{2}$$

$$U_2 = \min\{L_1 + R_1, L_2 + R_2\} + \|uv\| \tag{3}$$

We therefore get

$$U_1 \leq L_2 + R_1 \tag{4}$$
$$U_1 \leq L_1 + R_2 \tag{5}$$

from (2) and

$$U_2 \leq L_1 + R_1 + \|uv\| \tag{6}$$

from (3).

Adding inequalities (4–6) and using the inductive hypothesis (1) for L and R gives

$$2U_1 + U_2 \leq 2L_1 + L_2 + 2R_1 + R_2 + \|uv\| \leq |L| + |R| + \|uv\| = |U|. \quad \blacktriangleleft$$

We now have a multigraph $\widetilde{T}$ where every internal vertex has even degree. It follows that the edges of $\widetilde{T}$ can be partitioned into leaf-to-leaf paths, much like when creating an Eulerian tour in a graph where all vertices have even degree.

We still need to satisfy Property 3. Whenever two paths P_1 and P_2 violate this property, we repair this by swapping parts of the paths, without changing the number of remaining violating pairs, as follows: The paths P_1 and P_2 must have a common vertex, and thus also a common edge uv, because the maximum degree in T is 3. Orient P_1 and P_2 so that they use this edge in the direction uv, and cut them at v into $P_1 = Q_1 \cdot R_1$ and $P_2 = Q_2 \cdot R_2$. We now make a cross-over at v, forming the new paths $Q_1 \cdot R_2$ and $Q_2 \cdot R_1$. These new paths satisfy Property 3. To check that we did not create any new violations, we observe that, by Property 1, no other path can use the edge uv, because the capacity of 2 is already taken by P_1 and P_2. Thus, all other paths can either interact with Q_1 and Q_2, or with R_1 and R_2. Thus, swapping the parts of P_1 and P_2 in the other half of the tree T does not affect Property 3.

We have thus established Theorem 9. ◀

5.1 Finding a good fence in $\mathcal{A}$

The problem of finding a small cut in a planar graph $G = (V, E)$ that separates k different classes $T_1, \ldots, T_k \subset V$ of terminals was mentioned as a suggestion for future work by Dahlhaus et al. [8], but we have not found any subsequent work on that except for the case $k = 2$ [5]. We can, however, reduce the problem to the multiway cut problem in general graphs (also known as the multiterminal cut problem): For each class T_i, we add an "apex vertex" t_i which is connected to all vertices in T_i by edges of infinite weight. We then ask for the cut of minimum total weight that separates each pair t_i, t_j. Dahlhaus et al. gave a $(2 - 2/k)$-approximation algorithm for the problem. In our setup, the running time will be $O(kn^8 \log n)$. The approximation ratio was since then improved to $3/2 - 1/k$ by Călinescu et al. [6]. Finally, a randomized algorithm with approximation factor 1.3438 was given by Karger et al. [12], who also gave the best known bounds for various specific values of k. Together with Theorem 9, we obtain the following result.

▶ **Theorem 11.** *There is a randomized* $4/3 \cdot 1.3438$*-approximation algorithm and a deterministic* $(2 - \frac{4}{3k})$*-approximation algorithm for GEOMETRIC k-CUT, each of which runs in polynomial time.*

6 Concluding Remarks

We have initiated the study of the geometric multicut problem. As our NP-hardness reduction does not imply APX-hardness, an interesting open question is whether there exists a $(1+\varepsilon)$-approximation algorithm for any $\varepsilon > 0$.

There are other versions of the problem that could also be interesting to study. For example, apart from considering shortest paths in the plane, much attention has also been paid to minimum-link paths, i.e., paths connecting two points and consisting of a minimum number of line segments. The analogous problem in our setup is likewise interesting: Compute a simplest possible fence, i.e., one that is the union of as few line segments as possible. The fence can be required to be disjoint from the object interiors, or it can be allowed to pass through the objects, leading to two different problems.

References

1 Mikkel Abrahamsen, Anna Adamaszek, Karl Bringmann, Vincent Cohen-Addad, Mehran Mehr, Eva Rotenberg, Alan Roytman, and Mikkel Thorup. Fast fencing. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2018)*, pages 564–573, 2018. doi:10.1145/3188745.3188878.

2 Mikkel Abrahamsen, Panos Giannopoulos, Maarten Löffler, and Günter Rote. Geometric multicut, 2019. arXiv:1902.04045.

3 Oswin Aichholzer and Franz Aurenhammer. Straight skeletons for general polygonal figures in the plane. In Jin-Yi Cai and Chak Kuen Wong, editors, *Computing and Combinatorics (COCOON 1996)*, volume 1090 of *Lecture Notes in Computer Science*, pages 117–126. Springer-Verlag, 1996. doi:10.1007/3-540-61332-3_144.

4 Oswin Aichholzer and Franz Aurenhammer. Straight skeletons for general polygonal figures in the plane. In A. Samoilenko, editor, *Voronoi's Impact on Modern Sciences, Vol. II*, volume 21 of *Proceedings of the Institute of Mathematics of the National Academy of Sciences of Ukraine*, pages 7–21. Institute of Mathematics of the National Academy of Sciences of Ukraine, Kiew, 1998.

5 Glencora Borradaile, Philip N. Klein, Shay Mozes, Yahav Nussbaum, and Christian Wulff-Nilsen. Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. *SIAM Journal on Computing*, 46(4):1280–1303, 2017. doi:10.1137/15M1042929.

6 Gruia Călinescu, Howard Karloff, and Yuval Rabani. An improved approximation algorithm for Multiway Cut. *Journal of Computer and System Sciences*, 60(3):564–574, 2000. doi:10.1006/jcss.1999.1687.

7 Bernard Chazelle and Herbert Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *J. ACM*, 39(1):1–54, 1992. doi:10.1145/147508.147511.

8 Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23(4):864–894, 1994. doi:10.1137/S0097539792225297.

9 Paweł Gawrychowski and Adam Karczmarz. Improved bounds for shortest paths in dense distance graphs. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 61:1–61:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2018.61.

10 Edgar N. Gilbert and Henry O. Pollak. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16(1):1–29, 1968. doi:10.1137/0116001.

11 Dorothy M. Greig, Bruce T. Porteous, and Allan H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, 51(2):271–279, 1989. doi:10.1111/j.2517-6161.1989.tb01764.x.

12 David R. Karger, Philip Klein, Cliff Stein, Mikkel Thorup, and Neal E. Young. Rounding algorithms for a geometric embedding of minimum multiway cut. *Mathematics of Operations Research*, 29(3):436–461, 2004. doi:10.1287/moor.1030.0086.

13 Wolfgang Mulzer and Günter Rote. Minimum-weight triangulation is NP-hard. *Journal of the ACM*, 55:Article 11, 29 pp., May 2008. doi:10.1145/1346330.1346336.

Lower Bounds for Multiplication via Network Coding

Peyman Afshani
Computer Science Department, Aarhus University, Denmark
peyman@cs.au.dk

Casper Benjamin Freksen
Computer Science Department, Aarhus University, Denmark
cfreksen@cs.au.dk

Lior Kamma
Computer Science Department, Aarhus University, Denmark
lior.kamma@cs.au.dk

Kasper Green Larsen
Computer Science Department, Aarhus University, Denmark
larsen@cs.au.dk

Abstract

Multiplication is one of the most fundamental computational problems, yet its true complexity remains elusive. The best known upper bound, very recently proved by Harvey and van der Hoeven (2019), shows that two n-bit numbers can be multiplied via a boolean circuit of size $O(n \lg n)$. In this work, we prove that if a central conjecture in the area of network coding is true, then any constant degree boolean circuit for multiplication must have size $\Omega(n \lg n)$, thus almost completely settling the complexity of multiplication circuits. We additionally revisit classic conjectures in circuit complexity, due to Valiant, and show that the network coding conjecture also implies one of Valiant's conjectures.

2012 ACM Subject Classification Theory of computation; Theory of computation → Circuit complexity

Keywords and phrases Circuit Complexity, Circuit Lower Bounds, Multiplication, Network Coding, Fine-Grained Complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.10

Category Track A: Algorithms, Complexity and Games

Funding Peyman Afshani is supported by DFF (Det Frie Forskningsraad) of Danish Council for Independent Research under grant ID DFF-7014-00404. Casper Benjamin Freksen and Lior Kamma are supported by Villum Young Investigator Grant 13163. Kasper Green Larsen is supported by Villum Young Investigator Grant 13163 and an AUFF Starting Grant.

1 Introduction

Multiplication is one of the most fundamental computational problems and the simple "long multiplication" $O(n^2)$-time algorithm for multiplying two n-digit numbers is taught to elementary school pupils around the world. Despite its centrality, the true complexity of multiplication remains elusive. In 1960, Kolmogorov conjectured that the thousands of years old $O(n^2)$-time algorithm is optimal and he arranged a seminar at Moscow State University with the goal of proving this conjecture. However only a week into the seminar, the student Karatsuba came up with an $O(n^{\lg_2 3}) \approx O(n^{1.585})$ time algorithm [9]. The algorithm was presented at the next seminar meeting and the seminar was terminated. This sparked a sequence of improved algorithm such as the Toom-Cook algorithm [15, 4] and


the Schönhage-Strassen algorithm [14]. The Schönhage-Strassen algorithm, as well as the current fastest algorithm by Fürer [6], are both based on the Fast Fourier Transform (FFT). Fürer's algorithm can be shown to run in time $O(n \lg n \cdot 4^{\lg^* n})$ when multiplying two n-bit numbers [8]. It can even be implemented as a constant degree Boolean circuit of the same size. Here $\lg^* n$ is the very slowly growing iterated logarithm. A very recent and exciting result by Harvey and van der Hoeven [7] removes the extra $4^{\lg^* n}$ term, and presents an algorithm for integer multiplication that runs in time $O(n \lg n)$. But what is the true complexity of multiplying two n-bit numbers? Can it be done via e.g. a Boolean circuit of size $O(n)$ like addition? Or is multiplication strictly harder? Our main contribution is to show a connection between multiplication and a central conjecture by Li and Li [10] in the area of *network coding*. Our results show that if the conjecture by Li and Li [10] is true, then any constant degree Boolean circuit for computing the product of two n-bit numbers must have size $\Omega(n \lg n)$. This establishes a conditional lower bound for multiplication that is tight with respect to the upper bound presented by Harvey and van der Hoeven, and implies that multiplication is strictly harder than addition.

Before diving into the details of our results, we first give a brief introduction to network coding.

Network Coding. Network coding studies communication problems in graphs. Given a graph G with capacity constraints on the edges and k data streams, each with a designated source-sink pair of nodes (s_i, t_i) in G, what is the maximum rate at which data can be transmitted concurrently between the source-sink pairs? One solution is to just forward the data, which reduces the problem to a *multicommodity flow* problem. The central question in network coding is whether one can achieve a higher rate by using coding/bit tricks. This question is known to have a positive answer in directed graphs, where the rate increase may be as high as a factor $\Omega(|G|)$ (by sending XOR's of carefully chosen input bits), see e.g. [1]. However the question remains wide open for undirected graphs where there are no known examples for which network coding can do better than the multicommodity flow rate. A central conjecture in network coding, due to Li an Li [10], says that coding yields no advantage in undirected graphs.

▶ **Conjecture 1** (Undirected k-pairs Conjecture [10]). *The coding rate is equal to the Multi-commodity-Flow rate in undirected graphs.*

Despite the centrality of this conjecture, it has heretofore resisted all attempts at either proving or refuting it. Conjecture 1 has been used twice before for proving lower bounds for computational problems. Adler et al. [1] were the first to initiate this line of study. They presented conditional lower bounds for computing the transpose of a matrix via an *oblivious algorithm*. Here oblivious means that the memory access pattern is fixed and independent of the input. Very recently Farhadi et al. [5] showed how to remove the *obliviousness* assumption for external memory problems. Their main result was a tight lower bound for external memory integer sorting, conditioned on Conjecture 1 being true.

1.1 Our Results

Our main result is an exciting new connection between network coding and the complexity of multiplication. Formally, we prove the following theorem:

▶ **Theorem 1.** *Assuming Conjecture 1, every boolean circuit with arbitrary gates and bounded in and out degrees that computes the product of two numbers given as two n-bit strings has size $\Omega(n \lg n)$.*

In fact, we prove our $\Omega(n \lg n)$ lower bound for an even simpler problem than multiplication, namely the *shift problem*: In the shift problem, we are given an n-bit string x and an index $j \in [n]$. The goal is to construct a circuit that outputs the $2n$-bit string y whose ith bit equals the $(i - j + 1)$th bit of x for every $j \leq i \leq j + n - 1$. Here we think of the index j as being given in binary using $\lceil \lg n \rceil$ bits. We prove the following result:

▶ **Theorem 2.** *Assuming Conjecture 1, every boolean circuit with arbitrary gates and bounded in and out degrees that computes the shift problem has size* $\Omega(n \lg n)$.

Theorem 1 follows as a corollary of Theorem 2 by observing that shifting x by j positions is equivalent to multiplication by 2^j. Moreover, it is not hard to see that there is a linear sized circuit that has $\lceil \lg n \rceil$ input gates and n output gates, where on an index $j \in [n]$, it outputs the number 2^j in binary (i.e. a single 1-bit at position j).

We find it quite fascinating that even a simple instruction such as shifting requires circuits of size $\Omega(n \lg n)$, at least if we believe Conjecture 1.

Valiant's Depth Reduction and Circuit Complexity Lower Bounds. In addition to our main lower bound results for multiplication, we also demonstrate that the network coding conjecture sheds new light on some fundamental conjectures by Valiant. In a 1977 survey Valiant [16] outlined potentially plausible attacks on the problem of proving a lower bound for the size of any circuit that can compute a permutation or even shifts of a given input. The goal was to prove that achieving both $O(n)$ size and $O(\lg n)$ depth for such circuits is impossible. While most of his attacks were rebuffed due to existence of complex and highly connected graphs that only had $O(n)$ edges (superconcentrators), Valiant outlined one last potential approach that could still be fruitful. His main brilliant idea was to start with a circuit of some depth and by applying graph theoretical approaches reducing the depth of the circuit while eliminating only a small number of edges. The hope was that information theoretical approaches could finish the job once the depth of the circuit was very low and once the (graph theoretical) complexity of the circuit was peeled away.

More formally, Valiant showed that for every circuit C with n input and output gates, of size $O(n)$, depth $O(\lg n)$ and fan-in 2, and for every $\varepsilon > 0$, the function computed by C can be computed by a boolean circuit with arbitrary gates C' of depth 3 with n input and output gates and εn extra nodes. Moreover, the number of input gates directly connected to an output gate is bounded. That is, if we denote the set of input and output gates by X and Y respectively, then for every $y \in Y$, there are at most $O(n^\varepsilon)$ wires connecting y and X.

In turn, this reduction shows that it is enough to prove lower bounds on such depth 3 circuits. Almost 20 years later and based on these ideas, Valiant [17] put forward several conjectures that if resolved could open the way for proving circuit complexity lower bounds. Loosely speaking, Valiant conjectured that if $\varepsilon \leq 1/2$ then such depth 3 circuits cannot compute cyclic-shift permutation. Before discussing Valiant's conjectures more formally, we first state our second main result, which essentially shows that Conjecture 1 implies one of Valiant's conjectures, albeit with a smaller (but still constant) bound on ε.

▶ **Theorem 3.** *Let C be a depth 3 circuit that computes multiplication such that the following holds.*

1. *The number of gates in the second layer of C is at most εn for $\varepsilon \leq 1/300$; and*
2. *for every output gate y of C, the number of input gates directly connected to y is at most c.*

Then assuming Conjecture 1, $c = \Omega\left(\frac{\lg n}{\lg \lg n}\right)$.

As with Theorem 1, we prove Theorem 3 on an even restricted set of circuits, namely circuits that compute the shift function. We now turn to give a formal description of Valiant's Conjectures, and demonstrate how Theorem 3 brings us closer to settling them.

Valiant's Conjectures. Let Γ be a bipartite graph on two independent sets X and Y such that $X = \{x_1, \ldots, x_n\}$ denotes a set of inputs and $Y = \{y_1, \ldots, y_n\}$ denotes a set of outputs. Furthermore assume, let $f_1, \ldots f_{\varepsilon n}$ be εn extra nodes and connect them by edges to all the nodes in Γ. Denoting the resulting graph by G consider all possible boolean circuits with arbitrary gates whose underlying topology is G. We say such a circuit computes a permutation $\pi \colon Y \to X$ if for every assignment $x_1, \ldots, x_n \in \{0,1\}^n$ to the input gates, after the evaluation of the circuit y_j is assigned $\pi(y_j)$ for every $j \in [n]$. Valiant conjectured that this should be impossible if ε is too small or if Γ has too few edges. In particular, he proposed the following.

▶ **Conjecture 2.** *If Γ has maximum degree at most 3 and if $\varepsilon \leq 1/2$, then there exists a permutation π such that no circuit that has G as its underlying topology can compute the permutation π. Moreover, there exists such π that is a cyclic shift.*

Theorem 3 shows that conditioned on Conjecture 1, if $\varepsilon \leq 1/300$ then Valiant's first conjecture holds. We note that our proof for Theorem 3 continues to hold even if the gates' boolean functions are fixed after the shift offset is given. That is, if only the topology is fixed in advance. This coincides exactly with the formulation of Valiant's conjecture. Valiant further conjectured the following.

▶ **Conjecture 3.** *If Γ has at most $n^{2-\delta}$ edges for some constant $\delta > 0$, and if $\varepsilon \leq 1/2$, then there exists a permutation π such that no circuit that has G as its underlying topology can compute the permutation π. Moreover, there exists such π that is a cyclic shift.*

1.2 Related Work

Lower Bounds for Multiplication. There are a number of previous lower bounds for multiplication in various restricted models of computation. Clifford and Jalsenius [3] considered a streaming variant of multiplication, where one number is fixed and the other is revealed one digit at a time. They require that a digit of the output is reported before the next digit of the input is revealed. In this streaming setting, they prove an $\Omega((\delta/w)n \lg n)$ lower bound, where δ is the number of bits in a digit and w is the word size. For $\delta = 1$ and $w = O(1)$, this is $\Omega(n \lg n)$. Ponzio [12] considered multiplication via read-once branching programs, i.e. programs that have bounded working memory and may only read each input bit exactly once. He proved that any read-once branching program for computing the middle bit of the product of two n-bit numbers, must use $\Omega(\sqrt{n})$ bits of working memory. Finally, we also mention the work of Morgenstern [11] who proved lower bounds for computing the related FFT. Morgenstern proved an $\Omega(n \lg n)$ lower bound for computing the *unnormalied* FFT via an arithmetic circuit when all constants used in the circuit are bounded. Unfortunately this doesn't say anything about the complexity of multiplying two n-bit numbers.

Valiant's Conjectures. Despite their importance, Valiant's conjectures are still mostly open. One interesting development by Riis [13], shows that Conjecture 3 as stated is incorrect. Riis proved that all cyclic shifts are realizable for $\varepsilon = \frac{1}{2} - \frac{1}{2n^{1-\delta}}$ where $n^{1+\delta}$ is the total number of edges of Γ. Riis further conjectured that replacing the bound on ε by a slightly stricter bound should result in a correct conjecture. Specifically, Riis suggest bounding $\varepsilon = \Theta\left(\frac{1}{\lg \lg n}\right)$.

2 Preliminaries

We now give a formal definition of Boolean circuits with arbitrary gates, followed by definitions of the k-pairs communication problem, the multicommodity flow problem. In the two latter problems we reuse some of the definitions used by Farhadi et al. [5], which have been simplified a bit compared to the more general definition by Adler et al. [1]. In particular, we have forced communication networks to be directed acyclic graphs. This is sufficient to prove our lower bounds and simplifies the definitions considerably.

Boolean Circuits with Arbitrary Gates. A *Boolean Circuit with Arbitrary Gates* with n source or input nodes and m target or output nodes is a directed acyclic graph C with n nodes of in-degree 0, which are called *input gates*, and are labeled with input variables $X = \{x_i\}_{i\in[n]}$ and m nodes out-degree 0, which are called *output gates* and are labeled with output variables $Y = \{y_i\}_{i\in[m]}$. All other nodes are simply called *gates*. For every gate u of in-degree $k \geq 1$, u is labeled with an arbitrary function $f_u : \{0,1\}^k \to \{0,1\}$. The circuit is also equipped with a topological ordering $v_1, \ldots, v_t$ of C, in which $v_i = x_i$ for $i \in [n]$ and $v_{t-i+1} = y_{m-i+1}$ for all $i \in [m]$. The *depth* of a circuit C is the length of the longest path between an input and an output node in C. An *evaluation* of a circuit on an n bit input $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ is conducted as follows. For every $i \in [n]$, assign x_j to v_j. For every $j \geq n+1$, assign to v_j the value $f_{v_j}(u_1, \ldots, u_k)$, where $u_1, \ldots, u_k$ are the nodes of C with edges going into v_j in the order induced by the topological ordering. The *output* of C on an n bit input $x = (x_1, \ldots, x_n)$, denoted $C(x_1, \ldots, x_n)$ is the value assigned to $(y_1, \ldots, y_m)$ in the evaluation. We say a circuit computes a function $f : \{0,1\}^n \to \{0,1\}^m$ if for every $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, $f(x_1, \ldots, x_n) = C(x_1, \ldots, x_n)$.

For every $j \in [t]$ and $b \in \{0,1\}$, we *hardwire* b for v_j in C by removing v_j and all adjacent edges from C, and replacing v_j for b in the evaluation of f_{v_i} for every $i > j$ such that $v_j v_i$ is an edge in C.

k-Pairs Communication Problem. The input to the k-pairs communication problem is a directed acyclic graph $G = (V, E)$ where each edge $e \in E$ has a capacity $c(e) \in \mathbb{R}^+$. There are k sources $s_1, \ldots, s_k \in V$ and k sinks $t_1, \ldots, t_k \in V$.

Each source s_i receives a message A_i from a predefined set of messages $A(i)$. It will be convenient to think of this message as arriving on an in-edge. Hence we add an extra node S_i for each source, which has a single out-edge to s_i. The edge has infinite capacity.

A network coding solution specifies for each edge $e \in E$ an alphabet $\Gamma(e)$ representing the set of possible messages that can be sent along the edge. For a node $v \in V$, define $\mathrm{In}(u)$ as the set of in-edges at u. A network coding solution also specifies, for each edge $e = (u, v) \in E$, a function $f_e : \prod_{e' \in \mathrm{In}(u)} \Gamma(e') \to \Gamma(e)$ which determines the message to be sent along the edge e as a function of all incoming messages at node u. Finally, a network coding solution specifies for each sink t_i a decoding function $\sigma_i : \prod_{e \in \mathrm{In}(t_i)} \Gamma(e) \to A(i)$. The network coding solution is correct if, for all inputs $A_1, \ldots, A_k \in \prod_i A(i)$, it holds that σ_i applied to the incoming messages at t_i equals A_i, i.e. each source must receive the intended message.

In an execution of a network coding solution, each of the extra nodes S_i starts by transmitting the message A_i to s_i along the edge (S_i, s_i). Then, whenever a node u has received a message a_e along all incoming edges $e = (v, u)$, it evaluates $f_{e'}(\prod_{e \in \mathrm{In}(u)} a_e)$ on all out-edges and forwards the message along the edge e'.

We define the *rate* of a network coding solution as follows: Let each source receive a uniform random and independently chosen message A_i from $A(i)$. For each edge e, let A_e

denote the random variable giving the message sent on the edge e when executing the network coding solution with the given inputs. Let $H(\cdot)$ denote the binary Shannon entropy. The network coding solution achieves rate r if:

- $H(A_i) \geq r$ for all i.
- For each edge $e \in E$, we have $H(A_e) \leq c(e)$.

The intuition is that the rate is r, if the solution can handle sending a message of entropy r bits between every source-sink pair.

Multicommodity Flow. A multicommodity flow problem in an undirected graph $G = (V, E)$ is specified by a set of k source-sink pairs (s_i, t_i) of nodes in G. We say that s_i is the source of commodity i and t_i is the sink of commodity i. Each edge $e \in E$ has an associated capacity $c(e) \in \mathbb{R}^+$. A (fractional) solution to the multicommodity flow problem specifies for each pair of nodes (u, v) and commodity i, a flow $f^i(u, v) \in [0, 1]$. Intuitively $f^i(u, v)$ specifies how much of commodity i that is to be sent from u to v. The flow satisfies *flow conservation*, meaning that:

- For all nodes u that is not a source or sink, we have $\sum_{w \in V} f^i(u, w) - \sum_{w \in V} f^i(w, u) = 0$.
- For all sources s_i, we have $\sum_{w \in V} f^i(s_i, w) - \sum_{w \in V} f^i(w, s_i) = 1$.
- For all sinks we have $\sum_{w \in V} f^i(w, t_i) - \sum_{w \in V} f^i(t_i, w) = 1$.

The flow also satisfies that for any pair of nodes (u, v) and commodity i, there is only flow in one direction, i.e. either $f^i(u, v) = 0$ or $f^i(v, u) = 0$. Furthermore, if (u, v) is not an edge in E, then $f^i(u, v) = f^i(v, u) = 0$. A solution to the multicommodity flow problem achieves a rate of r if:

- For all edges $e = (u, v) \in E$, we have $r \cdot \sum_i (f^i(u, v) + f^i(v, u)) \leq c(e)$.

Intuitively, the rate is r if we can handle a demand of r for every commodity.

The Undirected k-Pairs Conjecture. Conjecture 1 implies the following for our setting: Given an input to the k-pairs communication problem, specified by a directed acyclic graph G with edge capacities and a set of k source-sink pairs, let r be the best achievable network coding rate for G. Similarly, let G' denote the undirected graph resulting from making each directed edge in G undirected (and keeping the capacities and source-sink pairs). Let r' be the best achievable flow rate in G'. Conjecture 1 implies that $r \leq r'$.

Having defined coding rate and flow rate formally, we also mention that a result of Braverman et al. [2] implies that if there exists a graph G where the network coding rate r, and the flow rate r' in the corresponding undirected graph G', satisfies $r \geq (1 + \varepsilon)r'$ for a constant $\varepsilon > 0$, then there exists an infinite family of graphs $\{G^*\}$ for which the corresponding gap is at least $(\lg |G^*|)^c$ for a constant $c > 0$. So far, all evidence suggest that no such gap exists, as formalized in Conjecture 1.

3 Key Tools and Techniques

The main idea in the heart of both proofs is the simple fact that in a graph with t vertices and maximum degree at most c, most node pairs lie far away from one another. Specifically, for every node u in G, at least $t - \sqrt{t}$ nodes have distance $\geq \frac{1}{2} \log_c t$ from u. While this key observation is almost enough to prove Theorem 2, the proof of Theorem 3 requires a much more subtle approach, as there is no bound on the maximum degree in the circuits in question. The only bound we have is on the number of wires going directly between from input gates into output gates. Specifically, every two nodes in the underlying undirected graph are at distance ≤ 3 (see Figure 1).

In order to overcome this obstacle, we present a construction of a communication network based on the circuit C that essentially eliminates the middle layer in the depth-3 circuit C, thus leaving a bipartite graph with bounded maximum degree. To this end, we observe that since the size of the middle layer is bounded by εn, then there exists a large set $\mathcal{F}$ of inputs in $\{0,1\}^n$ such that on all inputs from $\mathcal{F}$, the gates $f_1, \ldots, f_{\varepsilon n}$ attain the same values. By hardwiring these values to the circuit, we can evaluate the circuit for all inputs in $\mathcal{F}$ on a depth-2 circuit Γ obtained from C by removing $f_1, \ldots, f_{\varepsilon n}$. We next turn to construct the communication network. Employing ideas recently presented by Farhadi et al. [5], we "wrap" the depth-2 circuit by adding source and target nodes. In order to cope with inputs that do not belong to $\mathcal{F}$, we add a designated *supervisor* node u (see figure 2). Loosely speaking, the source nodes transmit their input to u, and u sends back the information needed to "edit" the input string x and construct an input string $x' \in \mathcal{F}$, which is then transferred to the circuit Γ as blackbox.

The Correction Game. In order to bound the edge capacities of the network G in a way that the supervisor node can transmit enough information to achieve a high communication rate, but then again not allow to much flow to go through the supervisor when considering G as a multicommodity flow instance, Farhadi et al. [5] defined a game between a set of m players and a supervisor, where given a fixed set $\mathcal{F} \subseteq \{0,1\}^n$ and a random string $\beta \in \{0,1\}^n$ given as a concatenation of m strings $\beta_1, \ldots, \beta_m$ of length n/m each, the goal is to "correct" β and produce a string $\chi \in \{0,1\}^n$ such that $\beta \oplus \chi \in \mathcal{F}$. The caveat is that the only communication allowed is between the players and the supervisor. That is, no communication, and thus no cooperation, is allowed between the m players. Formally, the game is defined as follows.

▶ **Definition 4.** *Let $\mathcal{F} \subseteq \{0,1\}^n$. The* $\mathcal{F}$-correction game *with $m+1$ players is defined as follows. The game is played by m ordinary players $p_1, \ldots, p_m$ and one designated* supervisor *player u. The supervisor u receives m strings $\beta_1, \ldots, \beta_m \in \{0,1\}^{n/m}$ chosen independently at random. For every $\ell \in [m]$, u then sends p_ℓ a message R_ℓ. Given R_ℓ, the player p_ℓ produces a string $\chi_\ell \in \{0,1\}^{n/m}$ such that $(\beta_1 \oplus \chi_1) \circ (\beta_2 \oplus \chi_2) \circ (\beta_m \oplus \chi_m) \in \mathcal{F}$.*

Farhadi et al. additionally present a protocol for the $\mathcal{F}$-correction game in which the supervisor player sends prefix-free messages to the m players, and moreover, they give a bound on the amount of communication needed as a function of the number of players and the size of $\mathcal{F}$.

▶ **Lemma 5** ([5]). *If $|\mathcal{F}| \geq 2^{(1-\varepsilon)n}$, then there exists a protocol for the $\mathcal{F}$-correction game with $m+1$ players such that the messages $\{R_\ell\}_{\ell \in [m]}$ are prefix-free and*

$$\sum_{\ell \in [m]} \mathbb{E}[|R_\ell|] \leq 3m + 2m \lg \left(\sqrt{\frac{\varepsilon}{2}} \cdot \frac{n}{m} + 1 \right) + \sqrt{\frac{\varepsilon}{8}} \cdot n \lg \frac{2}{\varepsilon} ,$$

4 A Lower Bound for Boolean Circuits Computing Multiplication

In this section we show that conditioned on Conjecture 1, every bounded degree circuit computing multiplication must have size at least $\Omega(n \lg n)$, thus proving Theorems 1 and 2. In fact, we will prove something slightly stronger. Define the shift function $s : \{0,1\}^n \times [n] \to \{0,1\}^{2n}$ as follows. For every $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ and $\ell \in [n]$, $s(x, \ell) = (y_1, \ldots, y_{2n})$ where $y_j = x_{j-\ell+1}$ if $\ell \leq j \leq \ell + n - 1$ and $y_j = 0$ otherwise. We will show that every circuit with bounded in and out degrees that computes the shift function on n-bit numbers has size

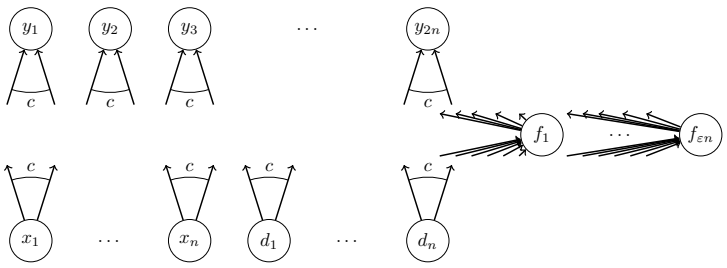


Figure 1 The depth 3 circuit C.

$\Omega(n \lg n)$. Clearly, a circuit that can compute the product of two n-bit numbers can also compute the shift function. Let c denote the maximum in and out degree in C, and let $j \in [n]$. Then in the undirected graph induced by C, there are at most $\sqrt{n}$ nodes whose distance from x_j is at most $\frac{1}{2}\log_{2c} n$. Therefore among $y_j, \ldots, y_{j+n-1}$, at least $n - \sqrt{n} - 1 \geq n - 2\sqrt{n}$ are at distance at least $\frac{1}{2}\log_{2c} n$. In other words, $\Pr_{\ell \in [n]}[d_{\bar{C}}(x_j, y_{j+\ell-1}) \geq \frac{1}{2}\log_{2c} n] \geq 1 - \frac{2}{\sqrt{n}}$, where $\bar{C}$ denotes the undirected graph induced by C (by removing edge directions). Therefore there exists a shift $\ell_0 \in [n]$ such that $|\{j \in [n] : d_{\bar{C}}(x_j, y_{j+\ell_0-1}) \geq \frac{1}{2}\log_{2c} n\}| \geq n - 2\sqrt{n} \geq n/2$.

Fixing ℓ_0, consider the following communication problem. For each $j \in [n]$, let $s_j = x_j$ and $t_j = y_{j+\ell_0-1}$ be a source-sink pair, and let $A_j = \{0,1\}$ be the set of possible messages. The circuit C equipped with 1-uniform edge capacities is a network coding solution to this problem with rate $r \geq 1$. By the undirected n-pairs conjecture, there is a multicommodity flow in $\bar{C}$ that transfers one unit of flow from each source to its corresponding sink. For every j, let $f^j : E \to [0,1]$ be the flow associated with commodity j. Then

$$|E| = \sum_{e \in E} c_e \geq \sum_{e \in E} \sum_{j \in [n]} f^j(e) \geq \Omega(n \log_c n) .$$

5 A Lower Bound for Depth 3 Boolean Circuits Computing Multiplication

Let C be a depth 3 circuit that computes multiplication such that the number of gates in the second layer of C is at most εn for some small $\varepsilon \in (0,1)$ and for every $u \in Y$, $deg_{\bar{C}[X \cup Y]}(u) \leq c$, where once again $\bar{C}$ denotes the undirected graph induced by C, and $\bar{C}[X \cup Y]$ is the subgraph of $\bar{C}$ induced by $X \cup Y$. By slightly increasing c and ε (by a small constant factor) and without loss of generality, we can assume that this applies for all $u \in X$ as well. To see this, note that by the assumption on Y, the total number of edges in $\bar{C}[X \cup Y]$ is at most cn. Therefore there are at most $n/10$ nodes in X whose degree in $\bar{C}[X \cup Y]$ is more than $10c$. For each such node $u \in X$, we can replace all edges adjacent to u in $\bar{C}[X \cup Y]$ by an extra second-layer node f_u connected to u and its neighbors in $\bar{C}[X \cup Y]$. Denote the input and output gates of C by $X = \{x_1, \ldots, x_n, \hat{x}_1, \ldots, \hat{x}_n\}$ and $Y = \{y_1, \ldots, y_{2n}\}$ respectively, and denote the set of the middle-layer gates by $F = \{f_1, \ldots, f_{\varepsilon n}\}$ (see Figure 1).

As before, we focus on computing the shift function, thus limiting the input to $(\hat{x}_1, \ldots, \hat{x}_n)$ to have exactly one 1-entry. We next partition $(x_1, \ldots, x_n)$ into consecutive blocks of size $k = 20$ bits each. For every $\ell \in [n/k]$ let $B_\ell = \{k(\ell-1)+1, \ldots, k\ell\}$ be the set of indices belonging to the ℓth block.

▶ **Definition 6.** *For every $\alpha \in [n]$ and $\ell \in [n/k]$, we say B_ℓ is* far from all targets *(with respect to α) if for all sources in the block are at distance at least $\frac{1}{2}\log_{2c} n$ from all respective destinations in $\bar{C}[X \cup Y]$. That is for every $u, v \in B_\ell$, $d_{\bar{C}[X \cup Y]}(x_u, y_{v+\alpha-1}) \geq \frac{1}{2}\log_{2c} n$.*

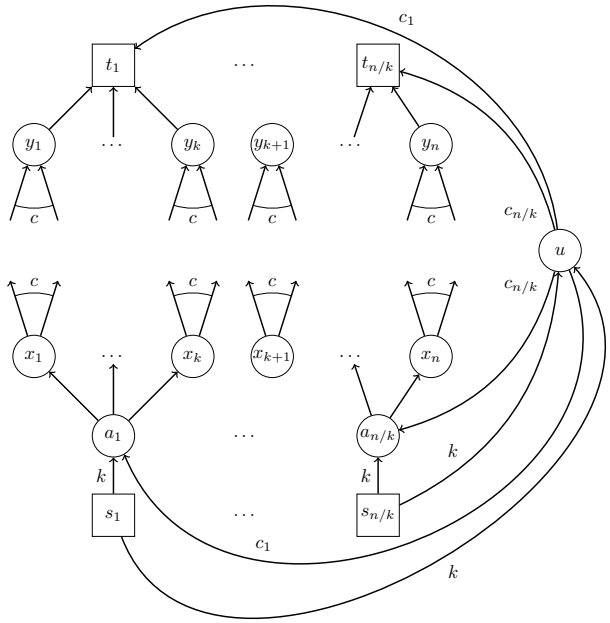


Figure 2 Given the 2-layer circuit Γ spanned by $x_1, \ldots, x_n, y_1, \ldots, y_n$, we construct the communication network graph G.

Let $\alpha \in_R [n]$. By the constraint on the degrees, for every $j \in [n]$, there are at most $\sqrt{n}$ nodes whose distance from x_j is at most $\frac{1}{2}\log_{2c} n$ in $\bar{C}[X \cup Y]$. Therefore for every $\ell \in [n/k]$,

$$\Pr_{\alpha \in_R [n]}[B_\ell \text{ is far from all targets}] \geq 1 - \frac{k^2}{\sqrt{n}}.$$

By averaging we get that for large enough n there is some $\alpha_0 \in [n]$ such that there are at least $\frac{n}{k} - k\sqrt{n} \geq \frac{9n}{10k}$ blocks which are far from all targets. Without loss of generality, we may assume for ease of notation that $\alpha_0 = 1$. By hardwiring 1 for α_0 into the circuit C, the circuit now simply transfers $(x_1, \ldots, x_n)$ to $(y_1, \ldots, y_n)$.

Reduction to Network Coding. Let $x = (x_1, \ldots, x_n)$ and $i \in [\varepsilon n]$. By slightly abusing notation, we denote the value of the gate f_i when evaluating the circuit by $f_i(x_1, \ldots, x_n)$. By averaging, there exist a string $(\hat{f}_1, \ldots, \hat{f}_{\varepsilon n})$ and a set $\mathcal{F} \subseteq \{0,1\}^n$ such that $|\mathcal{F}| \geq 2^{(1-\varepsilon)n}$ and such that for every $x = (x_1, \ldots, x_n) \in \mathcal{F}$ and $i \in [\varepsilon n]$, $f_i(x_1, \ldots, x_n) = \hat{f}_i$. By hardwiring $(\hat{f}_1, \ldots, \hat{f}_{\varepsilon n})$ for $(f_1, \ldots, f_n)$ into the circuit C, we get a new circuit denoted Γ that contains only the input and output gates of C, and transfers $(x_1, \ldots, x_n)$ to $(y_1, \ldots, y_n)$ for every $(x_1, \ldots, x_n) \in \mathcal{F}$. Moreover, the set of edges between X and Y in Γ is equal to the set of edges between X and Y in C.

Next, we construct a communication network G by adding some nodes and edges to Γ, as demonstrated also in Figure 2. We add a new set of nodes $\{s_j, a_j, t_j\}_{j=1}^{n/k} \cup \{u\}$. For every $\ell \in [n/k]$, add edges $s_\ell a_\ell$ and $s_\ell u$ of capacity k and edges ua_ℓ and ut_ℓ of capacity $c_\ell = \mathbb{E}[|R_\ell|]$,

where R_ℓ is the message sent to player p_ℓ by the supervisor player in the $\mathcal{F}$-correction game protocol for $n/k+1$ players guaranteed in Lemma 5. In addition, for every $\ell \in [n/k]$ and every $j \in B_\ell$ add edges $a_\ell x_j$ and $y_j t_\ell$ of capacity 1. All edges of Γ are assigned capacity of 1.

Transmitting Data. In what follows, we will lower bound the communication rate of the newly constructed network G.

▶ **Lemma 7.** *There exists a network coding solution on G that achieves rate k.*

To this end, let $A_1, \ldots, A_{n/k} \in \{0,1\}^k$ be independent uniform random variables. We next give a protocol by which the sources $s_1, \ldots, s_{n/k}$ transmit $A_1, \ldots, A_{n/k}$ to the targets $t_1, \ldots, t_{n/k}$. The protocol employs as an intermediate step the correction game protocol guaranteed by Lemma 5.

1. For every $\ell \in [n/k]$, s_ℓ sends A_ℓ to a_ℓ over the edge $s_\ell a_\ell$ and to u over the edge $s_\ell u$.
2. Employing the $\mathcal{F}$-correction game protocol with $n/k+1$ players, for every $\ell \in [n/k]$, u sends a message R_ℓ to a_ℓ over the edge ua_ℓ and to t_ℓ over the edge ut_ℓ. Following the correction game protocol, for every ℓ, given R_ℓ, a_ℓ and t_ℓ produce a string χ_ℓ satisfying that $(A_1 \oplus \chi_1) \circ \ldots \circ (A_{n/k} \oplus \chi_{n/k}) \in \mathcal{F}$.
3. For every $\ell \in [n/k]$ and every $i \in [k]$, a_ℓ transmits the ith bit of $A_\ell \oplus \chi_\ell$ to the ith gate in the ℓth block, namely $x_{(\ell-1)k+i}$. Note that $(x_1, \ldots, x_n) = (A_1 \oplus \chi_1) \circ \ldots \circ (A_{n/k} \oplus \chi_{n/k}) \in \mathcal{F}$.
4. Next, the communication network employs the circuit Γ and transmits $(x_1, \ldots, x_n)$ to $(y_1, \ldots, y_n)$. For every $\ell \in [n/k]$ and every $i \in B_\ell$, y_i transmits x_i to t_ℓ.
5. Finally, for every $\ell \in [n/k]$, t_ℓ now holds both $A_\ell \oplus \chi_\ell$ and χ_ℓ. Therefore t_ℓ can recover A_ℓ.

By invoking the protocol described above, every one of the n/k sources sends k bits to the corresponding target. For every edge $e \in G$, let A_e denote the random variable giving the message sent on the edge e when executing the protocol.

▷ Claim 8. For every $e \in G$, $H(A_e) \leq c_e$.

Proof. First note that for every $\ell \in [n/k]$, every edge e leaving s_ℓ has capacity k and transmits A_ℓ. Therefore $H(A_\ell) = k \leq c_e$. Every edge e that is not leaving any source nor u has capacity 1 and transmits exactly one bit (not necessarily uniformly random) of information. Therefore $c_e = 1 \geq H(A_e)$. Finally, let e be an edge leaving u. Then there exists some $\ell \in [n/k]$ such that $e = ua_\ell$ or $e = ut_\ell$. In both cases the message transmitted on e is R_ℓ and the capacity c_e of e satisfies $c_e = c_\ell = \mathbb{E}[|R_\ell|] \geq H(R_\ell)$, where the last inequality follows from Shannon's Source Coding theorem, as all messages are prefix-free. ◁

We can therefore conclude that the network G achieves rate $\geq k$, and the proof of Lemma 7 is complete.

Deriving the Lower Bound. By Conjecture 1, the underlying undirected graph $\bar{G}$ achieves a multicommodity-flow rate $\geq k$. Therefore there exists a multicommodity flow $\{f^\ell\}_{\ell \in [n/k]} \subseteq [0,1]^{E(\bar{G})}$ that achieves rate k. We first observe that at most a constant fraction of the flow can go through the supervisor node u. To see this, we note that as $|\mathcal{F}| \geq 2^{(1-\varepsilon)n}$, then by Lemma 5, for small enough (constant) ε, the expected total information sent by the supervisor in the $\mathcal{F}$-correction game with n/k players is at most

$$\frac{3n}{k} + \frac{2n}{k} \lg\left(k\sqrt{\frac{\varepsilon}{2}} + 1\right) + \sqrt{\frac{\varepsilon}{8}} \cdot n \lg \frac{2}{\varepsilon} \leq \frac{5n}{k} \tag{1}$$

Therefore by the definition of the capacities $\{c_\ell\}_{\ell\in[n/k]}$ we get that

$$\sum_{\ell\in[n/k]} c_{ua_\ell} = \sum_{\ell\in[n/k]} c_{ut_\ell} = \sum_{\ell\in[n/k]} c_\ell \leq \frac{5n}{k} \tag{2}$$

Since $\{f^\ell\}_{\ell\in[n/k]}$ achieves rate k we conclude that for every $v \in V(\bar{G})$ adjacent to u we have $k\sum_{\ell\in[n/k]}(f^\ell(u,v)+f^\ell(v,u)) \leq c_{uv}$. Therefore

$$\begin{aligned} k\cdot \sum_{v\in V(\bar{G}):uv\in E(\bar{G})} \sum_{\ell\in[n/k]} (f^\ell(u,v)+f^\ell(v,u)) &\leq \sum_{v\in V(\bar{G}):uv\in E(\bar{G})} c_{uv} \\ &= \sum_{\ell\in[n/k]} c_{us_\ell} + \sum_{\ell\in[n/k]} (c_{ua_\ell}+c_{ut_\ell}) \leq n + \frac{10n}{k}\,. \end{aligned}$$

Rearranging we get that

$$\sum_{v\in V(\bar{G}):uv\in E(\bar{G})} \sum_{\ell\in[n/k]} (f^\ell(u,v)+f^\ell(v,u)) \leq \frac{n}{k} + \frac{10n}{k^2} \leq 1.5\frac{n}{k}\,. \tag{3}$$

By the flow-conservation constraint, we know that therefore the total amount of flow that can go through u is $\leq 0.75\frac{n}{k}$. By averaging, at least a $1/6$ fraction of the sources send at least $1/10$ units of flow through $\bar{G}-u$. By the choice of α_0, in $\bar{G}-u$, at least a $1/15$ of the sources are at least $\frac{1}{2}\log_{2c}(n)$ away from their targets. Without loss of generality, assume these are the first $\frac{n}{15k}$ sources. We conclude that

$$\begin{aligned} cn \geq |E[X\cup Y]| = \sum_{e\in E[X\cup Y]} c_e &\geq k\cdot \sum_{e=vw\in E[X\cup Y]} \sum_{\ell\in[n/k]} f^\ell(v,w)+f^\ell(w,v) \\ &\geq k\cdot \sum_{\ell\in[n/15k]} \sum_{e=vw\in E[X\cup Y]} f^\ell(v,w)+f^\ell(w,v) \geq \frac{n}{30}\log_{2c}(n)\,, \end{aligned} \tag{4}$$

and therefore $c \geq \Omega\left(\frac{\lg n}{\lg\lg n}\right)$, and the proof of Theorem 3 is now complete.

5.1 Remarks and Extensions

For sake of fluency, some minor remarks and extensions were intentionally left out of the text, and will be discussed now.

Circuits with Bounded Average Degree. Our results still hold if we relax the second requirement of Theorem 3 and require instead that the number of edges in $\bar{C}[X\cup Y]$ is at most cn. That is, the average degree in $\bar{C}[X\cup Y]$ is at most c. To see this, note that under this assumption, there are at most $0.001n$ gates in $X\cup Y$ whose degree in $\bar{C}[X\cup Y]$ is larger than $1000c$. For each such gate v, add a new node f in the middle layer, and connect v and all the neighbors of v in $\bar{C}[X\cup Y]$ to f. Then delete all the edges adjacent to v in $\bar{C}[X\cup Y]$. The number of nodes added to the middle layer is at most $0.001n$, and the degree of all nodes in $\bar{C}[X\cup Y]$ is now bounded by $1000c$. The rest of our proof continues as before.

Shifts vs. Cyclic Shifts. In order to prove lower bounds for circuits computing multiplication, our results are stated in terms of shifts (which are a special case of products, as mentioned). This is in contrast to Valiant's conjectures, which are stated in terms of cyclic shifts. However, we draw the readers attention to the fact that our proofs work for cyclic shifts as well. The exact same arguments apply, and the proofs remain unchanged.

References

1 Micah Adler, Nicholas J. A. Harvey, Kamal Jain, Robert Kleinberg, and April Rasala Lehman. On the Capacity of Information Networks. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, pages 241–250. Society for Industrial and Applied Mathematics, 2006. URL: http://dl.acm.org/citation.cfm?id=1109557.1109585.

2 Mark Braverman, Sumegha Garg, and Ariel Schvartzman. Coding in Undirected Graphs Is Either Very Helpful or Not Helpful at All. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, pages 18:1–18:18, 2017.

3 Raphaël Clifford and Markus Jalsenius. Lower Bounds for Online Integer Multiplication and Convolution in the Cell-Probe Model. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*, pages 593–604, 2011.

4 Stephen Arthur Cook. *On the minimum computation time of functions.* PhD thesis, Harvard University, 1966.

5 Alireza Farhadi, MohammadTaghi Hajiaghayi, Kasper Green Larsen, and Elaine Shi. Lower Bounds for External Memory Integer Sorting via Network Coding. In *Proceedings of the 52st Symposium on Theory of Computing, STOC 2019*, 2019. To appear.

6 Martin Fürer. Faster Integer Multiplication. *SIAM Journal on Computing*, 39(3):979–1005, 2009. doi:10.1137/070711761.

7 D. Harvey and J. van der Hoeven. Integer multiplication in time $O(n \log n)$. Technical report, HAL, 2019. http://hal.archives-ouvertes.fr/hal-02070778.

8 David Harvey and Joris van der Hoeven. Faster integer multiplication using short lattice vectors. *CoRR*, 2018. arXiv:1802.07932.

9 Anatoly Alekseevich Karatsuba and Yurii Petrovich Ofman. Multiplication of Many-Digital Numbers by Automatic Computers. *Proceedings of the USSR Academy of Sciences*, 145:293–294, 1962.

10 Zongpeng Li and Baochun Li. Network Coding: The Case of Multiple Unicast Sessions. In *Proceedings of the 42nd Annual Allerton Conference on Communication, Control, and Computing*, 2004.

11 Jacques Morgenstern. Note on a Lower Bound on the Linear Complexity of the Fast Fourier Transform. *Journal of the ACM*, 20(2):305–306, 1973. doi:10.1145/321752.321761.

12 Stephen Ponzio. A Lower Bound for Integer Multiplication with Read-Once Branching Programs. *SIAM J. Comput.*, 28(3):798–815, 1998.

13 Søren Riis. Information flows, graphs and their guessing numbers. *The Electronic Journal of Combinatorics*, 14(1), 2007.

14 Arnold Schönhage and Volker Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7(3):281–292, September 1971. doi:10.1007/BF02242355.

15 Andrei Leonovich Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. *Proceedings of the USSR Academy of Sciences*, 150(3):496–498, 1963.

16 Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *Mathematical Foundations of Computer Science 1977*, pages 162–176, 1977.

17 Leslie G. Valiant. Why is Boolean Complexity Theory Difficult? In *Proceedings of the London Mathematical Society Symposium on Boolean Function Complexity*, pages 84–94, 1992.

Path Contraction Faster Than 2^n

Akanksha Agrawal
Ben-Gurion University of the Negev, Beersheba, Israel
agrawal@post.bgu.ac.il

Fedor V. Fomin
University of Bergen, Bergen, Norway
fomin@ii.uib.no

Daniel Lokshtanov
University of California Santa Barbara, Santa Barbara, California
daniello@ucsb.edu

Saket Saurabh
Institute of Mathematical Sciences, HBNI and UMI ReLaX Chennai, India
University of Bergen, Bergen, Norway
saket@imsc.res.in

Prafullkumar Tale
Institute of Mathematical Sciences, HBNI, Chennai, India
pptale@imsc.res.in

Abstract

A graph G is contractible to a graph H if there is a set $X \subseteq E(G)$, such that G/X is isomorphic to H. Here, G/X is the graph obtained from G by contracting all the edges in X. For a family of graphs $\mathcal{F}$, the $\mathcal{F}$-Contraction problem takes as input a graph G on n vertices, and the objective is to output the largest integer t, such that G is contractible to a graph $H \in \mathcal{F}$, where $|V(H)| = t$. When $\mathcal{F}$ is the family of paths, then the corresponding $\mathcal{F}$-Contraction problem is called Path Contraction. The problem Path Contraction admits a simple algorithm running in time $2^n \cdot n^{\mathcal{O}(1)}$. In spite of the deceptive simplicity of the problem, beating the $2^n \cdot n^{\mathcal{O}(1)}$ bound for Path Contraction seems quite challenging. In this paper, we design an exact exponential time algorithm for Path Contraction that runs in time $1.99987^n \cdot n^{\mathcal{O}(1)}$. We also define a problem called 3-Disjoint Connected Subgraphs, and design an algorithm for it that runs in time $1.88^n \cdot n^{\mathcal{O}(1)}$. The above algorithm is used as a sub-routine in our algorithm for Path Contraction.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Theory of computation → Graph algorithms analysis; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases path contraction, exact exponential time algorithms, graph algorithms, enumerating connected sets, 3-disjoint connected subgraphs

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.11

Category Track A: Algorithms, Complexity and Games

Funding *Akanksha Agrawal*: During some part of the work, the author was supported by ERC Consolidator Grant SYSTEMATIC-GRAPH (No. 725978).
Saket Saurabh: This work is supported by the European Research Council (ERC) via grant LOPPRE, reference no. 819416.

1 Introduction

Graph editing problems are one of the central problems in graph theory that have received a lot of attention in algorithm design. Some of the natural graph editing operations are vertex/edge deletion, edge addition, and edge contraction. For a family of graphs $\mathcal{F}$, the

$\mathcal{F}$-Editing problem takes as input a graph G, and the objective is to find the minimum number of operations required to transform G into a graph from $\mathcal{F}$. In fact, the $\mathcal{F}$-Editing problem, where the edit operations are restricted to one of vertex deletion, edge deletion, edge addition, or edge contraction have also received a lot of attention in algorithm design. The $\mathcal{F}$-Editing problems encompass several classical NP-hard problems like Vertex Cover, Feedback Vertex Set, Odd Cycle Transversal, etc.

The $\mathcal{F}$-Editing problem where the only allowed edit operation is edge contraction, is called $\mathcal{F}$-Contraction. For a graph G and an edge $e = uv \in E(G)$, *contraction* of an edge uv in G results in a graph G/e, which is obtained by deleting u and v from G, adding a new vertex w_e and making w_e adjacent to the neighbors of u or v (other than u, v). A graph G is *contractible* to a graph H, if there exists a subset $X \subseteq E(G)$, such that if we contract each edge from X, then the resulting graph is isomorphic to H. For several families of graphs $\mathcal{F}$, early papers by Watanabe et al. [18, 19] and Asano and Hirata [1] showed that $\mathcal{F}$-Contraction is NP-hard. The NP-hardness of problems like Tree Contraction and Path Contraction, which are the $\mathcal{F}$-Contraction problems for the family of trees and paths, respectively, follows easily from [1, 3]. A restricted version of Path Contraction, is the problem P_t Contraction, where t is a fixed constant. P_t-Contraction is shown to be NP-hard even for $t = 4$, while for $t \leq 3$, the problem is polynomial time solvable [3]. P_t-Contraction alone had received lot of attention for smaller values of t, even when the input graph is from a very structured family of graphs (for instance, see [3, 17, 10, 6, 8, 13], and the references therein).

Several NP-hard problem like SAT, k-SAT, Vertex Cover, Hamiltonian Path, etc. are known to admit an algorithm running in time $\mathcal{O}^\star(2^n)$[1]. These results are obtained by techniques like brute force search, dynamic programming over subsets, etc. One of the main questions that arise in this context is: can we break the $\mathcal{O}^\star(2^n)$ barrier for these problems. In fact, the hardness of SAT gives rise to the Strong Exponential Time Hypothesis (SETH) of Impagliazzo and Paturi [12, 11], which rules out existence of $\mathcal{O}^\star((2-\epsilon)^n)$-time algorithm for SAT, for any $\epsilon > 0$. SETH has been used to obtain such algorithmic lower bounds for many other NP-hard problems (see for example, [4, 14]). Not all NP-hard problems seem to be as "hard" as SAT. For many NP-hard problems, it is possible to break the $\mathcal{O}^\star(2^n)$ barrier. For instance problems like Vertex Cover and (undirected) Hamiltonian Path are known to admit algorithms running in time $\mathcal{O}^\star((2-\epsilon)^n)$, for some $\epsilon > 0$ [2, 15]. Thus, one of the natural question is for which NP-hard problems can we avoid the "brute force search", and say obtain algorithms that are better than $\mathcal{O}^\star(2^n)$.

In this article, we focus on the problem Path Contraction, which is formally defined below.

Path Contraction
Input: Graph G.
Output: Largest integer t, such that G is contractible to P_t.

Path Contraction is known to admit a simple algorithm that runs in time $\mathcal{O}^\star(2^n)$. Such an algorithm can be obtained by coloring the input graph with two colors and contracting connected components in the colored subgraphs. For a deceptively simple problem like Path Contraction, it seems quite challenging to break the $\mathcal{O}^\star(2^n)$ barrier. The problem 2-Disjoint Connected Subgraphs (2-DCS), can be "roughly" interpreted as solving P_4-Contraction. (We can use the algorithm for 2-DCS to solve P_4-Contraction.) There

[1] The $\mathcal{O}^\star$ notation hides polynomial factors in the running time expression.

have been studies, which break the $\mathcal{O}^\star(2^n)$ brute force barrier, for 2-DCS. In particular, Cygan et al. [5] designed a $\mathcal{O}^\star(1.933^n)$ algorithm for 2-DCS. This result was improved by Telle and Villanger, who designed an algorithm running in time $\mathcal{O}^\star(1.7804^n)$, for the problem [16]. The main goal of this article is to break the $\mathcal{O}^\star(2^n)$ barrier for Path Contraction. Obtaining such an algorithm for Path Contraction was stated as an open problem in [17].

Our Results. We design an algorithm for Path Contraction running in time $\mathcal{O}^\star(1.99987^n)$, where n is the number of vertices in the input graph. To the best of our knowledge, this is the first non-trivial algorithm for the problem, which breaks the $\mathcal{O}^\star(2^n)$ barrier. To obtain our main algorithm for Path Contraction, we design four different algorithms for the problem, which are used as subroutines to the main algorithm. We exploit the property that certain types of algorithms are better for certain instance, but may be inefficient for certain other instances. Roughly speaking, we look for solutions using different algorithms, and then the best suited algorithm for the instance is used to return the solution. When one of the four algorithms is called as a subroutine, it does not necessarily return an optimum solution for the instance, rather it only looks for solutions that satisfy certain conditions. These conditions are quantified by fractions associated with the input graph. We note that for appropriate values of these "fractions", each of our subroutine still serve as an algorithm for Path Contraction (and thus, can compute the optimal solution). We argue that there is always a solution which satisfies the conditions for one of the subroutines, by setting the values of the fractions appropriately. A saving over $\mathcal{O}^\star(2^n)$, in the running time achieved by our algorithm, also exploits the property that "small" connected sets with bounded neighborhood can be enumerated "efficiently".

In the following we very briefly explain the type of solutions we look for, in our subroutines. Consider a path P_t, such that G can be contracted to P_t, where t is the largest such integer. The solution t, can be "witnessed" by a partition $\mathcal{W} = \{W_1, W_2, \cdots, W_t\}$ of $V(G)$, where the vertices from W_i "merge" to the ith vertex of P_t (a formal definition for it can be found in Section 2). Such a "witness" is called a *P_t-witness structure*. The first (subroutine) algorithm for Path Contraction searches for a solution where the P_t-witness structure can be "split" into two connected disjoint parts which are "small". Then, it exploits the "smallness" of the parts to compute solutions efficiently, and combines them to compute the solution for whole graph. The second subroutine searches for a pair of sets in the P_t-witness structure which are very dense. Then it exploit the sparseness of the remaining graph to efficiently compute partial solutions for them. Moreover, the pair of dense parts are resolved using the algorithm of Telle and Villanger for 2-Disjoint Connected Subgraph [16]. The third routine works with a hope that the total number of vertices in one of odd/even sets from $\mathcal{W}$ can be bounded. Finally, the fourth subroutine work by exploiting a similar odd/even property as the third subroutine, but it relaxes the condition to "nearly" small odd/even set.

To design our algorithm, we also define a problem called 3-Disjoint Connected Subgraphs (3-DCS), which is a generalization of the 2-Disjoint Connected Subgraphs (2-DCS) problem. 3-DCS takes as input a graph G and disjoint sets $Z_1, Z_2 \subseteq V(G)$, and the goal is to partition $V(G)$ into three sets (V_1, U, V_2), such that graphs induced on each of the parts is connected and $Z_i \subseteq V_i$, for $i \in [2]$. We design an algorithm for 3-DCS running in time $\mathcal{O}^\star(1.88^n)$. The fourth subroutine of our algorithm uses the algorithm for 3-DCS as a subroutine. As a corollary to our $\mathcal{O}^\star(1.88^n)$-time algorithm for 3-DCS, we obtain that P_5-Contraction admits an algorithm running in time $\mathcal{O}^\star(1.88^n)$.

Due to space limitation, most proofs appear in full version of the paper.

2 Preliminaries

In this section, we state some basic definitions and introduce terminologies from graph theory. We use standard terminology from the book of Diestel [7] for the graph related terminologies which are not explicitly defined here. We also establish some notations that will be used throughout. We note that all graphs considered in this article are connected graphs on at least two vertices (unless stated otherwise).

We denote the set of natural numbers by $\mathbb{N}$ (including 0). For $k \in \mathbb{N}$, $[k]$ denotes the set $\{1, 2, \ldots, k\}$. A graph G is *isomorphic* to a graph H if there exists a bijective function $\phi : V(G) \to V(H)$, such that for $v, u \in V(G)$, $uv \in E(G)$ if and only if $(\phi(v), \phi(u)) \in E(H)$. A graph G is *contractible* to a graph H if there exists $F \subseteq E(G)$, such that G/F is isomorphic to H. In other words, G is contractible to H if there is a surjective function $\varphi : V(G) \to V(H)$, with $W(h) = \{v \in V(G) \mid \varphi(v) = h\}$, for $h \in V(H)$, with the following properties:

- for any $h \in V(H)$, the graph $G[W(h)]$ is connected, and
- for any two vertices $h, h' \in V(H)$, $hh' \in E(H)$ if and only if $W(h)$ and $W(h')$ are adjacent in G.

Let $\mathcal{W} = \{W(h) \mid h \in V(H)\}$. The sets in $\mathcal{W}$ are called *witness sets*, and $\mathcal{W}$ is an *H-witness structure* of G.

In this paper, we will restrict ourselves to contraction to paths. This allows us to use an ordered notation for witness sets, rather than just the set notation. This ordering of the sets in witness set is given by the ordering of vertices in the path. That is, for a $P_t = (h_1, h_2, \cdots, h_t)$-witness structure, $\mathcal{W} = \{W(h_1), W(h_2), \cdots, W(h_t)\}$ of a graph G, we use the ordered witness structure notation, $(W(h_1), W(h_2), \cdots, W(h_t))$, or simply, $(W_1, W_2, \cdots, W_t)$. We note that we use both unordered and ordered notation, as per the convenience.

In the following, we give some useful observations regarding contraction to paths.

▶ **Observation 2.1.** *Let G be a graph contractible to P_t. Then, there is a P_t-witness structure, $\mathcal{W} = (W_1, \ldots, W_t)$, of G such that W_1 is a singleton set. Moreover, if $t \geq 3$, then there is a P_t-witness structure, $\mathcal{W} = (W_1, \ldots, W_t)$, of G such that both W_1 and W_t are singleton set.*

▶ **Observation 2.2.** *For a set U with n elements and a constant $\delta < 1/2$, the number of subsets of U of size at most δn is bounded by $\mathcal{O}^\star([g(\delta)]^n)$, where $g(\delta) = \frac{1}{\delta^\delta \cdot (1-\delta)^{(1-\delta)}}$. Moreover, all such subsets can be enumerated in the same time.*

For a graph G, a non-empty set $Q \subseteq V(G)$, and integers $a, b \in \mathbb{N}$, a connected set A in G is a *(Q, a, b)-connected set* if $Q \subseteq A$, $|A| = a$, and $|N(A)| \leq b$. Moreover, a connected set A in G is an *(a, b)-connected set* if $|A| \leq a$ and $|N(A)| \leq b$. Next, we state results regarding (Q, a, b)-connected sets and connected sets, which follow from Lemma 3.1 of [9]. (We note that their result give slightly better bounds, but for simplicity, we only use the bounds stated in the following lemmas.)

▶ **Lemma 1.** *For a graph G, a non-empty set $Q \subseteq V(G)$, and integers $a, b \in \mathbb{N}$, the number of (Q, a, b)-connected sets in G is at most $2^{a+b-|Q|}$. Moreover, we can enumerate all (Q, a, b)-connected sets in G in time $2^{a+b-|Q|} \cdot n^{\mathcal{O}(1)}$.*

▶ **Lemma 2.** *For a graph G and integers $a, b \in \mathbb{N}$ the number of (a, b)-connected sets in G is at most $2^{a+b} \cdot n^{\mathcal{O}(1)}$. Moreover, we can enumerate all such sets in $2^{a+b} \cdot n^{\mathcal{O}(1)}$ time.*

3 3-Disjoint Connected Subgraph

In this section, we define a generalization of 2-Disjoint Connected Subgraphs (2-DCS), called 3-Disjoint Connected Subgraphs (3-DCS). We design an algorithm for 3-DCS

running in time $\mathcal{O}^\star(1.88^n)$, where n is number of vertices in input graph. This algorithm will be useful in designing our algorithm for Path Contraction.

In the following, we formally define the problem 2-DCS which is studied in [5, 16].

2-Disjoint Connected Subgraphs (2-DCS)
Input: A connected graph G and two disjoint sets Z_1 and Z_2.
Question: Does there exist a partition (V_1, V_2) of $V(G)$, such that for each $i \in [2]$, $Z_i \subseteq V_i$ and $G[V_i]$ is connected?

In the following we state a result regarding 2-DCS which will be useful later sections.

▶ **Proposition 3** ([16] Theorem 3). *There exists an algorithm that solves* 2-Disjoint Connected Subgraphs *problem in* $\mathcal{O}^\star(1.7804^n)$ *time where* n *is number of vertices in input graph.*

In the 3-DCS problem, the input is same as that of 2-DCS, but we are interested in a partition of $V(G)$ into three sets, rather than two. We formally define the problem below.

3-Disjoint Connected Subgraphs (3-DCS)
Input: A connected graph G and two disjoint sets Z_1 and Z_2.
Question: Does there exist a partition (V_1, U, V_2) of $V(G)$, such that 1) for each $i \in [2]$, $Z_i \subseteq V_i$ and $G[V_i]$ is connected, 2) $G[U]$ is connected, and 3) $G - U$ has exactly two connected components, namely, $G[V_1]$ and $G[V_2]$?

We note that the problem definitions for 2-DCS and 3-DCS do not require the sets Z_1, Z_2 to be non-empty. If either of this set is empty, we can guess a vertex for each of the non-empty sets. Since there are at most n^2 such guesses, it will not affect the running time of our algorithm. Thus, here after we assume that both Z_1 and Z_2 are non-empty sets.

In the following theorem, we state our result regarding 3-DCS.

▶ **Theorem 4.** 3-DCS *admits an algorithm running in time* $\mathcal{O}^\star(1.88^n)$*, where* n *is number of vertices in the input graph.*

4 Exact Algorithm for Path Contraction

In this section we design our algorithm for Path Contraction running in time $\mathcal{O}^\star(1.99987^n)$, where n is the number of vertices in the input graph. To design our algorithm, we design four different subroutines each solving the problem Path Contraction. Each of these subroutines are better than the other when a specific "type" of solution exists for the input instance. Thus the main algorithm will use these subroutines to search for solutions of the type they are the best for. We also design a sub-routine for enumerating special types of partial solution, which will be used in some of our algorithms for Path Contraction.

In the following we briefly explain the four subroutines and describe when they are useful. Let G be an instance for Path Contraction, where G is a graph on n vertices. Let t be the largest integer (which we do not know a priori), such that G is contractible to P_t with $(W_1, W_2, \cdots, W_t)$ as a P_t-witness structure of G. We let $\mathtt{OS}$ and $\mathtt{ES}$ be the union of vertices in odd and even witness sets, respectively. That is, $\mathtt{OS} = \cup_{x=1}^{\lceil t/2 \rceil} W_{2x-1}$ and $\mathtt{ES} = \bigcup_{x=1}^{\lfloor t/2 \rfloor} W_{2x}$.

We now give an intuitive idea of the purposes of each of our subroutines in the main algorithm, while deferring their implementations to the subsequent sections. We also describe a subroutine which will help us build "partial solutions", and this subroutine will be used in two of our subroutines for Path Contraction. (We refer the reader to Figure 1 for an illustration of it.)

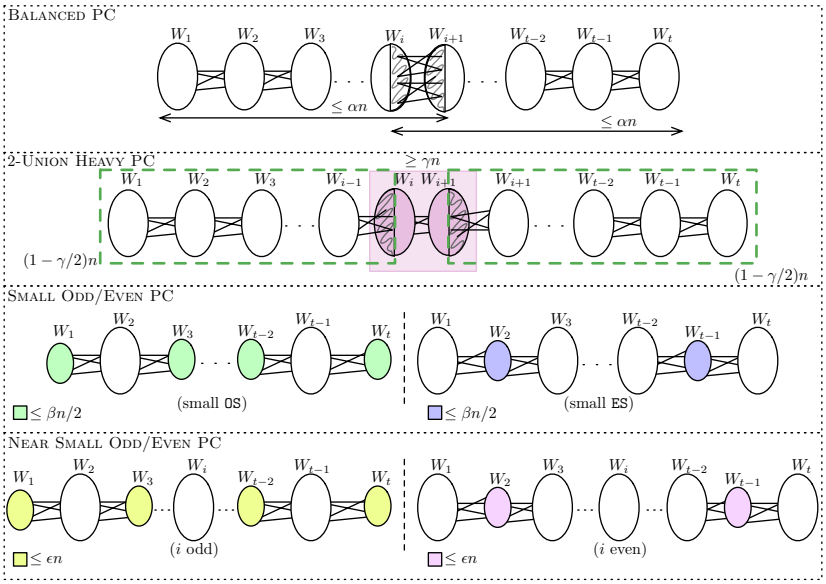


Figure 1 Various subroutines for the algorithm and their usage.

Balanced PC. This subroutine is useful when we can "break" the graph into two parts after a witness set, such that the closed neighborhood for each of the parts have small size, or in other words, the parts are "balanced". The quantification of the "balancedness" after a witness set will be done with the help of a rational number $0 < \alpha \leq 1$, which will be part of the input for the subroutine. The subroutine will only look for those P_t-witness structures for G for which there is an integer $i \in [t]$, such that the sizes of both $N[\cup_{j\in[i]}W_j]$ and $N[\cup_{j\in[t]\setminus[i]}W_j]$ are bounded by αn. Moreover, the algorithm will return the largest such t. Our algorithm for Balanced PC will run in time $\mathcal{O}^\star(2^{\alpha n})$. Note that when $\alpha = 1$, Balanced PC is an algorithm for Path Contraction running in time $\mathcal{O}^\star(2^n)$.

2-Union Heavy PC. This subroutine will be used when "large" part of the graph is concentrated in two consecutive witness sets and the neighborhood of the rest of the graph into them is "small". The quantification of term "large/small" will be done by a a fraction $0 < \gamma < 1$, which will be part of the input. The algorithm will only search for those P_t-witness structure of G where there is an integer $i \in [t-1]$, such that $|W_i \cup W_{i+1}| \geq \gamma n$, and $|N[\cup_{j\in[i-1]}W_j]|, |N[\cup_{j\in[t]\setminus[i+1]}W_j]| \leq (1-\gamma/2)n$. Moreover, the algorithm will return largest such t.

Small Odd/Even PC. Roughly speaking, this subroutine is particularly useful when one of OS or ES is "small". The "smallness" of OS/ES is quantified by a rational number $0 < \beta \leq 1$, which will be part of the input. The subroutine will only look for those P_t-witness structures for G where one of $|\texttt{OS}| \leq \beta n/2$ or $|\texttt{ES}| \leq \beta n/2$ holds. Moreover, the algorithm will return the largest integer $t \geq 1$, for which such a P_t-witness structure for G exists. Small Odd/Even PC will run in time $\mathcal{O}^\star(c^n)$, where $c = g(\beta/2)$. We note that when $\beta = 1$, then one of $|\texttt{OS}| \leq \beta n/2$ or $|\texttt{ES}| \leq \beta n/2$ definitely holds. Thus, for $\beta = 1$, Small Odd/Even PC is an algorithm for Path Contraction running in time $\mathcal{O}^\star(2^n)$ (see Observation 2.2).

Near Small Odd/Even PC. In the case when both OS and ES are "large", it may be the case that for one of OS/ES, there is just one witness set which is large. That is, when we remove this large witness set, then one of OS/ES becomes "small". The "smallness" of the remaining OS/ES (after removing a witness set) will be quantified by a rational number $0 < \epsilon \leq 1$, which will be part of the input. The subroutine will only look for those P_t-witness structures for G where the size of one of $|\texttt{OS}|$ or $|\texttt{ES}|$ after removal of a witness set is bounded by ϵn. Moreover, the algorithm will return the largest such t.

Our subroutines Balanced PC and 2-Union Heavy PC use a subroutine called Enum-Partial-PC for enumerating solutions for "small" subgraphs. The efficiency of the algorithm for Enum-Partial-PC is centered around the bounds for (Q, a, b)-connected sets. In Section 4.1 we (define and) design an algorithm for Enum-Partial-PC. In Section 4.2, 4.3, 4.4 and 4.5 we present our algorithms for Balanced PC, 2-Union Heavy PC, Small Odd/Even PC, and Near Small Odd/Even PC, respectively. Finally, in Section 4.6 we show how we can use the above algorithms to obtain an algorithm for Path Contraction, running in time $\mathcal{O}^\star(1.99987^n)$.

4.1 Algorithm for Enum-Partial-PC

In this section, we describe an algorithm which computes "nice solution" for all "ρ-small" subset of vertices of an input graph. In an input graph G, for a set $S \subseteq V(G)$, by $\Phi(S)$ we denote the set of vertices in S that have a neighbor outside S. That is, $\Phi(S) = \{s \in S \mid N(s) \setminus S \neq \emptyset\}$. A set $S \subseteq V(G)$ is *ρ-small* if $N[S] \leq \rho n$. For an ρ-small set $S \subseteq V(G)$, the largest integer t_S is called the *nice solution* if $G[S]$ is contractible to P_{t_S} with all the vertices in $\Phi(S)$ in the end bag. That is, there is a P_{t_S}-witness structure $(W_1, W_2, \cdots W_{t_S})$ of $G[S]$, such that $\Phi(S) \subseteq W_{t_S}$. We formally define the problem Enum-Partial-PC in the following way.

Enum-Partial-PC
Input: A graph G on n vertices and a fraction $0 < \rho \leq 1$.
Output: A table Γ which is indexed by *ρ-small* sets. For any *ρ-small* set S, $\Gamma[S]$ is the largest integer t for which $G[S]$ has a P_t-witness structure $\mathcal{W} = (W_1, W_2, \cdots, W_t)$, such that $\Phi(S) \subseteq W_t$.

We design an algorithm for Enum-Partial-PC running in time $\mathcal{O}^\star(2^{\rho n})$. We briefly explain how we can compute nice solutions for ρ-small set. Consider an ρ-small set S. Note that $|S| \leq \rho n$. Thus, by the method of 2-coloring (as was explained in the introduction), we can obtain the nice solution in time $2^{\rho n}$. This would lead us to an algorithm running in time $\mathcal{O}^\star(2^{\rho n} g(\rho)^n)$. By doing a simple dynamic programming we can also obtain an algorithm running in time $\mathcal{O}^\star(3^n)$. We will improve upon these algorithms by a dynamic programming algorithm where we update the values "forward" instead of looking "backward".

The Algorithm. We start by defining the tables entries for our dynamic programming routine, which is used for computation of nice solutions. Let $\mathcal{S}$ be the set of all connected sets S in G, such that $|N[S]| \leq \rho n$. That is, $\mathcal{S} = \{S \subseteq V(G) \mid G[S] \text{ is connected and } |N[S]| \leq \rho n\}$. For each $S \in \mathcal{S}$, we have an entry denoted by $\Gamma[S]$. $\Gamma[S]$ is the largest integer $q \geq 1$ for which $G[S]$ can be contracted to P_q with a P_q-witness structure $\mathcal{W} = (W_1, W_2, \cdots, W_q)$ of $G[S]$, such that $\Phi(S) \subseteq W_q$. The algorithm starts by initializing $\Gamma[S] = 1$, for each $S \in \mathcal{S}$.

In the following we introduce some notations that will be useful in stating the algorithm. Consider $S \in \mathcal{S}$. We will define a set $\mathcal{A}[S]$, which will be the set of all "potential extenders bags" for S, when we look at contraction to paths for larger graphs (containing S). For the

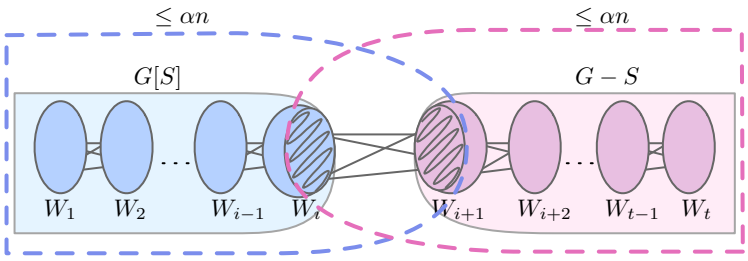


Figure 2 An illustration of construction of the solution using solutions for instances of smaller sizes.

sake of notational simplicity, we will define $\mathcal{A}_{a,b}[S] \subseteq \mathcal{A}[S]$, where the sets in $\mathcal{A}_{a,b}[S]$ will be of size exactly a and will have exactly b neighbors outside S. We will define the above sets only for "relevant" as and bs. We now move to the formal description of these sets. Consider $S \in \mathcal{S}$ and integers a, b, such that $|S| + a + b \leq \rho n$ and $|N(S)| \leq b$. We let $\mathcal{A}_{a,b}[S] = \{A \subseteq V(G - S) \mid G - S[A]$ is connected, $N_G(S) \subseteq A, |A| = a,$ and $|N_{G-S}(A)| = b\}$.

The algorithm now computes nice solutions. The algorithm considers sets from $S \in \mathcal{S}$, in increasing order of their sizes and does the following. (Two sets that have the same size can be considered in any order.) For every pair of integer a, b, such that $|S| + a + b \leq \rho n$ and $|N(S)| \leq b$, it computes the set $\mathcal{A}_{a,b}[S]$. Note that $\mathcal{A}_{a,b}[S]$ can be computed in time $\mathcal{O}^\star(2^{a+b-|S|})$, using Lemma 1. Now the algorithm considers $A \in \mathcal{A}_{a,b}[S]$. Intuitively speaking, A is the "new" witness set to be "appended" to the witness structure of $G[S]$, to obtain a witness structure for $G[S \cup A]$. Thus, the algorithm sets $\Gamma[S \cup A] = \max\{\Gamma[S \cup A], \Gamma[S] + 1\}$. This finishes the description of our algorithm.

In the following few lemmas we establish the correctness and runtime analysis of the algorithm.

▶ **Lemma 5.** *For each $S \in \mathcal{S}$, the algorithm computes $\Gamma[S]$ correctly.*

▶ **Lemma 6.** *The algorithm presented for* Enum-Partial-PC *runs in time $\mathcal{O}^\star(2^{\rho n})$.*

4.2 Algorithm for Balanced PC

We formally define the problem Balanced PC in the following.

Balanced PC
Input: A graph G on n vertices and a fraction $0 < \alpha \leq 1$.
Output: Largest integer $t \geq 2$ for which G has a P_t-witness structure $\mathcal{W} = (W_1, W_2, \cdots, W_t)$, such that there is $i \in [t]$ with $N[\cup_{j \in [i]} W_j] \leq \alpha n$ and $N[\cup_{j \in [t] \setminus [i]} W_j] \leq \alpha n$. Moreover, if no such t exists, then output 1.

We design an algorithm for Balanced PC running in time $\mathcal{O}^\star(2^{\alpha n})$. Let (G, α) be an instance of Balanced PC.

We begin by explaining the intuition behind the algorithm. Recall that for a α-small set $S \subseteq V(G)$, integer t_S is called the *nice solution* if $G[S]$ is contractible to P_{t_S} with all the vertices in $\Phi(S)$ in the end bag. That is, there is a P_{t_S}-witness structure $(W_1, W_2, \cdots W_{t_S})$ of $G[S]$, such that $\Phi(S) \subseteq W_{t_S}$. Suppose that we know the value of t_S for every α-small set S. Now we see how we can use these nice solutions for α-small sets to solve our problem (see Figure 2). Recall that we are looking for the largest integer t, such that G is contractible to P_t, with $\mathcal{W} = (W_1, W_2, \cdots, W_t)$ as a P_t-witness structure of G, such that there is $i \in [t]$ with

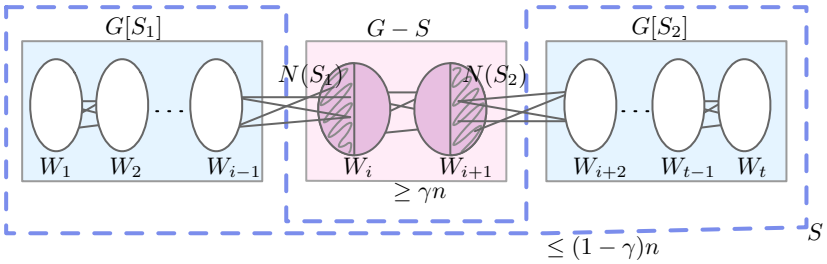


Figure 3 An intuitive illustration of the algorithm for 2-Union Heavy PC.

$|\cup_{j\in[i+1]} W_j| \leq \alpha n$ and $|\cup_{j\in[t]\setminus[i-1]} W_j| \leq \alpha n$. Let $S = \cup_{j\in[i]} W_j$. As $|\cup_{j\in[i+1]} W_j| \leq \alpha n$ and $N(S) \subseteq W_{i+1}$, the set S is an α-small set. Similarly, we can argue that $V(G) \setminus S$ is an α-small set. Thus, for S and $V(G) \setminus S$, we know the nice solutions t_S and $t_{V(G)\setminus S}$, respectively. Notice that the solution to the whole graph is actually $t_S + t_{V(G)\setminus S}$.

The Algorithm. The algorithm initializes $t = 1$. (At the end, t will be the output of the algorithm.) The algorithm computes table $\Gamma =$Enum-Partial-PC(G, α) using algorithm from Section 4.1. Let $\mathcal{S}$ be the set of all connected sets S in G, such that $|N[S]| \leq \alpha n$. That is, $\mathcal{S} = \{S \subseteq V(G) \mid G[S]$ is connected and $|N[S]| \leq \alpha n\}$. For each $S \in \mathcal{S}$, we have an entry denoted by $\Gamma[S]$. The algorithm considers each $S \in \mathcal{S}$ for which $V(G) \setminus S \in \mathcal{S}$. It sets $t = \max\{t, \Gamma[S] + \Gamma[V(G) \setminus S]\}$. Finally, the algorithm returns t as the output. This completes the description of the algorithm.

▶ **Lemma 7.** *The algorithm presented for* Balanced PC *is correct.*

▶ **Lemma 8.** *The algorithm presented for* Balanced PC *runs in time* $\mathcal{O}^\star(2^{\alpha n})$.

4.3 Algorithm for 2-Union Heavy PC

We formally define the problem 2-Union Heavy PC in the following (also see Figure 1).

2-Union Heavy PC
Input: A graph G on n vertices and a fraction $0 < \gamma \leq 1$.
Output: Largest integer $t \geq 3$ for which G has a P_t-witness structure $\mathcal{W} = (W_1, W_2, \cdots, W_t)$, such that there is $i \in [t-1]$ for which the following conditions hold: 1) $|W_i \cup W_{i+1}| \geq \gamma n$ and 2) $|N[\cup_{j\in[i-1]} W_j]|, |N[\cup_{j\in[t]\setminus[i+1]} W_j]| \leq (1-\gamma/2)n$. Moreover, if no such t exists, then output 2.

We design an algorithm for 2-Union Heavy PC running in time $\mathcal{O}^\star(2^{(1-\gamma/2)n} + c^n)$, where $c = \max_{\gamma\leq\delta\leq 1}\{1.7804^\delta \cdot g(1-\delta)\}$. The first term in the running time expression will be due to a call made to Enum-Partial-PC with $\rho = (1-\gamma/2)$, and the second term will be due to enumerating sets of size at most $(1-\gamma)n$ and running the algorithm for solving 2-Disjoint Connected Subgraphs for an instance created for each of them, using the algorithm of Telle and Villanger [16].

Let (G, γ) be an instance of 2-Union Heavy PC. We start by explaining the intuitive idea behind our algorithm (see Figure 3). Consider a P_t-witness structure $\mathcal{W} = (W_1, W_2, \cdots, W_t)$ of G, such that there is $i \in [t-1]$ for which the following conditions hold: 1) $|W_i \cup W_{i+1}| \geq \gamma n$ and 2) $|N[\cup_{j\in[i-1]} W_j]|, |N[\cup_{j\in[t]\setminus[i+1]} W_j]| \leq (1-\gamma/2)n$. Let $S = W_i \cup W_{i+1}$. As $W_i \cup W_{i+1}$ is "large", the number of vertices in $S = V(G) \setminus (W_i \cup W_{i+1})$ is "small". That is, we can bound $|S|$ by $(1-\gamma)n$. Note that $G[S]$ has exactly two connected components, which we

denote by $G[S_1]$ and $G[S_2]$. Also note that $N[S_1], N[S_2] \leq (1-\gamma/2)n$. The algorithm starts by enumerating all such "potential candidates" for S. As for each of the two components of $G[S]$, the sizes of $N[S_1]$ and $N[S_2]$ can be bounded, the algorithm computes the "optimum solution" for them using the algorithm for Enum-Partial-PC. In the above we use the algorithm for Enum-Partial-PC because we are only interested in those solutions where the vertices of $\Phi(S_1)$ and $\Phi(S_2)$ are contained in one of the "end bags" of their respective solutions. Now we see how we can use these solutions to obtain the solution for the whole graph. Note that we have to "split" vertices in $V(G) \setminus S$ into two "connected sets", where the first set must contain all the vertices from $N(S_1)$ and the second set must contain all the vertices from $N(S_1)$. For the above we employ the algorithm for 2-Disjoint Connected Subgraphs (see Section 3 for its definition) by Telle and Villanger [16].

We now formally describe our algorithm. The algorithm will output an integer t, which is initially set to 2. Let $\mathcal{S} = \{S \subseteq V(G) \mid |S| \leq (1-\gamma)n$ and $G[S]$ has exactly two connected components $G[S_1], G[S_2]$, s.t. $|N[S_1]|, |N[S_2]| \leq (1-\gamma/2)n\}$. Let $\widehat{\mathcal{S}} = \{\widehat{S} \subseteq V(G) \mid |N[\widehat{S}]| \leq (1-\gamma/2)n$ and $G[\widehat{S}]$ is connected$\}$. The algorithm will now computes a table Γ, which has an entry $\Gamma[\widehat{S}]$, for each $\widehat{S} \in \widehat{\mathcal{S}}$. The definition of Γ is the same as that in Section 4.2, where $\rho = 1-\gamma/2$. That is, for $\widehat{S} \in \widehat{\mathcal{S}}$, $\Gamma[\widehat{S}]$ is the largest integer $q \geq 1$ for which $G[\widehat{S}]$ can be contracted to P_q with a P_q-witness structure $\mathcal{W} = (W_1, W_2, \cdots, W_q)$ of $G[\widehat{S}]$, such that $\Phi(\widehat{S}) \subseteq W_q$. Compute the value of $\Gamma[\widehat{S}]$, for each $\widehat{S} \in \widehat{\mathcal{S}}$, by using Enum-Partial-PC$(G, 1-\gamma/2)$. For each $S \in \mathcal{S}$, the algorithm does the following. Recall that $G[S]$ has exactly two connected components. Let the two connected components in $G[S]$ be $G[S_1]$ and $G[S_2]$, where $S_1 \cup S_2 = S$. Recall that $|N[S_1]|, N[S_1]| \leq (1-\gamma/2)n$. Thus, $S_1, S_2 \in \widehat{\mathcal{S}}$. If $(G - S, N_G(S_1), N_G(S_2))$ is a yes-instance of 2-DCS, then the algorithm sets $t = \max\{t, \Gamma[S_1] + \Gamma[S_2] + 2\}$, and otherwise, it moves to the next set in $\mathcal{S}$. Finally, the algorithm outputs t. This completes the description of the algorithm.

In the following two lemmas we present the correctness and runtime analysis of the algorithm, respectively.

▶ **Lemma 9.** *The algorithm presented for* 2-Union Heavy PC *is correct.*

▶ **Lemma 10.** *The algorithm presented for* 2-Union Heavy PC *runs in time* $\mathcal{O}^\star(2^{(1-\gamma/2)n} + c^n)$*, where* $c = \max_{\gamma \leq \delta \leq 1}\{1.7804^\delta \cdot g(1-\delta)\}$.

4.4 Algorithm for Small Odd/Even PC

We formally define the problem Small Odd/Even PC in the following.

Small Odd/Even PC
Input: A graph G on n vertices and a fraction $0 < \beta \leq 1$.
Output: Largest integer t for which G can be contracted to P_t, with $\mathcal{W} = (W_1, W_2, \cdots, W_t)$ as a P_t-witness structure of G, such that $|\mathtt{OS}_\mathcal{W}| \leq \beta n/2$ or $|\mathtt{ES}_\mathcal{W}| \leq \beta n/2$, where $\mathtt{OS}_\mathcal{W} = \cup_{i \in [\lceil t/2 \rceil]} W_{2i-1}$ and $\mathtt{ES}_\mathcal{W} = \cup_{i \in [\lfloor t/2 \rfloor]} W_{2i}$.

In this section, we design an algorithm for Small Odd/Even PC running in time $\mathcal{O}^\star(c^n)$, where $c = g(\beta/2)$.

Let (G, β) be an instance of Small Odd/Even PC. The algorithm is fairly simple. It starts by enumerating all "potential candidates" for $\mathtt{OS}_\mathcal{W}$ (resp. $\mathtt{ES}_\mathcal{W}$), i.e., the set of all subsets of $V(G)$ of size at most $\beta n/2$. Then, for each such "potential set", it contracts G appropriately, and finds the length of the path to which G is contracted (and stores 0, if the contracted graph is not a path). Finally, it returns the maximum over such path lengths.

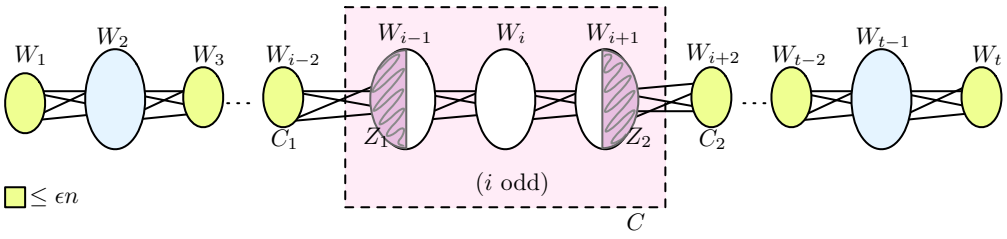


Figure 4 An intuitive illustration of the algorithm for Near Small Odd/Even PC.

We now move to formal description of the algorithm. We start by enumerating the set of all subsets of $V(G)$ of size at most $\beta n/2$. That is, $\mathcal{S} = \{S \subseteq V(G) \mid |S| \leq \beta n/2\}$. Note that $\mathcal{S}$ can be computed in time $\mathcal{O}^\star(g(\beta/2)^n)$, using Observation 2.2. For each $S \in \mathcal{S}$ the algorithm does the following. Let $\mathcal{C}_S$ and $\overline{\mathcal{C}}_S$ be the set of connected components of $G[S]$ and $G - S$, respectively. Let G_S be the graph obtained from G by contracting each $C \in \mathcal{C}_S \cup \overline{\mathcal{C}}_S$ to a single vertex. Set $\mathsf{len}_S = |V(G_S)|$, if G_S is a path, and $\mathsf{len}_S = 0$, otherwise. Finally, the algorithm returns $\max_{S \in \mathcal{S}} \mathsf{len}_S$.

In the following lemma we prove the correctness and runtime analysis of the algorithm.

▶ **Lemma 11.** *The algorithm presented for* Small Odd/Even PC *is correct and runs in time* $\mathcal{O}^\star(g(\beta/2)^n)$.

4.5 Algorithm for Near Small Odd/Even PC

We formally define the problem Near Small Odd/Even PC in the following (also see Figure 1).

Near Small Odd/Even PC
Input: A graph G on n vertices and a fraction $0 < \epsilon \leq 1$.
Output: Largest integer $t \geq 3$ for which there is a P_t-witness structure $\mathcal{W} = (W_1, W_2, \cdots, W_t)$ of G, for which there is $i \in \{2, 3, \cdots, t-1\}$, such that if i is odd, then $|\mathtt{OS}_{\mathcal{W}} \setminus W_i| \leq \epsilon n$ and otherwise, $|\mathtt{ES}_{\mathcal{W}} \setminus W_i| \leq \epsilon n$. Here, $\mathtt{OS}_{\mathcal{W}} = \cup_{i \in [\lceil t/2 \rceil]} W_{2i-1}$ and $\mathtt{ES}_{\mathcal{W}} = \cup_{i \in [\lfloor t/2 \rfloor]} W_{2i}$. If no such $t \geq 3$ exists, then output 2.

We design an algorithm for Near Small Odd/Even PC running in time $\mathcal{O}^\star(c^n)$ where $c = \max_{0 \leq \delta \leq \epsilon}\{1.88^{(1-\delta)} \cdot g(\delta)\}$. The second term in multiplicative factor will be due enumeration of sets, and the first term will be due to calls made to the algorithm for 3-Disjoint Connected Subgraphs, from Section 3.

Let (G, ϵ) be an instance of Near Small Odd/Even PC. We start by explaining the intuitive idea behind our algorithm (see Figure 4). Consider a P_t-witness structure $\mathcal{W} = (W_1, W_2, \cdots, W_t)$ of G, for which there is $i \in \{2, 3, \cdots, t-2\}$, such that if i is odd, then $|\mathtt{OS}_{\mathcal{W}} \setminus W_i| \leq \epsilon n$ and otherwise, $|\mathtt{ES}_{\mathcal{W}} \setminus W_i| \leq \epsilon n$. In the above, $\mathtt{OS}_{\mathcal{W}} = \cup_{i \in [\lceil t/2 \rceil]} W_{2i-1}$ and $\mathtt{ES}_{\mathcal{W}} = \cup_{i \in [\lfloor t/2 \rfloor]} W_{2i}$. Let us consider the case when i is odd (the other case is symmetric). Let $S = \mathtt{OS}_{\mathcal{W}} \setminus W_i$. (The union of vertices from yellow sets in Figure 4 is the set S.) As $|S| \leq \epsilon n$, the algorithm starts by enumerating all "potential candidates" for the set S. All the components of $G - S$, except for the component C, containing W_i, must each be contracted to a single vertex. Similarly, the components of $G[S]$ must each be contracted to a single vertex. Moreover, the component containing W_i must be "split" into three sets. The first and the last sets in the "split" must contain the neighbors of W_{i-2} and W_{i+2} in C, respectively. To obtain such a "split", we use the algorithm for 3-Disjoint Connected Subgraphs that we designed in Section 3.

We now formally describe our algorithm. The algorithm will output an integer t, which is initially set to 2. Let $\mathcal{S} = \{S \subset V(G) \mid |S| \leq \epsilon n\}$. For each $S \in \mathcal{S}$, the algorithm does the following. Let $\mathcal{C}_S$ and $\overline{\mathcal{C}}_S$ be the sets of connected components in $G[S]$ and $G - S$, respectively. Let H_S be obtained from G by contracting component in $\mathcal{C}_S \cup \overline{\mathcal{C}}_S$ to single vertices. That is, H_S has a vertex v_C for each $C \in \mathcal{C}_S \cup \overline{\mathcal{C}}_S$, and two vertices $v_C, v_{C'} \in V(H_S)$ are adjacent in H_S if and only if C and C' are adjacent in G. If H_S is not a path, then the algorithm moves to the next set in $\mathcal{S}$. Otherwise, for each $C^* \in \overline{\mathcal{C}}_S$ it does the following. Intuitively speaking, C^* is the current guess for the component containing vertices from W_i for the witness structure that we are looking for. Note that C^* can be adjacent to at most two components from $\mathcal{C}_S$, as H_S is a path. Moreover, C^* must be adjacent to at least one component from $\mathcal{C}_S$, as G is connected and S is a strict subset of $V(G)$, i.e., $S \neq V(G)$. Let C_1 be a component from $\mathcal{C}_S$ that is adjacent to C^* in G, and $Z_1 = N(C_1) \cap V(C^*)$. Let $C_2 \in \mathcal{C}_S \setminus \{C_1\}$ be a component of $G[S]$ that is adjacent to C^*, and $Z_2 = N(C_2) \cap V(C^*)$. If such a C_2 does not exist, then we set $Z_2 = \emptyset$. If $(G[C^*], Z_1, Z_2)$ is a yes-instance of 3-DCS, then the algorithm updates $t = \max\{t, |V(H_S)| + 2\}$. After finishing the processing for each $S \in \mathcal{S}$, the algorithm outputs t. This finishes the description of our algorithm.

In the following two lemmas we present the correctness and runtime analysis of the algorithm, respectively.

▶ **Lemma 12.** *The algorithm presented for* Near Small Odd/Even PC *is correct.*

▶ **Lemma 13.** *The algorithm presented for* Near Small Odd/Even PC *runs in time* $\mathcal{O}^\star(c^n)$*, where* $c = \max_{0 \leq \delta \leq \epsilon}\{1.88^{(1-\delta)} \cdot g(\delta)\}$.

4.6 Algorithm for Path Contraction

We are now ready to present our algorithm for Path Contraction. The algorithm calls four of the subroutines Small Odd/Even PC, Balanced PC, 2-Union Heavy PC, and Near Small Odd/Even PC for appropriate instances, and returns the maximum of their outputs. In the following theorem, we present the algorithm, which is the main result of this paper.

▶ **Theorem 14.** Path Contraction *admits an algorithm running in time* $\mathcal{O}^\star(1.99987^n)$*, where* n *is the number of vertices in the input graph.*

5 Conclusion

We generalized the 2-Disjoint Connected Subgraphs problem, to a problem called 3-Disjoint Connected Subgraphs, where instead of partitioning the vertex set into two connected sets, we are required to partition it into three connected sets. We gave an algorithm for 3-Disjoint Connected Subgraphs running in time $\mathcal{O}^\star(1.88^n)$. We believe that this algorithm can be of independent interest and may find other algorithmic applications. We designed an algorithm for Path Contraction which breaks the $\mathcal{O}^\star(2^n)$ barrier. It was surprising that even for a simple problem like Path Contraction, there was no known algorithm that solves it faster than $\mathcal{O}^\star(2^n)$. Our algorithm for Path Contraction relied the fact that the number of (Q, a, b)-connected sets can be bounded by $\mathcal{O}^\star(2^{a+b-|Q|})$. This gives us savings in the number of states that we consider, in our dynamic programming routine (for enumerating partial solutions). We designed four different algorithms for Path Contraction and used them for appropriate instances, to obtain the main algorithm for Path Contraction.

References

1 Takao Asano and Tomio Hirata. Edge-Contraction Problems. *Journal of Computer and System Sciences*, 26(2):197–208, 1983.

2 Andreas Björklund. Determinant Sums for Undirected Hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014.

3 Andries Evert Brouwer and Hendrik Jan Veldman. Contractibility and NP-completeness. *Journal of Graph Theory*, 11(1):71–79, 1987.

4 Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On Problems as Hard as CNF-SAT. *ACM Trans. Algorithms*, 12(3):41:1–41:24, 2016.

5 Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. Solving the 2-disjoint connected subgraphs problem faster than 2^n. *Algorithmica*, 70(2):195–207, 2014.

6 Konrad K Dabrowski and Daniël Paulusma. Contracting bipartite graphs to paths and cycles. *Information Processing Letters*, 127:37–42, 2017.

7 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

8 Jiří Fiala, Marcin Kamiński, and Daniël Paulusma. A note on contracting claw-free graphs. *Discrete Mathematics and Theoretical Computer Science*, 15(2):223–232, 2013.

9 Fedor V. Fomin and Yngve Villanger. Treewidth computation and extremal combinatorics. *Combinatorica*, 32(3):289–308, 2012.

10 Pinar Heggernes, Pim van 't Hof, Benjamin Lévêque, and Christophe Paul. Contracting chordal graphs and bipartite graphs to paths and trees. *Discrete Applied Mathematics*, 164:444–449, 2014.

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

12 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

13 Walter Kern and Daniel Paulusma. Contracting to a Longest Path in H-Free Graphs. *arXiv preprint*, 2018. `arXiv:1810.01542`.

14 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known Algorithms on Graphs of Bounded Treewidth Are Probably Optimal. *ACM Trans. Algorithms*, 14(2):13:1–13:30, 2018.

15 Robert Endre Tarjan and Anthony E. Trojanowski. Finding a Maximum Independent Set. *SIAM J. Comput.*, 6(3):537–546, 1977.

16 Jan Arne Telle and Yngve Villanger. Connecting terminals and 2-disjoint connected subgraphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 418–428. Springer, 2013.

17 Pim van't Hof, Daniël Paulusma, and Gerhard J Woeginger. Partitioning graphs into connected parts. *Theoretical Computer Science*, 410(47-49):4834–4843, 2009.

18 Toshimasa Watanabe, Tadashi Ae, and Akira Nakamura. On the removal of forbidden graphs by edge-deletion or by edge-contraction. *Discrete Applied Mathematics*, 3(2):151–153, 1981.

19 Toshimasa Watanabe, Tadashi Ae, and Akira Nakamura. On the NP-hardness of Edge-Deletion and Contraction Problems. *Discrete Applied Mathematics*, 6(1):63–78, 1983.

Deterministic Combinatorial Replacement Paths and Distance Sensitivity Oracles

Noga Alon
Department of Mathematics, Princeton University, Princeton, NJ 08544, USA
Schools of Mathematics and Computer Science, Tel Aviv University, Tel Aviv 69978, Israel
nogaa@tau.ac.il

Shiri Chechik
Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel
shiri.chechik@gmail.com

Sarel Cohen
Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel
sarelcoh@post.tau.ac.il

Abstract

In this work we derandomize two central results in graph algorithms, replacement paths and distance sensitivity oracles (DSOs) matching in both cases the running time of the randomized algorithms.

For the replacement paths problem, let $G = (V, E)$ be a directed unweighted graph with n vertices and m edges and let P be a shortest path from s to t in G. The *replacement paths* problem is to find for every edge $e \in P$ the shortest path from s to t avoiding e. Roditty and Zwick [ICALP 2005] obtained a randomized algorithm with running time of $\widetilde{O}(m\sqrt{n})$. Here we provide the first deterministic algorithm for this problem, with the same $\widetilde{O}(m\sqrt{n})$ time. Due to matching conditional lower bounds of Williams et al. [FOCS 2010], our deterministic combinatorial algorithm for the replacement paths problem is optimal up to polylogarithmic factors (unless the long standing bound of $\widetilde{O}(mn)$ for the combinatorial boolean matrix multiplication can be improved). This also implies a deterministic algorithm for the second simple shortest path problem in $\widetilde{O}(m\sqrt{n})$ time, and a deterministic algorithm for the k-simple shortest paths problem in $\widetilde{O}(km\sqrt{n})$ time (for any integer constant $k > 0$).

For the problem of distance sensitivity oracles, let $G = (V, E)$ be a directed graph with real-edge weights. An f-Sensitivity Distance Oracle (f-DSO) gets as input the graph $G = (V, E)$ and a parameter f, preprocesses it into a data-structure, such that given a query (s, t, F) with $s, t \in V$ and $F \subseteq E \cup V, |F| \leq f$ being a set of at most f edges or vertices (failures), the query algorithm efficiently computes the distance from s to t in the graph $G \setminus F$ (i.e., the distance from s to t in the graph G after removing from it the failing edges and vertices F).

For weighted graphs with real edge weights, Weimann and Yuster [FOCS 2010] presented several randomized f-DSOs. In particular, they presented a combinatorial f-DSO with $\widetilde{O}(mn^{4-\alpha})$ preprocessing time and subquadratic $\widetilde{O}(n^{2-2(1-\alpha)/f})$ query time, giving a tradeoff between preprocessing and query time for every value of $0 < \alpha < 1$. We derandomize this result and present a combinatorial deterministic f-DSO with the same asymptotic preprocessing and query time.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Dynamic graph algorithms

Keywords and phrases replacement paths, distance sensitivity oracles, derandomization

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.12

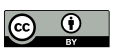
Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at [3], `https://arxiv.org/abs/1905.07483`.

Funding *Noga Alon*: Research supported in part by NSF grant DMS-1855464, ISF grant 281/17 and GIF grant G-1347-304.6/2016.

Shiri Chechik: Research supported in part by the Israel Science Foundation grant No. 1528/15 and the Blavatnik Fund.

Sarel Cohen: Research supported in part by the Israel Science Foundation grant No. 1528/15 and the Blavatnik Fund.



46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 12; pp. 12:1–12:14

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

1 Introduction

In many algorithms used in computing environments such as massive storage devices, large scale parallel computation, and communication networks, recovering from failures must be an integral part. Therefore, designing algorithms and data structures whose running time is efficient even in the presence of failures is an important task. In this paper we study variants of shortest path queries in setting with failures.

The computation of shortest paths and distances in the presence of failures was extensively studied. Two central problems researched in this field are the Replacement Paths problem and Distance Sensitivity Oracles, we define these problems hereinafter.

The Replacement Paths problem. (See, e.g., [26, 28, 16, 14, 21, 27, 6, 30, 22, 23, 24, 25, 29, 15].) Let $G = (V, E)$ be a graph (directed or undirected, weighted or unweighted) with n vertices and m edges and let $P_G(s, t)$ be a shortest path from s to t. For every edge $e \in P_G(s, t)$ a replacement path $P_G(s, t, e)$ is a shortest path from s to t in the graph $G \setminus \{e\}$ (which is the graph G after removing the edge e). Let $d_G(s, t, e)$ be the length of the path $P_G(s, t, e)$. The replacement paths problem is as follows: given a shortest path $P_G(s, t)$ from s to t in G, compute $d_G(s, t, e)$ (or an approximation of it) for every $e \in P_G(s, t)$.

Distance Sensitivity Oracles. (See, e.g., [9, 17, 7, 8, 10, 11, 12, 13, 19].) An f-Sensitivity Distance Oracle (f-DSO) gets as input a graph $G = (V, E)$ and a parameter f, preprocesses it into a data-structure, such that given a query (s, t, F) with $s, t \in V$ and $F \subseteq E \cup V, |F| \leq f$ being a set of at most f edges or vertices (failures), the query algorithm efficiently computes (exactly or approximately) $d_G(s, t, F)$ which is the distance from s to t in the graph $G \setminus F$ (i.e., in the graph G after removing from it the failing edges and vertices F). Here we would like to optimize several parameters of the data-structure: minimize the size of the oracle, support many failures f, have efficient preprocessing and query algorithms, and if the output is an approximation of the distance then optimize the approximation-ratio.

An important line of research in the theory of computer science is derandomization. In many algorithms and data-structures there exists a gap between the best known randomized algorithms and the best known deterministic algorithms. There has been extensive research on closing the gaps between the best known randomized and deterministic algorithms in many problems or proving that no deterministic algorithm can perform as good as its randomized counterpart. There also has been a long line of work on developing derandomization techniques, in order to obtain deterministic versions of randomized algorithms (e.g., Chapter 16 in [2]).

In this paper we derandomize algorithms and data-structures for computing distances and shortest paths in the presence of failures. Many randomized algorithms for computing shortest paths and distances use variants of the following sampling lemma (see Lemma 1 in Roditty and Zwick [26]).

▶ **Lemma 1** (Lemma 1 in [26]). *Let $D_1, D_2, \ldots, D_q \subseteq V$ satisfy $|D_i| > L$ for $1 \leq i \leq q$ and $|V| = n$. If $R \subseteq V$ is a random subset obtained by selecting each vertex, independently, with probability $(c \ln n)/L$, for some $c > 0$, then with probability of at least $1 - q \cdot n^{-c}$ we have $D_i \cap R \neq \emptyset$ for every $1 \leq i \leq q$.*

Our derandomization step of Lemma 1 is very simple, as described in Section 1.3, we use the folklore greedy approach to prove the following lemma, which is a deterministic version of Lemma 1.

▶ **Lemma 2** (See also Section 1.3). *Let $D_1, D_2, \ldots, D_q \subseteq V$ satisfy $|D_i| > L$ for $1 \leq i \leq q$ and $|V| = n$. One can deterministically find in $\widetilde{O}(qL)$ time a set $R \subset V$ such that $|R| = \widetilde{O}(n/L)$ and $D_i \cap R \neq \emptyset$ for every $1 \leq i \leq q$.*

We emphasize that the use of Lemma 2 is very standard and is not our main contribution. The main technical challenge is how to efficiently and deterministically compute a small number of sets $D_1, D_2, \ldots, D_q \subseteq V$ so that the invocation of Lemma 2 is fast.

1.1 Derandomizing the Replacment Paths Algorithm of Roditty and Zwick [26]

We derandomize the algorithm of Roditty and Zwick [26] and obtain a near optimal deterministic algorithm for the replacement paths problem in directed unweighed graphs (a problem which was open for more than a decade since the randomized algorithm was published) as stated in the following theorem.

▶ **Theorem 3.** *There exists a deterministic algorithm for the replacement paths problem in unweighted directed graphs whose runtime is $\widetilde{O}(m\sqrt{n})$. This algorithm is near optimal assuming the conditional lower bound of combinatorial boolean matrix multiplication of [29].*

The term "combinatorial algorithms" is not well-defined, and it is often interpreted as non-Strassen-like algorithms [4], or more intuitively, algorithms that do not use any matrix multiplication tricks. Arguably, in practice, combinatorial algorithms are to some extent considered more efficient since the constants hidden in the matrix multiplication bounds are high. On the other hand, there has been research done to make fast matrix multiplication practical, *e.g.*, [18, 5].

Vassilevska Williams and Williams [29] proved a subcubic equivalence between $\sqrt{n}$ occurrences of the combinatorial replacement paths problem in unweighted directed graphs and the combinatorial boolean multiplication (BMM) problem. More precisely, they proved that there exists some fixed $\epsilon > 0$ such that the combinatorial replacement paths problem can be solved in $O(mn^{1/2-\epsilon})$ time if and only if there exists some fixed $\delta > 0$ such that the combinatorial boolean matrix multiplication (BMM) can be solved in subcubic $O(n^{3-\delta})$ time. Giving a subcubic combinatorial algorithm to the BMM problem, or proving that no such algorithm exists, is a long standing open problem. This implies that either both problems can be polynomially improved, or neither of them does. Hence, assuming the conditional lower bound of combinatorial BMM, our combinatorial $\widetilde{O}(m\sqrt{n})$ algorithm for the replacement paths problem in unweighted directed graphs is essentially optimal (up to $n^{o(1)}$ factors).

The replacement paths problem is related to the k simple shortest paths problem, where the goal is to find the k simple shortest paths between two vertices. Using known reductions from the replacement paths problem to the k simple shortest paths problem, we close this gap as the following Corollary states.

▶ **Corollary 4.** *There exists a deterministic algorithm for computing k simple shortest paths in unweighted directed graphs whose runtime is $\widetilde{O}(km\sqrt{n})$.*

The trivial $\widetilde{O}(mn)$ time algorithm for solving the replacement paths problem in directed weighted graphs (simply, for every edge $e \in P_G(s,t)$ run Dijkstra in the graph $G \setminus \{e\}$) is deterministic and near optimal (according to a conditional lower bound by [29]). To the best of our knowledge the only deterministic combinatorial algorithms known for directed unweighted graphs are the algorithms for general directed weighted graphs whose runtime is $\widetilde{O}(mn)$ leaving a significant gap between the randomized and deterministic algorithms. As mentioned above, in this paper we derandomize the $\widetilde{O}(m\sqrt{n})$ algorithm of Roditty and Zwick [26] and close this gap. More related work can be found in the full version.

1.2 Derandomizing the Combinatorial Distance Sensitivity Oracle of Weimann and Yuster [27]

Our second result is derandomizing the combinatorial distance sensitivity oracle of Weimann and Yuster [27] and obtaining the following theorem.

▶ **Theorem 5.** *Let $G = (V, E)$ be a directed graph with real edge weights, let $|V| = n$ and $|E| = m$. There exists a deterministic algorithm that given G and parameters $f = O(\frac{\log n}{\log \log n})$ and $0 < \alpha < 1$ constructs an f-sensitivity distance oracle in $\widetilde{O}(mn^{4-\alpha})$ time. Given a query (s, t, F) with $s, t \in V$ and $F \subseteq E \cup V, |F| \leq f$ being a set of at most f edges or vertices (failures), the deterministic query algorithm computes in $\widetilde{O}(n^{2-2(1-\alpha)/f})$ time the distance from s to t in the graph $G \setminus F$.*

We remark that while our focus in this paper is in computing distances, one may obtain the actual shortest path in time proportional to the number of edges of the shortest paths, using the same algorithm for obtaining the shortest paths in the replacement paths problem [26], and in the distance sensitivity oracles case [27].

1.3 Technical Contribution and Our Derandomization Framework

Let $\mathcal{A}$ be a random algorithm that uses Lemma 1 for sampling a subset of vertices $R \subseteq V$. We say that $\mathcal{P} = \{D_1, \ldots, D_q\}$ is a set of *critical* paths for the randomized algorithm $\mathcal{A}$ if $\mathcal{A}$ uses the sampling Lemma 1 and it is sufficient for the correctness of algorithm $\mathcal{A}$ that R is a hitting set for $\mathcal{P}$ (i.e., every path in $\mathcal{P}$ contains at least one vertex of R). According to Lemma 2 one can derandomize the random selection of the hitting set R in time that depends on the number of paths in $\mathcal{P}$. Therefore, in order to obtain an efficient derandomization procedure, we want to find a small set $\mathcal{P}$ of critical paths for the randomized algorithms.

Our main technical contribution is to show how to compute a small set of critical paths that is sufficient to be used as input for the greedy algorithm stated in Lemma 2.

Our framework for derandomizing algorithms and data-structures that use the sampling Lemma 1 is given in Figure 1.

1. **Step 1**: Prove the existence of a small set of critical paths $\{D_1, \ldots, D_q\}$ such that $|D_i| > L$ and show that it is sufficient for the correctness of the randomized algorithm that the set R obtained by Lemma 1 hits all the paths $D_1, \ldots, D_q$.
2. **Step 2**: Find an efficient algorithm to compute the paths $D_1, \ldots, D_q$.
3. **Step 3**: Use a deterministic algorithm to compute a small subset $R \subseteq V$ of vertices such that $D_i \cap R \neq \emptyset$ for every $1 \leq i \leq q$. For example, one can use the greedy algorithm of Lemma 2 or the blocker set algorithm of [20] to find a subset $R \subset V$ of $\widetilde{O}(n/L)$ vertices.

Figure 1 Our derandomization framework to derandomize algorithms that use the sampling Lemma 1.

Our first main technical contribution, denoted as Step 1 in Figure 1, is proving the existence of small sets of critical paths for the randomized replacement path algorithm of Roditty and Zwick [26] and for the distance sensitivity oracles of Weimann and Yuster [27]. Our second main technical contribution, denoted as Step 2 in Figure 1, is developing algorithms to efficiently compute these small sets of critical paths.

For the replacement paths problem, Roditty and Zwick [26] proved the existence of a critical set of $O(n^2)$ paths, each path containing at least $\lceil\sqrt{n}\rceil$ edges. Simply applying Lemma 2 on this set of paths requires $\widetilde{O}(n^{2.5})$ time which is too much, and it is also not clear from their algorithm how to efficiently compute this set of critical paths. As for Step 1, we prove the existence of a small set of $O(n)$ critical paths, each path contains $\lceil\sqrt{n}\rceil$ edges, and for Step 2, we develop an efficient algorithm that computes this set of critical paths in $\widetilde{O}(m\sqrt{n})$ time.

For the problem of distance sensitivity oracles, Weimann and Yuster [27] proved the existence of a critical set of $O(n^{2f+3})$ paths, each path containing $n^{(1-\alpha)/f}$ edges (where $0 < \alpha < 1$). Simply applying Lemma 2 on this set of paths requires $\widetilde{O}(n^{2f+3+(1-\alpha)/f})$ time which is too much, and here too, it is also not clear from their algorithm how to efficiently and deterministically compute this set of critical paths. As for Step 1, we prove the existence of a small set of $O(n^{2+\epsilon})$ critical paths, each path contains $n^{(1-\alpha)/f}$ edges, and for Step 2, we develop an efficient deterministic algorithm that computes this set of critical paths in $\widetilde{O}(mn^{1+\epsilon})$ time.

For Step 3, we use the folklore greedy deterministic algorithm denoted here by GreedyPivotsSelection($\{D_1, \ldots, D_q\}$). Given as input the paths $D_1, \ldots, D_q$, each path contains at least L vertices, the algorithm chooses a set of pivots $R \subseteq V$ such that for every $1 \leq i \leq q$ it holds that $D_i \cap R \neq \emptyset$. In addition, it holds that $|R| = \widetilde{O}(\frac{n}{L})$ and the runtime of the algorithm is $\widetilde{O}(qL)$.

The GreedyPivotsSelection algorithm works as follows. Let $\mathcal{P} = \{D_1, \ldots, D_q\}$. Starting with $R \leftarrow \emptyset$, find a vertex $v \in V$ which is contained in the maximum number of sets of $\mathcal{P}$, add it to R and remove all the sets that contain v from $\mathcal{P}$. Repeat this process until $\mathcal{P} = \emptyset$.

The following greedy selection lemma is folklore and we prove it in the full version.

▶ **Lemma 6.** *Let $1 \leq L \leq n$ and $1 \leq q < poly(n)$ be two integers. Let $D_1, \ldots, D_q \subseteq V$ be paths satisfying $|D_i| \geq L$ for every $1 \leq i \leq q$. The algorithm GreedyPivotsSelection($\{D_1, \ldots, D_q\}$) finds in $\widetilde{O}(qL)$ time a set $R \subset V$ such that for every $1 \leq i \leq q$ it holds that $R \cap D_i \neq \emptyset$ and $|R| = O(\frac{n \log q}{L}) = \widetilde{O}(n/L)$.*

Related Work - the Blocker Set Algorithm of King. We remark that the GreedyPivotsSelection algorithm is similar to the blocker set algorithm described in [20] for finding a hitting set for a set of paths. The blocker set algorithm was used in [20] to develop sequential dynamic algorithms for the APSP problem. Additional related work is that of Agarwal et al. [1]. They presented a deterministic distributed algorithm to compute APSP in an edge-weighted directed or undirected graph in $\widetilde{O}(n^{3/2})$ rounds in the Congest model by incorporating a deterministic distributed version of the blocker set algorithm.

While our derandomization framework uses the greedy algorithm (or the blocker set algorithm) to find a hitting set of vertices for a critical set of paths $D_1, \ldots, D_q$, we stress that our main contribution are the techniques to reduce the number of sets q the greedy algorithm must hit (Step 1), and the algorithms to efficiently compute the sets $D_1, \ldots, D_q$ (Step 2). These techniques are our main contribution, which enable us to use the greedy algorithm (or the blocker set algorithm) for a wider range of problems. Specifically, these techniques allow us to derandomize the best known random algorithms for the replacement paths problem and distance sensitivity oracles. We believe that our techniques can also be leveraged for additional related problems which use a sampling lemma similar to Lemma 1.

Outline. The structure of the paper is as follows. In Section 1.4 we describe some preliminaries and notations. In Section 2 we apply our framework to the replacement paths algorithm of Roditty and Zwick [26]. In Section 3 we apply our framework to the DSO of Weimann and Yuster for graphs with real-edge weights [27].

1.4 Preliminaries

Let $G = (V, E)$ be a directed weighted graph with n vertices and m edges with real edge weights $\omega(\cdot)$. Given a path P in G we define its weight $\omega(P) = \Sigma_{e \in E(P)} \omega(e)$.

Given $s, t \in V$, let $P_G(s,t)$ be a shortest path from s to t in G and let $d_G(s,t) = \omega(P_G(s,t))$ be its length, which is the sum of its edge weights. Let $|P_G(s,t)|$ denote the number of edges along $P_G(s,t)$. Note that for unweighted graphs we have $|P_G(s,t)| = d_G(s,t)$. When G is known from the context we sometimes abbreviate $P_G(s,t), d_G(s,t)$ with $P(s,t), d(s,t)$ respectively.

We define the path concatenation operator $\circ$ as follows. Let $P_1 = (x_1, x_2, \ldots, x_r)$ and $P_2 = (y_1, y_2, \ldots, y_t)$ be two paths. Then $P = P_1 \circ P_2$ is defined as the path $P = (x_1, x_2, \ldots, x_r, y_1, y_2, \ldots, y_t)$, and it is well defined if either $x_r = y_1$ or $(x_r, y_1) \in E$.

For a graph H we denote by $V(H)$ the set of its vertices, and by $E(H)$ the set of its edges. When it is clear from the context, we abbreviate $e \in E(H)$ by $e \in H$ and $v \in V(H)$ by $v \in H$.

Let P be a path which contains the vertices $u, v \in V(P)$ such that u appears before v along P. We denote by $P[u..v]$ the subpath of P from u to v.

For every edge $e \in P_G(s,t)$ a replacement path $P_G(s,t,e)$ for the triple (s,t,e) is a shortest path from s to t avoiding e. Let $d_G(s,t,e) = \omega(P_G(s,t,e))$ be the length of the replacement path $P_G(s,t,e)$.

We will assume, without loss of generality, that every replacement path $P_G(s,t,e)$ can be decomposed into a common prefix $\text{CommonPref}_{s,t,e}$ with the shortest path $P_G(s,t)$, a detour $\text{Detour}_{s,t,e}$ which is disjoint from the shortest path $P_G(s,t)$ (except for its first vertex and last vertex), and finally a common suffix $\text{CommonSuff}_{s,t,e}$ which is common with the shortest path $P_G(s,t)$. Therefore, for every edge $e \in P_G(s,t)$ it holds that $P_G(s,t,e) = \text{CommonPref}_{s,t,e} \circ \text{Detour}_{s,t,e} \circ \text{CommonSuff}_{s,t,e}$ (the prefix and/or suffix may be empty).

Let $F \subseteq V \cup E$ be a set of vertices and edges. We define the graph $G \setminus F = (V \setminus F, E \setminus F)$ as the graph obtained from G by removing the vertices and edges F. We define a replacement path $P_G(s,t,F)$ as a shortest path from s to t in the graph $G \setminus F$, and let $d_G(s,t,F) = w(P_G(s,t,e))$ be its length.

2 Deterministic Replacement Paths in $\widetilde{O}(m\sqrt{n})$ Time

In this section we apply our framework from Section 1.3 to the replacement paths algorithm of Roditty and Zwick [26].

The randomized algorithm by Roddity and Zwick as described in [26] takes $\widetilde{O}(m\sqrt{n})$ expected time. They handle separately the case that a replacement path has a short detour containing at most $\lceil \sqrt{n} \rceil$ edges, and the case that a replacement path has a long detour containing more than $\lceil \sqrt{n} \rceil$ edges. The first case is solved deterministically. The second case is solved by first sampling a subset of vertices R according to Lemma 1, where each vertex is sampled uniformly independently at random with probability $c \ln n / \sqrt{n}$ for large enough constant $c > 0$. Using this uniform sampling, it holds with high probability (of at least $1 - n^{-c+2}$) that for every long triple (s,t,e) (as defined hereinafter), the detour $\text{Detour}_{s,t,e}$ of the replacement path $P_G(s,t,e)$ contains at least one vertex of R.

▶ **Definition 7.** *Let $s,t \in V, e \in P_G(s,t)$. The triple (s,t,e) is a long triple if every replacement path from s to t avoiding e has its detour part containing more than $\lceil\sqrt{n}\rceil$ edges.*

Note that in Definition 7 we defined (s,t,e) to be a long triple if **every** replacement path from s to t avoiding e has a long detour (containing more than $\lceil\sqrt{n}\rceil$ edges). We could have defined (s,t,e) to be a long triple even if at least one replacement path from s to t avoiding e has a long detour (perhaps more similar to the definitions in [26]), however we find Definition 7 more convenient for the following reason. If (s,t,e) has a replacement path whose detour part contains at most $\lceil\sqrt{n}\rceil$ edges, then the algorithm of [26] for handling short detours finds deterministically a replacement path for (s,t,e). Hence, we only need to find the replacement paths for triples (s,t,e) for which every replacement path from s to t avoiding e has a long detour, and this is the case for which we define (s,t,e) as a long triple.

It is sufficient for the correctness of the replacement paths algorithm that the following condition holds; For every long triple (s,t,e) the detour $\text{Detour}_{s,t,e}$ of the replacement path $P_G(s,t,e)$ contains at least one vertex of R. As the authors of [26] write, the choice of the random set R is the only randomization used in their algorithm. To obtain a deterministic algorithm for the replacement paths problem and to prove Theorem 3, we prove the following deterministic alternative of Lemma 2.

▶ **Lemma 8** (Our derandomized version of Lemma 2 for the replacement paths algorithm)**.** *There exists an $\widetilde{O}(m\sqrt{n})$ time deterministic algorithm that computes a set $R \subseteq V$ of $\widetilde{O}(\sqrt{n})$ vertices, such that for every long triple (s,t,e) there exists a replacement path $P_G(s,t,e)$ whose detour part contains at least one of the vertices of R.*

Following the above description, in order to prove Theorem 3, that there exists an $\widetilde{O}(m\sqrt{n})$ deterministic replacement paths algorithm, it is sufficient to prove the derandomization Lemma 8, we do so in the following sections.

2.1 Step 1: the Method of Reusing Common Subpaths - Defining the Set $\mathcal{D}_n$

In this section we prove the following lemma.

▶ **Lemma 9.** *There exists a set $\mathcal{D}_n$ of at most n paths, each path of length exactly $\lceil\sqrt{n}\rceil$ with the following property; for every long triple (s,t,e) there exists a path $D \in \mathcal{D}_n$ and a replacement path $P_G(s,t,e)$ such that D is contained in the detour part of $P_G(s,t,e)$.*

In order to define the set of paths $\mathcal{D}_n$ and prove Lemma 9 we need the following definitions. Let $G' = G \setminus E(P_G(s,t))$ be the graph obtained by removing the edges of the path $P_G(s,t)$ from G. For two vertices u and v, let $d_{G'}(u,v)$ be the distance from u to v in G'.

We use the following definitions of the index $\rho(x)$, the set of vertices $V_{\sqrt{n}}$ and the set of paths $\mathcal{D}_n$.

▶ **Definition 10** (The index $\rho(x)$)**.** *Let $P_G(s,t) =< v_0,\ldots,v_k >$ and let $X = \{x \in V \mid \exists_{0\le i\le k}\ d_{G'}(v_i,x) = \lceil\sqrt{n}\rceil\}$ be the subset of all the vertices $x \in V$ such that there exists at least one index $0 \le i \le k$ with $d_{G'}(v_i,x) = \lceil\sqrt{n}\rceil$.*

For every vertex $x \in X$ we define the index $0 \le \rho(x) \le k$ to be the minimum index such that $d_{G'}(v_{\rho(x)},x) = \lceil\sqrt{n}\rceil$.

▶ **Definition 11** (The set of vertices $V_{\sqrt{n}}$)**.** *We define the set of vertices $V_{\sqrt{n}} = \{x \in X | \forall_{i<\rho(x)} d_{G'}(v_i,x) > \lceil\sqrt{n}\rceil\}$. In other words, $V_{\sqrt{n}}$ is the set of all vertices $x \in X$ such that for all the vertices v_i before $v_{\rho(x)}$ along $P_G(s,t)$ it holds that $d_{G'}(v_i,x) > \lceil\sqrt{n}\rceil$.*

▶ **Definition 12** (A set of paths $\mathcal{D}_n$). *For every vertex $x \in V_{\sqrt{n}}$, let $D(x)$ be an arbitrary shortest path from $v_{\rho(x)}$ to x in G' (whose length is $\lceil\sqrt{n}\rceil$ as $d_{G'}(v_{\rho(x)}, x) = \lceil\sqrt{n}\rceil$). We define $\mathcal{D}_n = \{D(x) | x \in V_{\sqrt{n}}\}$.*

Note that while $V_{\sqrt{n}}$ is uniquely defined (as it is defined according to distances between vertices) the set of paths $\mathcal{D}_n$ is not unique, as there may be many shortest paths from $v_{\rho(x)}$ to x in G', and we take $D(x) = P_{G'}(v_{\rho(x)}, x)$ to be an arbitrary such shortest path.

The basic intuition for the method of reusing common subpaths is as follows. Let $P_G(s,t,e_1), \ldots, P_G(s,t,e_r)$ be arbitrary replacement paths such that x is the $(\lceil\sqrt{n}\rceil + 1)^{\text{th}}$ vertex along the detours of all the replacement path $P_G(s,t,e_1), \ldots, P_G(s,t,e_r)$. Then one can construct replacement paths $P'_G(s,t,e_1), \ldots, P'_G(s,t,e_r)$ such that the subpath $D(x) \in \mathcal{D}_n$ is contained in all these replacement paths. Therefore, the subpath $D(x)$ is reused as a common subpath in many replacement paths. We utilize this observation in the following proof of Lemma 9.

Proof of Lemma 9. Obviously, the set $\mathcal{D}_n$ described in Definition 12 contains at most n paths, each path is of length exactly $\lceil\sqrt{n}\rceil$.

We prove that for every long triple (s,t,e) there exists a path $D \in \mathcal{D}_n$ and a replacement path $P'(s,t,e)$ *s.t.* D is contained in the detour part of $P'(s,t,e)$.

Let $P_G(s,t,e)$ be a replacement path for (s,t,e). Since (s,t,e) is a long triple then the detour part $\text{Detour}_{s,t,e}$ of $P_G(s,t,e)$ contains more than $\lceil\sqrt{n}\rceil$ edges. Let $x \in \text{Detour}_{s,t,e}$ be the $(\lceil\sqrt{n}\rceil + 1)^{\text{th}}$ vertex along $\text{Detour}_{s,t,e}$, and let v_j be the first vertex of $\text{Detour}_{s,t,e}$. Let P_1 be the subpath of $\text{Detour}_{s,t,e}$ from v_j to x and let P_2 be the subpath of $P_G(s,t,e)$ from x to t. In other words, $P_G(s,t,e) =< v_0, \ldots, v_j > \circ P_1 \circ P_2$. Since $\text{Detour}_{s,t,e}$ contains more than $\lceil\sqrt{n}\rceil$ edges and is disjoint from $P_G(s,t)$ except for the first and last vertices of $\text{Detour}_{s,t,e}$ and $P_1 \subset \text{Detour}_{s,t,e}$ it follows that P_1 is disjoint from $P_G(s,t)$ (except for the vertex v_j). In particular, since P_1 is a shortest path in $G \setminus \{e\}$ that is edge-disjoint from $P_G(s,t)$, then P_1 is also a shortest path in $G' = G \setminus E(P_G(s,t))$. We get that $d_{G'}(v_j, x) = |P_1| = \lceil\sqrt{n}\rceil$.

We prove that $j = \rho(x)$ and $x \in V_{\sqrt{n}}$. As we have already proved that $d_{G'}(v_j, x) = \lceil\sqrt{n}\rceil$, we need to prove that for every $0 \leq i < j$ it holds that $d_{G'}(v_i, x) > \lceil\sqrt{n}\rceil$. Assume by contradiction that there exists an index $0 \leq i < j$ such that $d_{G'}(v_i, x) \leq \lceil\sqrt{n}\rceil$. Then the path $\hat{P} =< v_0, \ldots, v_i > \circ P_{G'}(v_i, x) \circ P_2$ is a path from s to t that avoids e and its length is:

$$\begin{aligned} |\hat{P}| &= |< v_0, \ldots, v_i > \circ P_{G'}(v_i, x) \circ P_2| \\ &\leq i + \lceil\sqrt{n}\rceil + |P_2| \\ &< j + \lceil\sqrt{n}\rceil + |P_2| \\ &= |P_G(s, v_j) \circ P_1 \circ P_2| \\ &= |P_G(s,t,e)| \end{aligned}$$

This means that the path $\hat{P}$ is a path from s to t in $G \setminus \{e\}$ and its length is shorter than the length of the shortest path $P_G(s,t,e)$ from s to t in $G \setminus \{e\}$, which is a contradiction. We get that $d_{G'}(v_j, x) = \lceil\sqrt{n}\rceil$ and for every $0 \leq i < j$ it holds that $d_{G'}(v_i, x) > \lceil\sqrt{n}\rceil$. Therefore, according to Definitions 10 and 11 it holds that $j = \rho(x)$ and $x \in V_{\sqrt{n}}$.

Let $D(x) \in \mathcal{D}_n$, then according to Definition 12, $D(x)$ is a shortest path from $v_{\rho(x)}$ to x in G'. We define the path $P'(s,t,e) =< v_0, \ldots, v_{\rho(x)} > \circ D(x) \circ P_2$. It follows that $P'(s,t,e)$ is a path from s to t that avoids e and $|P'(s,t,e)| = |< v_0, \ldots, v_{\rho(x)} > \circ D(x) \circ P_2| = \rho(x) + \lceil\sqrt{n}\rceil + |P_2| = |P_G(s,t,e)| = d_G(s,t,e)$. Hence, $P'(s,t,e)$ is a replacement path for (s,t,e) such that $D(x) \subset P'(s,t,e)$ so the lemma follows. ◀

2.2 Step 2: the Method of Decremental Distances from a Path - Computing the Set $\mathcal{D}_n$

In this section we describe a decremental algorithm that enables us to compute the set of paths $\mathcal{D}_n$ in $\widetilde{O}(m\sqrt{n})$ time, proving the following lemma.

▶ **Lemma 13.** *There exists a deterministic algorithm for computing the set of paths $\mathcal{D}_n$ in $\widetilde{O}(m\sqrt{n})$ time.*

Our algorithm for computing the set of path $\mathcal{D}_n$ is a variant of the decremental SSSP (single source shortest paths) algorithm of King [20]. Our variant of the algorithm is used to find distances of vertices from a path rather than from a single source vertex as we define below.

Overview of the Deterministic Algorithm for Computing $\mathcal{D}_n$ in $\widetilde{O}(m\sqrt{n})$ Time. In the following description let $P = P_G(s,t)$. Consider the following assignment of weights ω to edges of G. We assign weight ϵ for every edge e on the path P, and weight 1 for all the other edges where ϵ is a small number such that $0 < \epsilon < 1/n$. We define a graph $G^w = (G, w)$ as the weighted graph G with edge weights ω. We define for every $0 \leq i \leq k$ the graph $G_i = G \setminus \{v_{i+1}, \ldots, v_k\}$ and the path $P_i = P \setminus \{v_{i+1}, \ldots, v_k\}$. We define the graph $G_i^w = (G_i, w)$ as the weighted graph G_i with edge weights ω.

The algorithm computes the graph G^w by simply taking G and setting all edge weights of $P_G(s,t)$ to be ϵ (for some small ϵ such that $\epsilon < 1/n$) and all other edge weights to be 1. The algorithm then removes the vertices of $P_G(s,t)$ from G^w one after the other (starting from the vertex that is closest to t). Loosely speaking after each vertex is removed, the algorithm computes the distances from s in the current graph. In each such iteration, the algorithm adds to $V^w_{\sqrt{n}}$ all vertices such that their distance from s in the current graph is between $\lceil\sqrt{n}\rceil$ and $\lceil\sqrt{n}\rceil + 1$. We will later show that at the end of the algorithm we have $V^w_{\sqrt{n}} = V_{\sqrt{n}}$. Unfortunately, we cannot afford running Dijkstra after the removal of every vertex of $P_G(s,t)$ as there might be n vertices on $P_G(s,t)$. To overcome this issue, the algorithm only maintains nodes at distance at most $\lceil\sqrt{n}\rceil + 1$ from s. In addition, we observe that to compute the SSSP from s in the graph after the removal of a vertex v_i we only need to spend time on nodes such that their shortest path from s uses the removed vertex. Roughly speaking, for these nodes we show that their distance from s rounded down to the closest integer must increase by at least 1 as a result of the removal of the vertex. Hence, for every node we spend time on it in at most $\lceil\sqrt{n}\rceil + 1$ iterations until its distance from s is bigger than $\lceil\sqrt{n}\rceil + 1$. As we will show later this will yield our desired running time.

In the full version we analyse the algorithm and prove Lemma 13.

Proof of Theorem 3. We summarize the $\widetilde{O}(m\sqrt{n})$ deterministic replacement paths algorithm and outline the proof of Theorem 3. First, compute in $\widetilde{O}(m\sqrt{n})$ time the set of paths $\mathcal{D}_n$ as in Lemma 13. Given $\mathcal{D}_n$, the deterministic greedy selection algorithm GreedyPivotsSelection($\mathcal{D}_n$) (as described in Lemma 2) computes a set $R \subset V$ of $\widetilde{O}(\sqrt{n})$ vertices in $\widetilde{O}(n\sqrt{n})$ time with the following property; every path $D \in \mathcal{D}_n$ contains at least one of the vertices of R. Theorem 3 follows from Lemmas 8, 9 and 13.

3 Deterministic Distance Sensitivity Oracles

Let $0 < \epsilon < 1$ and $1 \leq f = O(\frac{\log n}{\log\log n})$ be two parameters. In [27], Weimann and Yuster considered the following notion of intervals (note that in [27] they use a parameter $0 < \alpha < 1$ and we use a parameter $0 < \epsilon < 1$ such that $\epsilon = 1 - \alpha$). They define an interval of a long

simple path P as a subpath of P consisting of $n^{\epsilon/f}$ consecutive vertices, so every simple path induces less than n (overlapping) intervals. For every subset $F \subset E$ of at most f edges, and for every pair of vertices $u, v \in V$, let $P_G(u, v, F)$ be a shortest path from u to v in $G \setminus F$. The path $P_G(u, v, F)$ induces less than n (overlapping) intervals. The total number of possible intervals is less than $O(n^{2f+3})$ as each one of the (at most) $O(n^{2f+2})$ possible queries (u, v, F) corresponds to a shortest path $P_G(u, v, F)$ that induces less than n intervals.

▶ **Definition 14.** *Let $\mathcal{D}_f$ be defined as all the intervals (subpaths containing $n^{\epsilon/f}$ edges) of all the replacement paths $P_G(s, t, F)$ for every $s, t \in V, F \subseteq E \cup V$ with $|F| \leq f$.*

Weimann and Yuster apply Lemma 1 to find a set $R \subseteq V$ of $\widetilde{O}(n^{1-\epsilon/f})$ vertices that hit w.h.p. all the intervals $\mathcal{D}_f$. According to these bounds (that $\mathcal{D}_f$ contains $O(n^{2f+3})$ paths, each containing exactly $n^{\epsilon/f}$ edges) applying the greedy algorithm to obtain the set R deterministically according to Lemma 2 takes $\widetilde{O}(n^{2f+3+\epsilon/f})$ time, which is very inefficient.

In this section we assume that all weights are non-negative (so we can run Dijkstra's algorithm) and that shortest paths are unique, we justify these assumptions in the full version.

3.1 Step 1: the Method of Using Fault-Tolerant Trees to Significantly Reduce the Number of Intervals

In Lemma 15 we prove that the set of intervals $\mathcal{D}_f$ actually contains at most $O(n^{2+\epsilon})$ unique intervals, rather than the $O(n^{2f+3})$ naive upper bound mentioned above. From Lemmas 15 and 2 it follows that the GreedyPivotsSelection($\mathcal{D}_f$) finds in $\widetilde{O}(n^{2+\epsilon+\epsilon/f})$ time the subset $R \subseteq V$ of $\widetilde{O}(n^{1-\epsilon/f})$ vertices that hit all the intervals $\mathcal{D}_f$. In the full version we further reduce the time it takes for the greedy algorithm to compute the set of pivots R to $\widetilde{O}(n^{2+\epsilon})$.

▶ **Lemma 15.** $|\mathcal{D}_f| = O(n^{2+\epsilon})$.

In order to prove Lemma 15 we describe the fault-tolerant trees data-structure, which is a variant of the trees which appear in Appendix A of [9].

▶ **Definition 16.** *Let $P^L_G(s, t, F)$ be the shortest among the s-to-t paths in $G \setminus F$ that contain at most L edges and let $d^L_G(s, t, F) = \omega(P^L_G(s, t, F))$. In other words, $d^L_G(s, t, F) = \min\{\omega(P) \mid P \text{ is an } s-to-t \text{ path on at most } L \text{ edges}\}$. If there is no path from s to t in $G \setminus F$ containing at most L edges then we define $P^L_G(s, t, F) = \emptyset$ and $d^L_G(s, t, F) = \infty$. For $F = \emptyset$ we abbreviate $P^L_G(s, t, \emptyset) = P^L_G(s, t)$ as the shortest path from s to t that contains at most L edges, and $d^L_G(s, t) = d^L_G(s, t, \emptyset)$ as its length.*

Let $s, t \in V$ be vertices and let $L, f \geq 1$ be fixed integer parameters, we define the trees $FT^{L,f}(s, t)$ as follows.

- In the root of $FT^{L,f}(s, t)$ we store the path $P^L_G(s, t)$ (and its length $d^L_G(s, t)$), and also store the vertices and edges of $P^L_G(s, t)$ in a binary search tree $BST^L(s, t)$; If $P^L_G(s, t) = \emptyset$ then we terminate the construction of $FT^{L,f}(s, t)$.
- For every edge or vertex a_1 of $P^L_G(s, t)$ we recursively build a subtree $FT^{L,f}(s, t, a_1)$ as follows. Let $P^L_G(s, t, \{a_1\})$ be the shortest path from s to t that contains at most L edges in the graph $G \setminus \{a_1\}$. Then in the subtree $FT^{L,f}(s, t, a_1)$ we store the path $P^L_G(s, t, \{a_1\})$ (and its length $d^L_G(s, t, \{a_1\})$) and we also store the vertices and edges of $P^L_G(s, t, \{a_1\})$ in a binary search tree $BST^L(s, t, a_1)$; If $P^L_G(s, t, \{a_1\}) = \emptyset$ we terminate the construction of $FT^{L,f}(s, t, a_1)$. If $f > 1$ then for every vertex or edge a_2 in $P^L_G(s, t, \{a_1\})$ we recursively build the subtree $FT^{L,f}(s, t, a_1, a_2)$ as follows.

- For the recursive step, assume we want to construct the subtree $FT^{L,f}(s,t,a_1,\ldots,a_i)$. In the root of $FT^{L,f}(s,t,a_1,\ldots,a_i)$ we store the path $P^L_G(s,t,\{a_1,\ldots,a_i\})$ (and its length $d^L_G(s,t,\{a_1,\ldots,a_i\})$) and we also store the vertices and edges of $P^L_G(s,t,\{a_1,\ldots,a_i\})$ in a binary search tree $BST^L(s,t,a_1,\ldots,a_i)$. If $P^L_G(s,t,\{a_1,\ldots,a_i\})=\emptyset$ then we terminate the construction of $FT^{L,f}(s,t,a_1,\ldots,a_i)$. If $i<f$ then for every vertex or edge a_{i+1} in $P^L_G(s,t,\{a_1,\ldots,a_i\}))$ we recursively build the subtree $FT^{L,f}(s,t,a_1,\ldots,a_i,a_{i+1})$.

Observe that there are two conditions in which we terminate the recursive construction of $FT^{L,f}(s,t,a_1,\ldots,a_i)$:

- Either $i=f$ in which case $FT^{L,f}(s,t,a_1,\ldots,a_f)$ is a leaf node of $FT^{L,f}(s,t)$ and we store in the leaf node $FT^{L,f}(s,t,a_1,\ldots,a_f)$ the path $P^L_G(s,t,\{a_1,\ldots,a_f\})$.
- Or there is no path from s to t in $G\setminus\{a_1,\ldots,a_i\}$ that contains at most L edges and then $FT^{L,f}(s,t,a_1,\ldots,a_i)$ is a leaf vertex of $FT^{L,f}(s,v)$ and we store in it $P^L_G(s,t,\{a_1,\ldots,a_i\})=\emptyset$.

Querying the tree $FT^{L,f}(s,t)$. Given a query (s,t,F) such that $F\subset V\cup E$ with $|F|=f$ we would like to compute $d^L_G(s,t,F)$ using the tree $FT^{L,f}(s,t)$.

The query procedure is as follows. Let $P^L_G(s,t)$ be the path stored in the root of $FT^{L,f}(s,t)$ (if the root of $FT^{L,f}(s,t)$ contains $\emptyset$ then we output that $d^L_G(s,t,F)=\infty$). First we check if $P^L_G(s,t)\cap F=\emptyset$ by checking if any of the elements $a_1\in F$ appear in $BST^L(s,t)$ (which takes $O(\log L)$ time for each element $a_1\in F$). If $P^L_G(s,t)\cap F=\emptyset$ we output $d^L_G(s,t,F)=d^L_G(s,t)$ (as $P^L_G(s,t)$ does not contain any of the vertices or edges in F). Otherwise, let $a_1\in P^L_G(s,t)\cap F$.

We continue the search similarly in the subtree $FT^{L,f}(s,t,a_1)$ as follows. Let $P^L_G(s,t,\{a_1\})$ be the path stored in the root of $FT^{L,f}(s,t,a_1)$ (if the root of $FT^{L,f}(s,t,a_1)$ contains $\emptyset$ then we output that $d^L_G(s,t,F)=\infty$). First we check if $P^L_G(s,t,\{a_1\})\cap F=\emptyset$ by checking if any of the elements $a_2\in F$ appear in $BST^L(s,t,a_1)$ (which takes $O(\log L)$ time for each element $a_2\in F$). If $P^L_G(s,t,\{a_1\})\cap F=\emptyset$ we output $d^L_G(s,t,F)=d^L_G(s,t,\{a_1\})$ (as $P^L_G(s,t,\{a_1\})$ does not contain any of the vertices or edges in F). Otherwise, let $a_2\in P^L_G(s,t,\{a_1\})\cap F$. We continue the search similarly in the subtrees $FT^{L,f}(s,t,a_1,a_2)$, $FT^{L,f}(s,t,a_1,a_2,\ldots,a_i)$ until we either reach a leaf node which contains $\emptyset$ (and in this case we output that $d^L_G(s,t,F)=\infty$) or we find a path $P^L_G(s,t,\{a_1,\ldots,a_i\})$ such that $P^L_G(s,t,\{a_1,\ldots,a_i\})\cap F=\emptyset$ and then we output $d^L_G(s,t,F)=d^L_G(s,t,\{a_1,\ldots,a_i\})$.

In the full version we prove the following lemma.

▶ **Lemma 17.** *Given the tree $FT^{L,f}(s,t)$ and a set of failures $F\subset V\cup E$ with $|F|\le f$, the query procedure computes the distance $d^L_G(s,t,F)$ in $O(f^2\log L)$ time.*

We are now ready to prove lemma 15 asserting that $|\mathcal{D}_f|=O(n^{2+\epsilon})$.

Proof of Lemma 15. Let $L=n^{\epsilon/f}$ and let $\mathcal{D}$ be the set of all the unique shortest paths $P^L_G(s,t,\{a_1,\ldots,a_i\})$ stored in all the nodes of all the trees $\{FT^{L,f}(s,t)\}_{s,t\in V}$. Since the number of nodes in every tree $FT^{L,f}(s,t)$ is at most $L^f=(n^{\epsilon/f})^f=n^\epsilon$, and there are $O(n^2)$ trees (one tree for every pair of vertices $s,t\in V$) we get that the number of nodes in all the trees $\{FT^{L,f}(s,t)\}_{s,t\in V}$ is $O(n^{2+\epsilon})$ and hence $|\mathcal{D}|=O(n^{2+\epsilon})$.

We prove that $\mathcal{D}_f\subseteq\mathcal{D}$. By definition, $\mathcal{D}_f$ contains all the intervals (subpaths containing $n^{\epsilon/f}$ edges) of all the replacement paths $P_G(s,t,F)$ for every $s,t\in V, F\subseteq E\cup V$ with $|F|\le f$. Let $P\in\mathcal{D}_f$ be the unique shortest path, then P is a subpath containing $n^{\epsilon/f}$ edges of the replacement paths $P_G(s,t,F)$. Let u be the first vertex of P, and let v be the last vertex of P. Then P is a shortest path from u to v in $G\setminus F$, and since we assume

that the shortest paths our algorithms compute are unique then $P = P_G(u, v, F)$ is the unique shortest path from u to v in $G \setminus F$. Since P is assumed to be a path on exactly $L = n^{\epsilon/f}$ edges, then $P = P_G(u, v, F) = P_G^L(u, v, F)$. According to the query procedure in the tree $FT^{L,f}(u, v)$ and Lemma 17, if we query the tree $FT^{L,f}(u, v)$ with (u, v, F) then we reach a node $FT^{L,f}(u, v, a_1, \ldots, a_i)$ which contains the path $P_G^L(u, v, \{a_1, \ldots, a_i\})$ with $\{a_1, \ldots, a_i\} \subseteq F$ such that $P_G^L(u, v, \{a_1, \ldots, a_i\}) = P_G^L(u, v, F) = P$ is the shortest u-to-v path in $G \setminus F$. Hence, $P \in \mathcal{D}$ and thus $\mathcal{D}_f \subseteq \mathcal{D}$ and $|\mathcal{D}_f| \leq |\mathcal{D}| = O(n^{2+\epsilon})$ ◀

3.2 Step 2: Efficient Construction of the Fault-Tolerant Trees – Computing the Paths $\mathcal{D}_f$

Recall that we defined the trees $FT^{L,f}(u, v)$ with respect the parameters f (the maximum number of failures) and L (where we search for shortest paths among paths of at most L edges). The idea is to build the trees $FT^{L,f}(u, v)$ using dynamic programming having the trees $FT^{L-1,f}(u, v)$ with parameters $f, L-1$ as subproblems.

Assume we have already built the trees $FT^{i,f}(u, v)$, where $u, v \in V, 1 \leq i < L$, we describe how to build the trees $FT^{i+1,f}(u, v)$. Let (u, v, F) be a query for which we want to compute the distance $d^{i+1}(u, v, F)$ (as part of the construction of the tree $FT^{i+1,f}(u, v)$). Scan all the edges $(u, z) \in E$ and query the tree $FT^{i,f}(z, v)$ with the set F to find the distance $d^i(z, v, F)$. Querying the tree $FT^{i,f}(z, v)$ takes $O(f^2 \log i) = O(f^2 \log L)$ time as described in Lemma 17 (note that $f^2 \log L = \widetilde{O}(1)$ for $f \leq \log n$ as $L \leq n$), and we run $O(\text{out-degree(u)})$ such queries and take the minimum of the following equation.

$$d^{i+1}(u, v, F) = \min_z \{\omega(u, z) + d^i(z, v, F) \mid (u, z) \in E \;\; AND \;\; u, z, (u, z) \notin F\} \tag{1}$$

$$\text{parent}^{i+1}(u, v, F) = \arg\min_z \{\omega(u, z) + d^i(z, v, F) \mid (u, z) \in E \;\; AND \;\; u, z, (u, z) \notin F\} \tag{2}$$

Note that in Equation 1 we assume that for every vertex $u \in V$ it holds that G contains the self loops $(u, u) \in E$ such that $\omega(u, u) = 0$.

So the time to compute $d^{i+1}(u, v, F)$ is $\widetilde{O}(\text{out-degree(u)})$. Next, we describe how to reconstruct the path $P^{i+1}(u, v, F)$ in $O(L)$ additional time. We reconstruct the shortest path $P^{i+1}(u, v, F)$ by simply following the (at most L) parent pointers. In more details, let $z = \text{parent}^{i+1}(u, v, F)$ be the vertex defined according to Equation 2. We reconstruct the shortest path $P^{i+1}(u, v, F)$ by concatenating (u, z) with the shortest path $P^i(z, v, F)$ (which we reconstruct in the same way), thus we can reconstruct $P^{i+1}(u, v, F)$ edge by edge in constant time per edge, and hence it takes $O(L)$ time to reconstruct the path $P^{i+1}(u, v, F)$ that contains at most L edges.

The tree $FT^{i,f}(u, v)$ contains $i^f \leq L^f$ nodes, and thus all the trees $\{FT^{i,f}(u, v)\}$ for all $i \leq L, u, v \in V$ contain $O(n^2 L^{f+1})$ nodes together.

In each such node we compute the distance $d^i(u, v, \{a_1, \ldots, a_j\})$ in $\widetilde{O}(\text{out-degree(u)})$ time and reconstruct the path $P^i(u, v, \{a_1, \ldots, a_j\})$ in additional $O(L)$ time. Theretofore, computing all the distances $d^i(u, v, \{a_1, \ldots, a_j\})$ and all the paths $P^i(u, v, \{a_1, \ldots, a_j\})$ in all the nodes of all the trees $\{FT^{i,f}(u, v)\}_{u,v \in V, 1 \leq i \leq L}$ takes $\widetilde{O}(\sum_{i \leq L, u, v \in V} L^f(\text{out-degree(u)} + L)) = \widetilde{O}(mnL^{f+1} + n^2 L^{f+2})$ time. substituting $L = \widetilde{O}(n^{\epsilon/f})$ we get an algorithm to compute the trees $\{FT^{L,f}(u, v)\}_{u,v \in V}$ in $\widetilde{O}(mn^{1+\epsilon+\epsilon/f} + n^{2+\epsilon+2\epsilon/f})$ time.

This proves the following Lemma.

▶ **Lemma 18.** *One can deterministically construct the trees $FT^{L,f}(s, t)$ for every $s, t \in V$ in $\widetilde{O}(mn^{1+\epsilon+\epsilon/f} + n^{2+\epsilon+2\epsilon/f})$ time.*

In the full version we further reduce the runtime to $\widetilde{O}(mn^{1+\epsilon})$ by using dynamic programming only for computing the first $f-1$ levels of the trees $FT^{L,f}(s,t)$ and then applying Dijkstra in a sophisticated manner to compute the last layer of the trees $FT^{L,f}(s,t)$. In addition, we also boost-up the runtime of the greedy pivots selection algorithm from $\widetilde{O}(n^{2+\epsilon+\epsilon/f})$ to $\widetilde{O}(n^{2+\epsilon})$ time.

References

1. Udit Agarwal, Vijaya Ramachandran, Valerie King, and Matteo Pontecorvi. A Deterministic Distributed Algorithm for Exact Weighted All-Pairs Shortest Paths in Õ(n 3/2) Rounds. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 199–205, 2018.
2. N. Alon and J.H. Spencer. *The Probabilistic Method*. Fourth Edition. Wiley, 2016.
3. Noga Alon, Shiri Chechik, and Sarel Cohen. Deterministic combinatorial replacement paths and distance sensitivity oracles. *CoRR*, abs/1905.07483, 2019. `arXiv:1905.07483`.
4. Grey Ballard, James Demmel, Olga Holtz, and Oded Schwartz. Graph Expansion and Communication Costs of Fast Matrix Multiplication. *J. ACM*, 59(6):32:1–32:23, January 2013. `doi:10.1145/2395116.2395121`.
5. Austin R. Benson and Grey Ballard. A Framework for Practical Parallel Fast Matrix Multiplication. *SIGPLAN Not.*, 50(8):42–53, January 2015. `doi:10.1145/2858788.2688513`.
6. Aaron Bernstein. A Nearly Optimal Algorithm for Approximating Replacement Paths and K Shortest Simple Paths in General Graphs. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 742–755, 2010. URL: `http://dl.acm.org/citation.cfm?id=1873601.1873662`.
7. Aaron Bernstein and David Karger. Improved Distance Sensitivity Oracles via Random Sampling. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, pages 34–43, 2008. URL: `http://dl.acm.org/citation.cfm?id=1347082.1347087`.
8. Aaron Bernstein and David Karger. A Nearly Optimal Oracle for Avoiding Failed Vertices and Edges. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing (STOC)*, pages 101–110, 2009. `doi:10.1145/1536414.1536431`.
9. Shiri Chechik, Sarel Cohen, Amos Fiat, and Haim Kaplan. $(1+\epsilon)$ approximate f-sensitive Distance Oracles. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 1479–1496, 2017. URL: `http://dl.acm.org/citation.cfm?id=3039686.3039782`.
10. Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. f-Sensitivity Distance Oracles and Routing Schemes. *Algorithmica*, 63(4):861–882, 2012. `doi:10.1007/s00453-011-9543-0`.
11. Camil Demetrescu and Mikkel Thorup. Oracles for Distances Avoiding a Link-failure. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, pages 838–843, 2002. URL: `http://dl.acm.org/citation.cfm?id=545381.545490`.
12. Camil Demetrescu, Mikkel Thorup, Rezaul Alam Chowdhury, and Vijaya Ramachandran. Oracles for Distances Avoiding a Failed Node or Link. *SIAM J. Comput.*, 37(5):1299–1318, January 2008. `doi:10.1137/S0097539705429847`.
13. Ran Duan and Seth Pettie. Dual-failure Distance and Connectivity Oracles. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, pages 506–515, 2009. URL: `http://dl.acm.org/citation.cfm?id=1496770.1496826`.
14. Yuval Emek, David Peleg, and Liam Roditty. A Near-linear-time Algorithm for Computing Replacement Paths in Planar Directed Graphs. *ACM Trans. Algorithms*, 6(4):64:1–64:13, September 2010. Appeared also in the Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '08). `doi:10.1145/1824777.1824784`.
15. David Eppstein. Finding the k Shortest Paths. *SIAM J. Comput.*, 28(2):652–673, 1998. `doi:10.1137/S0097539795290477`.

16 Zvi Gotthilf and Moshe Lewenstein. Improved Algorithms for the K Simple Shortest Paths and the Replacement Paths Problems. *Inf. Process. Lett.*, 109(7):352–355, March 2009. doi:10.1016/j.ipl.2008.12.015.

17 Fabrizio Grandoni and Virginia Vassilevska Williams. Improved Distance Sensitivity Oracles via Fast Single-Source Replacement Paths. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, 20-23, 2012*, pages 748–757, 2012. doi:10.1109/FOCS.2012.17.

18 J. Huang, L. Rice, D. A. Matthews, and R. A. v. d. Geijn. Generating Families of Practical Fast Matrix Multiplication Algorithms. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 656–667, May 2017. doi:10.1109/IPDPS.2017.56.

19 Neelesh Khanna and Surender Baswana. Approximate Shortest Paths Avoiding a Failed Vertex: Optimal Size Data Structures for Unweighted Graphs. In *27th International Symposium on Theoretical Aspects of Computer Science, STACS*, pages 513–524, 2010. doi:10.4230/LIPIcs.STACS.2010.2481.

20 Valerie King. Fully Dynamic Algorithms for Maintaining All-Pairs Shortest Paths and Transitive Closure in Digraphs. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 81–91, 1999. doi:10.1109/SFFCS.1999.814580.

21 Philip N. Klein, Shay Mozes, and Oren Weimann. Shortest Paths in Directed Planar Graphs with Negative Lengths: A Linear-space $O(n \log^2 n)$-time Algorithm. *ACM Trans. Algorithms*, 6(2):30:1–30:18, April 2010. doi:10.1145/1721837.1721846.

22 Eugene L. Lawler. A Procedure for Computing the K Best Solutions to Discrete Optimization Problems and Its Application to the Shortest Path Problem. *Management Science*, 18(7):401–405, 1972. doi:10.1287/mnsc.18.7.401.

23 Cheng-Wei Lee and Hsueh-I Lu. Replacement Paths via Row Minima of Concise Matrices. *SIAM J. Discrete Math.*, 28(1):206–225, 2014. doi:10.1137/120897146.

24 K. Malik, A. K. Mittal, and S. K. Gupta. The K Most Vital Arcs in the Shortest Path Problem. *Oper. Res. Lett.*, 8(4):223–227, August 1989. doi:10.1016/0167-6377(89)90065-5.

25 Enrico Nardelli, Guido Proietti, and Peter Widmayer. A Faster Computation of the Most Vital Edge of a Shortest Path. *Inf. Process. Lett.*, 79(2):81–85, June 2001. doi:10.1016/S0020-0190(00)00175-7.

26 Liam Roditty and Uri Zwick. Replacement Paths and k Simple Shortest Paths in Unweighted Directed Graphs. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP, 2005*, pages 249–260. See also ACM Trans. Algorithms, 8(4):33:1–11, 2012, 2005. doi:10.1007/11523468_21.

27 Oren Weimann and Raphael Yuster. Replacement Paths and Distance Sensitivity Oracles via Fast Matrix Multiplication. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 655–662, Washington, DC, USA, 2010. IEEE Computer Society. doi:10.1109/FOCS.2010.68.

28 Virginia Vassilevska Williams. Faster Replacement Paths. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, 23-25, 2011*, pages 1337–1346, 2011. doi:10.1137/1.9781611973082.102.

29 Virginia Vassilevska Williams and Ryan Williams. Subcubic Equivalences between Path, Matrix and Triangle Problems. In *51st Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010*, pages 645–654, 2010. doi:10.1109/FOCS.2010.67.

30 Jin Y. Yen. Finding the K Shortest Loopless Paths in a Network. *Management Science*, 17(11):712–716, 1971. doi:10.1287/mnsc.17.11.712.

Algorithms and Hardness for Diameter in Dynamic Graphs

Bertie Ancona
MIT, Cambridge, MA, USA
bancona@mit.edu

Monika Henzinger
University of Vienna, Austria
monika.henzinger@univie.ac.at

Liam Roditty
Bar Ilan University, Ramat Gan, Israel
liam.roditty@biu.ac.il

Virginia Vassilevska Williams
MIT, Cambridge, MA, USA
virgi@mit.edu

Nicole Wein
MIT, Cambridge, MA, USA
nwein@mit.edu

Abstract

The diameter, radius and eccentricities are natural graph parameters. While these problems have been studied extensively, there are no known *dynamic* algorithms for them beyond the ones that follow from trivial recomputation after each update or from solving dynamic All-Pairs Shortest Paths (APSP), which is very computationally intensive. This is the situation for dynamic *approximation* algorithms as well, and even if only edge insertions or edge deletions need to be supported.

This paper provides a comprehensive study of the dynamic approximation of Diameter, Radius and Eccentricities, providing both conditional lower bounds, and new algorithms whose bounds are optimal under popular hypotheses in fine-grained complexity. Some of the highlights include:

- Under popular hardness hypotheses, there can be no significantly better *fully dynamic approximation* algorithms than recomputing the answer after each update, or maintaining full APSP.
- Nearly optimal *partially dynamic* (incremental/decremental) algorithms can be achieved via efficient reductions to (incremental/decremental) maintenance of Single-Source Shortest Paths. For instance, a nearly $(3/2+\epsilon)$-approximation to Diameter in directed or undirected n-vertex, m-edge graphs can be maintained decrementally in total time $m^{1+o(1)}\sqrt{n}/\epsilon^2$. This nearly matches the static 3/2-approximation algorithm for the problem that is known to be conditionally optimal.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases fine-grained complexity, graph algorithms, dynamic algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.13

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1811.12527`.

Funding *Monika Henzinger*: The research leading to these results has received funding from the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement No. 340506.
Virginia Vassilevska Williams: Supported by an NSF CAREER Award, NSF Grants CCF-1417238, CCF-1528078 and CCF-1514339, a BSF Grant BSF:2012338 and a Sloan Research Fellowship.
Nicole Wein: Supported by an NSF Graduate Fellowship and NSF Grant CCF-1514339.

Acknowledgements The authors would like to thank Roei Tov for discussions.

1 Introduction

Computing the shortest paths distances between all pairs of vertices in a graph, the All-Pairs Shortest Paths (APSP) problem, has been studied since the beginning of computer science as a field. Nevertheless, the fastest known algorithms [26, 20, 19] for APSP in n-vertex m-edge graphs are only slightly faster (by $n^{o(1)}$ factors) than the simple $\tilde{O}(mn)$ time[1] algorithm running Dijkstra's algorithm from every vertex[2]. For dense graphs with small integer weights there are improved algorithms [22, 28] using fast matrix multiplication [23, 17], but these algorithms are not faster than mn for sparser graphs or when the weights can be large.

There are many important graph parameters that can be easily computed when all the distances are known. These include the *eccentricity* of each vertex (the maximum length of a shortest path from the vertex to another vertex), the graph *diameter* (the maximum over all eccentricities), the *radius* (the minimum over all eccentricities) and many more. These parameters are of particular importance in the analysis of social networks (e.g. [3]), but also in graphs generated for entities such as images and search queries (and web pages).

Unfortunately, there are no significantly faster algorithms to compute these parameters than just solving APSP, and this is far from practical. In many cases the analyzed networks are so large that even enumerating all pairs of vertices is prohibitively expensive. Thus, obtaining all pairwise distances is essentially impossible. For graph parameters, on the other hand, the output is a single number; in principle looking at all vertex pairs might not be necessary, and subquadratic time algorithms (in the number of vertices) might exist for sparse graphs (whereas quadratic time is necessary for APSP as this is the size of the output). The existence of such fast algorithms is an important, practically motivated question.

In recent years, much progress has been made in understanding the complexity of graph parameter computation. Results from fine-grained complexity give that even obtaining a $(3/2-\epsilon)$-approximation for graph diameter [21] or radius [2], or a $(5/3-\epsilon)$-approximation of all eccentricities [8] (for $\epsilon > 0$) requires $n^{2-o(1)}$ time even in very sparse graphs, assuming the Strong Exponential Time Hypothesis (SETH) and related conjectures. Even stronger hardness results were obtained by Backurs et al. [6], altogether showing that most of the known algorithms for diameter, radius and eccentricities are conditionally optimal.

In addition to computing graph parameters in a static graph, a very natural goal is to maintain estimates of these parameters in a *dynamic* graph, where edges are inserted and deleted. In this setting, we would like to have a fast algorithm which preprocesses the given graph, and builds a data structure which can support edge updates efficiently and can answer queries about the parameter of interest in the current state of the data structure. This dynamic version of the problems is even more practically motivated, as real networks are naturally dynamic.

Unfortunately, the state of the art of dynamic algorithms for graph parameters such as Diameter is somewhat disappointing. The best known dynamic algorithms either just use the best known dynamic algorithms for APSP, or recompute the parameter estimate from scratch after each update. This leads to the following bounds:

(1) Demetrescu and Italiano [10] obtained a fully dynamic exact APSP algorithm with an amortized update time of $\tilde{O}(n^2)$ and $O(1)$ query time; this is the best exact dynamic algorithm for the graph parameters as well. Abboud and Vassilevska W. [1] showed that under SETH, any $(4/3-\epsilon)$-approximation fully dynamic algorithm for diameter (for $\epsilon > 0$) requires

[1] $\tilde{O}$ notation supresses polylogarithmic factors.
[2] after Johnson's trick to make the weights nonnegative

$n^{2-o(1)}$ amortized update or query time even in sparse graphs. Thus the APSP approach is conditionally optimal for fully dynamic $(4/3-\epsilon)$-approximate diameter algorithms. It is unclear however whether a 4/3-approximation with better update time is possible, and whether the APSP bounds are best for Radius.

(2) By recomputing the parameter estimates after each query, one can maintain a 2-approximation for Diameter in directed graphs and Radius in undirected graphs in worst case $O(m+n)$ time per update, and a $(2+\epsilon)$-approximation for all Eccentricities in directed graphs for all ϵ in $\tilde{O}(m/\epsilon)$ time per update using the algorithm of Backurs et al. [6]. One can also maintain a 3/2-approximation for Diameter and Radius, and a 5/3-approximation of all Eccentricities in worst case time $\tilde{O}(m^{3/2})$ per update using the algorithms of [21, 8] . More algorithms follow from the static results of Cairo, Grossi and Rizzi [7]. Can any of these algorithms be improved or are they conditionally optimal? The only related lower bounds here are (a) by Henzinger et al. [14] which showed that under the Online Matrix Vector hypothesis (OMv), any fully dynamic Diameter algorithm that achieves a $(2-\epsilon)$-approximation for undirected *weighted* graphs, or any finite approximation in directed graphs needs $n^{0.5-o(1)}$ amortized update time and (b) by Henzinger et al. [15] which proved under the combinatorial Boolean Matrix Multiplication conjecture any fully dynamic Diameter or Eccentricity algorithm that achieves a $(4/3-\epsilon)$-approximation in undirected *unweighted* graphs with $n^{3-o(1)}$ preprocessing time requires $n^{2-o(1)}$ update or query time (and the same result for undirected *weighted* graph using the APSP conjecture). While these results give some limitation, they are far from tight.

The first contributions of our paper are strong conditional lower bounds for fully dynamic graph parameter estimation. Our first result is a strengthening of the conditional lower bound for Diameter of [1]: we increase the approximation ratio from $(4/3-\epsilon)$ to $(3/2-\epsilon)$.

▶ **Theorem 1.** *Under SETH, every fully dynamic $(3/2-\epsilon)$-approximation algorithm for Diameter with polynomial preprocessing time requires $n^{2-o(1)}$ amortized update or query time in the word-RAM model of computation with $O(\log n)$ bit words, even for dynamic undirected unweighted graphs that are always sparse.*

The same limitation applies for fully dynamic $(5/3-\epsilon)$-approximation algorithms for Eccentricities with polynomial preprocessing time, and for fully dynamic $(3/2-\epsilon)$-approximation algorithms for Radius with polynomial preprocessing time, under the related Hitting Set hypothesis.

These conditional lower bounds imply that the $\tilde{O}(m^{3/2})$ time estimation algorithms that recompute the answer from scratch are optimal in the sense that any improvement of the approximation factor causes the update time to grow to n^2, and Demetrescu and Italiano's algorithm achieves $\tilde{O}(n^2)$ update time even for the exact maintenance of APSP.

We also show that recomputing a 2-approximation from scratch in linear time is close to optimal under SETH.

▶ **Theorem 2.** *Under SETH, any fully dynamic algorithm with polynomial preprocessing time that can maintain for $\epsilon > 0$ any of the following in an n node, m-edge undirected unweighted graph requires either $m^{1-o(1)}$ amortized update or query time, even when $m = \tilde{O}(n)$:*

- *a $(2-\epsilon)$-approximation of the eccentricity of a fixed vertex, or*
- *a $(2-\epsilon)$-approximation of the Radius, or*
- *a $(2-\epsilon)$-approximation of the Diameter.*

This result significantly strengthens the OMv based lower bound of [14]: the update time lower bound is now linear as opposed to $\sqrt{n}$, the result holds for unweighted graphs as well, it also holds for Radius and single-node eccentricity, and it has implications for partially dynamic algorithms with worst case time bounds, unlike the one of [14]. Furthermore, the lower bound for the eccentricity of a fixed vertex is tight in the sense that one can simply recompute the answer exactly from scratch in time $O(m)$.

Much stronger lower bounds are possible for *directed* graphs. Our first hardness result for directed graphs is that under SETH and the Hitting Set hypothesis, respectively, nearly quadratic time is needed for Eccentricities and Radius, even for $(2-\epsilon)$-approximation. (Recall that for undirected graphs we could only show this for $(3/2-\epsilon)$-approximate Radius and $(5/3-\epsilon)$-approximate Eccentricities.) This means that for directed graphs, recomputing a 2-approximation from scratch (in linear time) after each update is very much optimal – for any better approximation one might as well use the exact dynamic APSP algorithms.

▶ **Theorem 3.** *Every fully dynamic algorithm with polynomial preprocessing time for $(2-\epsilon)$-approximate (for $\epsilon > 0$) Eccentricities (under SETH) or Radius (under a version of the Hitting Set Hypothesis) in directed, unweighted graphs with n vertices and $m = \tilde{O}(n)$ edges requires amortized update or query time $m^{2-o(1)}$.*

Surprisingly, we also show conditionally that no *finite* approximation can be maintained in *sublinear* time. Henzinger et al. [14], building upon [1], showed that any finite approximation for Diameter in directed graphs requires $m^{0.5-o(1)}$ time under the OMv Hypothesis. We strengthen the lower bound to linear, using a very natural hypothesis on the complexity of k-Cycle.

All known algorithms for detecting k-cycles in sparse directed graphs with m edges run at best in time $m^{2-c/k}$ for various small constants c [27, 4, 9], even if you use powerful tools such as fast matrix multiplication. A natural hypothesis completely consistent with the state of the art of cycle detection is that one needs $m^{2-f(k)-o(1)}$ time to find a k-cycle, for some continuous (over the reals) $f(k)$ that goes to 0 as k goes to infinity. Let us call this the k-Cycle Hypothesis. We obtain:

▶ **Theorem 4.** *Under the k-Cycle Hypothesis, any fully dynamic algorithm with polynomial preprocessing time that can maintain a* finite *approximation for any of the following in an n node, $m = \tilde{O}(n)$-edge directed unweighted graph requires either $m^{1-o(1)}$ amortized update or query time:*

- *the eccentricity of a fixed vertex, or*
- *the Radius, or*
- *the Diameter.*

All known approaches to estimating graph proximity parameters such as the Diameter, at the very least require maintaining approximate distances from a single node, up to some distance. The conditional lower bounds above say that even if we only want to maintain an estimate of the largest distance from a fixed node, and even if that distance is never more than a constant, we still need linear update time. Thus, to have better than 2 approximations of our undirected distance parameters or any finite approximation in the directed case that can be maintained in sublinear time we probably need to abandon our need for fully dynamic algorithms. We thus turn to *partially* dynamic algorithms that handle either only edge insertions (incremental) or only edge deletions (decremental).

Our conditional lower bounds for the fully dynamic setting also apply to incremental and decremental algorithms that have *worst case* update and query time guarantees. This is due to the nature of our reductions: they all produce an initial graph on which we perform update

stages that only insert or only delete (we can choose which) a small batch of edges, ask a query and undo the changes just made, returning to the initial graph. An incremental/decremental algorithm can be used to implement such reductions by performing the deletions/insertions by rolling back the data structure. Because of this, we have very strong worst case lower bounds, and it makes sense to focus on amortized guarantees instead.

The strong conditional lower bounds in the static case, imply strong limitations for partially dynamic algorithms as well. For *undirected* graphs, these limitations are as follows: Due to [21, 6, 8], under SETH, every incremental and decremental algorithm for Diameter in n node undirected unweighted graphs requires total time at least $m^{3/2-o(1)}$ to maintain a $(8/5-\epsilon)$-approximation, and at least $m^{2-o(1)}$ total time to maintain a $(3/2-\epsilon)$-approximation for $\epsilon > 0$ under $m = \tilde{O}(n)$ insertions or deletions. For Eccentricities, the static conditional lower bounds are slightly stronger. For partially dynamic algorithms they imply that under SETH, for every $k \geq 1$, maintaining a $((3k+2)/(k+2)-\epsilon)$-approximation for $\epsilon > 0$ requires total time $m^{1+1/k}$ for $m = \tilde{O}(n)$ insertions or deletions. For Radius, they just imply that under the Hitting Set hypothesis [24, 2] maintaining a $(3/2-\epsilon)$-approximation requires total time $m^{2-o(1)}$ even in a sparse graph.

For *directed* graphs, there are stronger lower bounds: maintaining a $(2-\epsilon)$-approximation for Radius and Eccentricities requires total time $m^{2-o(1)}$ under Hitting Set and SETH, respectively [2, 6].

The incremental lower bounds directly follow from the static ones by starting from an empty graph and inserting edges until we reach the graph from the static construction. The decremental lower bounds hold since the static lower bound instances are all subgraphs of the same global graph, independent of the SAT/Hitting Set instance that the reduction is trying to solve; thus we start with the global graph and delete edges until reaching the graph from the static construction.

A natural question is, are these conditional lower bounds tight? Can one create partially dynamic algorithms that can achieve the same total runtime as the known static approximation algorithms? We give positive answers to these questions by developing new partially dynamic algorithms that are essentially optimal. Our algorithms are actually very efficient reductions to incremental and decremental single source shortest paths (SSSP), so that any improvement over dynamic SSSP would improve our parameter estimation algorithms.

Let D_0 and D_f be the initial and final values of the diameter, respectively. Let $T_{inc}(n, m, k, \epsilon)$ (resp., $T_{dec}(n, m, k, \epsilon)$) be the total time of an incremental (decremental) approximate SSSP algorithm from source u that maintains an estimate $d'(u, v)$ for all v such that if $d(u, v) \leq k$ then $(1-\epsilon)d(u, v) \leq d'(u, v) \leq d(u, v)$. For directed graphs we assume that the approximate SSSP algorithm works in directed graphs, and for undirected graphs, the SSSP algorithm only needs to work in undirected graphs. Our black-box reductions can be summarized in the theorem below.

▶ **Theorem 5.** *There is a Las Vegas randomized algorithm for incremental (resp., decremental) diameter in unweighted, directed graphs against an oblivious (resp., adaptive) adversary that given $\epsilon > 0$, runs in total time $\tilde{O}(\max_{D_f \leq D' \leq D_0}\{T_{inc}(n, m, D', \epsilon)\frac{\sqrt{n/D'}}{\epsilon^2}\})$ (resp., $\tilde{O}(\max_{D_0 \leq D' \leq D_f}\{T_{dec}(n, m, D', \epsilon)\frac{\sqrt{n/D'}}{\epsilon^2}\})$) with high probability, and maintains an estimate $\hat{D}$ such that $\frac{2(1-\epsilon)}{3}D - \frac{2}{3} \leq \hat{D} \leq D$ where D is the diameter of the current graph.*

We obtain similar black-box reductions for nearly $(5/3+\epsilon)$-approximate Eccentricities and $(3/2+\epsilon)$-approximate Radius in undirected graphs.

Henzinger et al. [13] obtained a randomized $(1+\epsilon)$-approximate decremental algorithm for SSSP in undirected unweighted graphs against an oblivious adversary with total expected update time $m^{1+o(1/\epsilon)}$. As an immediate corollary we obtain:

▶ **Corollary 6.** *There is a Las Vegas randomized algorithm for decremental diameter in unweighted, undirected graphs against an oblivious adversary that given $\epsilon > 0$, runs in total time $m^{1+o(1/\epsilon)}\sqrt{n}/\epsilon^2$ in expectation, and maintains an estimate $\frac{2(1-\epsilon)}{3}D - \frac{2}{3} \leq \hat{D} \leq D$, where D is the diameter of the current graph.*

Due to lower bounds in the static setting described above, this result is conditionally optimal in terms of both running time and approximation factor except for a small loss in the approximation factor. A similar result hold for decremental undirected Radius with a conditionally essentially optimal approximation factor. A similar result also holds for Eccentricities, which is conditionally essentially optimal in terms of both running time and approximation factor.

For decremental algorithms in directed graphs and for incremental algorithms in undirected or directed graphs, the best known algorithms for SSSP up to distance k are achieved by the Even and Shiloach Trees data structure [11], giving amortized update time $O(k)$. Henzinger and King recognized that this data structure can be extended to directed graphs [12]. As a corollary we obtain:

▶ **Corollary 7.** *There is a Las Vegas randomized algorithm for incremental/decremental diameter in unweighted, directed graphs that given $\epsilon > 0$, runs in total time $\tilde{O}(m\sqrt{nD_{\max}}/\epsilon^2)$ with high probability where $D_{\max}$ is the maximum diameter throughout the algorithm, and maintains an estimate $\hat{D}$ such that $\frac{2(1-\epsilon)}{3}D - 1 \leq \hat{D} \leq D$, where D is the diameter of the current graph. The incremental algorithm works against an oblivious adversary and the decremental algorithm works against an adaptive adversary.*

Similar results hold for Radius and Eccentricities but only for undirected graphs. Recall that static conditional lower bounds rule out such algorithms in directed graphs.

The algorithms so far are all randomized. We present some deterministic incremental algorithms as well, again via a reduction to incremental SSSP. Let D_0, D_f and $T_{inc}(n, m, k, \epsilon)$ be as before.

▶ **Theorem 8.** *There is a deterministic algorithm for incremental diameter in unweighted, directed graphs that, for any ϵ with $0 < \epsilon < 2$, runs in total time $\tilde{O}(\max_{D_f \leq D' \leq D_0}\{(T_{inc}(n, m, D', \epsilon) + m)n/(\epsilon^2 D')\})$, and maintains an estimate $\hat{D}$ such that $(1-\epsilon)D \leq \hat{D} \leq D$, where D is the diameter of the current graph.*

Using Even and Shiloach trees we obtain as a corollary a deterministic incremental $(1+\epsilon)$-approximation algorithm for diameter with total update time $\tilde{O}(mn/\epsilon^2)$. The running time is essentially tight for $\epsilon < 1/2$ by the SETH based quadratic lower bound for $(3/2-\delta)$-approximate static diameter [21]. Similar algorithms with essentially tight running times hold for radius in directed graphs and eccentricities in directed, strongly connected graphs.

1.1 Our techniques for partially dynamic algorithms

Our partially dynamic nearly 3/2-approximation algorithms for diameter and radius and our nearly 5/3-approximation algorithm for eccentricities are based on known algorithms in the static setting [21, 8]. These static algorithms work by carefully choosing a set U of vertices, performing SSSP from every vertex in U, and showing that at least one of these SSSP instantiations yields a good estimate for the parameter of interest. The set U is chosen as follows. We pick a random sample S of $\tilde{\Theta}(\sqrt{n})$ vertices and let w^* be the vertex that is farthest from S; that is, w^* is the vertex that maximizes $\min_{s \in S} d(w^*, s)$. Then, letting $N(w, \sqrt{n})$ be the closest $\sqrt{n}$ vertices to w, we set $U = S \cup \{w^*\} \cup N(w^*, \sqrt{n})$.

Adapting these static algorithms to the dynamic setting presents two main challenges:

(1) Firstly, given a set S of vertices, the farthest vertex w^* from S can change over time. We wish to minimize the total number of vertices that we ever run dynamic SSSP from, as reinitializing dynamic SSSP from a new vertex is expensive. Suppose we run dynamic SSSP from every vertex in $N(w^*, \sqrt{n})$ at all times. Then, every time w^* changes, we must reinitialize the dynamic SSSP data structure from $\sqrt{n}$ new vertices. If w^* changes frequently, this is prohibitively slow. To overcome this issue, we show that it suffices to choose a vertex w that *approximates* w^* (for a careful notion of approximation); and furthermore, by doing so we can limit the number of times we choose a new w.

Due to inherent differences between the incremental and decremental settings, we choose w in different ways in the different settings. In the decremental setting, distances can only increase, so our current choice of w can only become a poor approximation for w^* if $d(w^*, S)$ increases. Then, we use the fact that $d(w^*, S)$ is monotonically increasing to bound the number of times we need to choose a new w.

The incremental setting is more involved. Since distances can only decrease, our current choice of w becomes a poor approximation of w^* if $d(w, S)$ decreases. A challenge arises because unlike $d(w^*, S)$, the distance $d(w, S)$ does not change monotonically. One can imagine a scenario in which whenever we choose a new w, an edge is added causing $d(w, S)$ to immediately decrease to 1, which mandates that we choose a new w. We address this challenge by carefully employing randomness against an oblivious adversary. We argue that by randomly sampling w from a specifically chosen set of vertices, in expectation it will take a long time for w to become a poor approximation for w^*.

(2) Secondly, we wish to apply a partially dynamic SSSP algorithm as a subroutine, however the state of such algorithms is much better for undirected graphs than directed graphs. For instance, for *undirected* decremental graphs, there is a randomized $(1+\epsilon)$-approximate SSSP algorithm that runs amortized $m^{o(1)}$ time [13] (and it is believed, but not published, than a similar result is possible for incremental graphs), while for incremental/decremental *directed* graphs the best known algorithms for SSSP up to distance k run in amortized time $O(k)$ [11]. To address this discrepancy, we carefully exploit the fact that longer paths are easier to hit by randomly sampling: we augment the algorithm with an additional subsampling routine that quadratically decreases the time dependence on the diameter D.

2 Preliminaries

Let $G = (V, E)$ be a graph, where $|V| = n$ and $|E| = m$. For every $u, v \in V$ let $d_G(u, v)$ be the length of the shortest path from u to v. We omit the subscript when G is clear from context. Let $N_{out}(v, s)$ (resp., $N_{in}(v, s)$) be the set of the s closest outgoing (incoming) vertices of v, where ties are broken by taking the vertex with the smaller ID. The eccentricity $\varepsilon(v)$ of a vertex v is defined as $\max_{u \in V} d(v, u)$. The diameter D of a graph is $\max_{v \in V} \varepsilon(v)$. The radius R of a graph is $\min_{v \in V} \varepsilon(v)$.

2.1 Algorithms

For all of our algorithms for diameter, radius, and eccentricities in undirected graphs as well as diameter in directed graphs, we assume that the diameter is finite. One can easily check if this is the case by running a dynamic reachability algorithm from a single vertex. For our partially dynamic algorithms, we let D_0 and D_f be the initial and final values of the diameter, respectively. Similarly, R_0 and R_f are the initial and final values of the radius, respectively.

The running times of our randomized algorithms are with high probability, which we take to mean with probability at least $1 - 1/n^c$ for all constants c. The running times of our algorithms is written in terms of n and m, which refer to an upper bound on the number of vertices and edges, respectively, over the entire sequence of updates. That is, for incremental algorithms, the running time is written in terms of the final values of n and m and for decremental algorithms the running time written is in terms of the initial values of n and m.

Each of our algorithms is written as a reduction to a black-box incremental or decremental approximation algorithm for *truncated SSSP*; that is, SSSP which provides a distance estimate for all nodes whose distance from the source is at most a given value k. For generality, our algorithms are written for directed graphs and use directed SSSP algorithms, however if the graph is undirected one can simply run an undirected SSSP algorithm instead. Let *out*-$\mathcal{A}_{inc}(u, k, \delta)$ (resp., *out*-$\mathcal{A}_{dec}(u, k, \delta)$) be an incremental (resp., decremental) algorithm that maintains for all v an estimate $d'(u, v)$ such that if $d(u, v) \leq k$ then $(1 - \delta)d(u, v) \leq d'(u, v) \leq d(u, v)$. Analogously, let *in*-$\mathcal{A}_{inc}(u, k, \delta)$ (resp., *in*-$\mathcal{A}_{dec}(u, k, \delta)$) be an incremental (decremental) algorithm that maintains an estimate $d'(v, u)$ for all v such that if $d(u, v) \leq k$ then $(1 - \delta)d(v, u) \leq d'(v, u) \leq d(v, u)$. We assume that after every update, these algorithms output all nodes whose distance estimate has changed. Let $T_{inc}(n, m, k, \delta)$ (resp., $T_{dec}(n, m, k, \delta)$) be the total time of out-$\mathcal{A}_{inc}(u, k, \delta)$ and in-$\mathcal{A}_{inc}(u, k, \delta)$ (resp., out-$\mathcal{A}_{inc}(u, k, \delta)$ and in-$\mathcal{A}_{inc}(u, k, \delta)$) (or the corresponding undirected algorithms).

The running times of our algorithms are written as the maximum of an expression over all values of the diameter D (or radius R) throughout the entire sequence of updates. Although the maximum value of D and R in a partially dynamic graph either occurs at the beginning or end of the update sequence, the maximum value of the running time expression could occur for any value of D or R.

Suppose we run in-$\mathcal{A}_{inc}$, in-$\mathcal{A}_{dec}$, out-$\mathcal{A}_{inc}$, or out-$\mathcal{A}_{dec}$ from a vertex v. Then, let $B_{out}(v, r)$ be the set of vertices u with $d'(v, u) \leq r$.

For a subset $S \subseteq V$ of vertices and a vertex $v \in V$ we define $d(S, v) := \min_{s \in S} d(s, v)$. Similarly, $d(v, S) := \min_{s \in S} d(v, s)$. When the algorithms call for an approximation $d'(S, v)$ of $d(S, v)$, we add a dummy vertex x with an edge to every vertex in S and run out-$\mathcal{A}_{inc}$ (or out-$\mathcal{A}_{dec}$) from x; let $d'(S, v) = d'(x, v) - 1$. We define and maintain $d'(v, S)$ analogously by adding a dummy vertex with an edge from every vertex in S.

▷ Claim 9. For all $u \notin S$, $(1 - 2\delta)d(u, S) \leq d'(u, S) \leq d(u, S)$.

Proof. $(1 - 2\delta)d(u, S) = (1 - 2\delta)(d(u, x) - 1) = d(u, x) - \delta d(u, x) - 1 + \delta(2 - d(u, x)) \leq d(u, x) - \delta d(u, x) - 1 = (1 - \delta)d(u, x) - 1 \leq d'(u, x) - 1 = d'(u, S)$ and $d'(u, S) = d'(u, x) - 1 \leq d(u, x) - 1 = d(u, S)$. ◁

For all of our algorithms for diameter and eccentricities, the bulk of the argument is to prove a lemma of the following form: if one is given values P' and ϵ such that P' is at most the true value P of the parameter of interest, then there is an algorithm that outputs an estimate $\hat{P}$ such that $\alpha(1 - \epsilon)P' - \beta \leq \hat{P} \leq P$ for appropriate α and β. Lemma 10, whose proof we defer to the full version [5], states that a lemma of the above form suffices to prove our theorems. (In Lemma 10, the number k of parameters is 1 for the case of diameter and n for the case of eccentricities.) Lemma 10 also requires a fast constant-factor approximation for the parameter of interest in the static setting. Such algorithms exist for directed diameter and eccentricities in near-linear time.

Lemma 10 does not apply to radius since radius is a minimization problem, however an analogous lemma holds for radius; we defer it to the full version [5].

▶ **Lemma 10.** *Let $\pi_1, \pi_2, \ldots, \pi_k$ be a set of graph parameters (e.g. eccentricities). Suppose there is a static $\tilde{O}(T'(n,m))$ time algorithm that gives a constant-factor approximation for all π_i. Let $P_1, P_2, \ldots, P_k$ be the dynamically changing values of $\pi_1, \pi_2, \ldots, \pi_k$, respectively. Suppose there is an algorithm $\mathcal{P}$ that given a partially dynamic graph and values P' and $\epsilon > 0$, runs in total time $T(n,m,P',\epsilon)$ where T is a polynomial, and maintains a set of estimates $\hat{P}_1 \leq P_1, \ldots, \hat{P}_k \leq P_k$ such that for all i, if $P' \leq P_i$, then $\hat{P}_i \geq \alpha(1-\epsilon)P' - \beta$ for constants α and β.*

Let P_{min} and P_{max} be the minimum and maximum respectively over all P_i over the entire sequence of updates. Then there is an algorithm $\mathcal{P}'$ that given a partially dynamic graph and $\epsilon > 0$, runs in total time $\tilde{O}(T'(n,m) + \max_{P_{min} \leq P' \leq P_{max}} T(n,m,P',\epsilon)/\epsilon)$ and maintains estimates $\hat{P}_1, \ldots, \hat{P}_k$ such that for all i, $\alpha(1-\epsilon)P_i - \beta \leq \hat{P}_i \leq P_i$.

2.2 Lower bounds

Let $k \geq 2$. The k-Orthogonal Vectors Problem (k-OV) is as follows: given k sets $S_1, \ldots, S_k$, where each S_i contains n vectors in $\{0,1\}^d$, determine whether there exist $v_1 \in S_1, \ldots, v_k \in S_k$ so that their generalized inner product is 0, i.e. $\sum_{i=1}^{d} \prod_{j=1}^{k} v_j[i] = 0$. The k-OV Hypothesis is that k-OV requires $n^{k-o(1)}$ time in the word-RAM model of computation with $O(\log n)$ bit words, even for randomized algorithms.

The *unbalanced* version of k-OV has the sets S_i potentially have different sizes, $|S_i| = n_i$. The unbalanced k-OV Hypothesis is that unbalanced k-OV requires $(\prod_i n_i)^{1-o(1)}$ time. When each n_i is polynomial in n, the unbalanced k-OV Hypothesis is known to be equivalent to the k-OV Hypothesis.

R. Williams [25] (see also [24]) showed that if for some $\epsilon > 0$ there is an $n^{k-\epsilon}\text{poly}\,(d)$ time algorithm for k-OV, then CNF-SAT on formulas with N variables and m clauses can be solved in $2^{N(1-\epsilon/k)}\text{poly}\,(m)$ time. In particular, such an algorithm would contradict the Strong Exponential Time Hypothesis (SETH) of Impagliazzo, Paturi and Zane [16] which states that for every $\epsilon > 0$ there is a K such that K-SAT on N variables cannot be solved in $2^{(1-\epsilon)N}\text{poly}\,N$ time (on a word-RAM with $O(\log N)$ bit words) even by randomized algorithms. Thus SETH implies the k-OV Hypothesis for all constants k. Each of our lower bounds conditional upon SETH is a reduction from either unbalanced 2 or 3-OV

The Hitting Set (HS) problem [2] is: given two sets U and V of n vectors each in $\{0,1\}^d$, is there $u \in U$ so that for all $v \in V$, $u \cdot v \neq 0$? The HS Hypothesis states that HS requires $n^{2-o(1)}$ time in the word-RAM model with $O(\log n)$ bit words, even for randomized algorithms.

We introduce the unbalanced version of HS, for three unbalanced sets. Unbalanced 3-HS is the problem, given $U, V, W \subseteq \{0,1\}^d$ with $|U| = n, |V| = n^a, |W| = n^b$ for constants $a, b > 0$, are there $u \in U, w \in W$ so that for all $v \in V$, $u \cdot v \cdot w \neq 0$? This is in similar spirit to unbalanced 3-OV. The unbalanced 3-HS Hypothesis is that unbalanced 3-HS requires $(|U| \cdot |V| \cdot |W|)^{1-o(1)} = n^{1+a+b-o(1)}$ time in the word-RAM model with $O(\log n)$ bit words, even for randomized algorithms. Due to its similarity to 3-OV and the lack of good algorithms, the 3-HS Hypothesis is believable. Refuting it would imply some very interesting improved algorithms for a balanced variant of Quantified Boolean Formulas with 2 quantifiers [18].

As mentioned in the introduction, the k-Cycle Hypothesis is that in the word-RAM model with $O(\log n)$ bit words, any possibly randomized algorithm needs $m^{2-f(k)-o(1)}$ time to find a k-cycle in an m-edge graph, for some continuous (over the reals) $f(k)$ that goes to 0 as k goes to infinity. The Hypothesis is completely consistent with the state of the art k-Cycle algorithms (e.g. [27, 4, 9]).

3 Nearly 3/2-approximation of Diameter

In this section we present a nearly 3/2-approximation for incremental/decremental diameter, given access to a black-box incremental/decremental approximate SSSP algorithm as specified in the preliminaries. We defer the remaining algorithms to the full version [5].

▶ **Theorem 11.** *There is a Las Vegas randomized algorithm for incremental (resp., decremental) diameter in unweighted, directed graphs against an oblivious (resp., adaptive) adversary that given $\epsilon > 0$, runs in $\tilde{O}(\max_{D_f \le D' \le D_0}\{T_{inc}(n,m,D',\epsilon)\frac{\sqrt{n/D'}}{\epsilon^2}\})$ (resp., $\tilde{O}(\max_{D_0 \le D' \le D_f}\{T_{dec}(n,m,D',\epsilon)\frac{\sqrt{n/D'}}{\epsilon^2}\})$) total time with high probability, and maintains an estimate $\hat{D}$ such that $\frac{2(1-\epsilon)}{3}D - \frac{2}{3} \le \hat{D} \le D$ where D is the diameter of the current graph.*

By Lemma 10, the following lemma implies Theorem 11.

▶ **Lemma 12.** *There is a Las Vegas algorithm for incremental (resp., decremental) diameter in unweighted, directed graphs against an oblivious (resp., adaptive) adversary that given $D', \epsilon > 0$, runs in $\tilde{O}\left(T_{inc}(n,m,D',\epsilon)\frac{\sqrt{n/D'}}{\epsilon}\right)$ (resp., $\tilde{O}\left(T_{dec}(n,m,D',\epsilon)\frac{\sqrt{n/D'}}{\epsilon}\right)$) total time with high probability, and maintains an estimate $\hat{D} \le D$ such that if $D' \le D$ then $\hat{D} \ge \frac{2(1-\epsilon)}{3}D' - \frac{2}{3}$ where D is the diameter of the current graph.*

Algorithm

Here we give the algorithm for Lemma 12. We defer the proof to the full version [5]. Let $\delta = 2\epsilon/11$. Throughout the incremental (resp., decremental) algorithm we will run in-$\mathcal{A}_{inc}$ (in-$\mathcal{A}_{dec}$) and out-$\mathcal{A}_{inc}$ (out-$\mathcal{A}_{dec}$) from certain sets of vertices. For ease of notation, we let in-$\mathcal{A}$ denote either in-$\mathcal{A}_{inc}$ or in-$\mathcal{A}_{dec}$, depending on the setting, and similarly we let out-$\mathcal{A}$ denote either out-$\mathcal{A}_{inc}$ or out-$\mathcal{A}_{dec}$.

Initialization. Let α be such that $D' = \Theta(n^{1-2\alpha})$. We randomly sample a set S of size $\Theta(n^\alpha \log^2 n)$ so that with high probability, for every vertex v, after every update, S hits $N_{out}(v, n^{1-\alpha})$. Throughout the entire execution of the algorithm, for all $s \in S$ we run in-$\mathcal{A}(s, D', \delta)$. Additionally, we maintain the approximate distance $d'(v, S)$ from every vertex v to S as described in the preliminaries. Let W be the dynamically changing set of vertices v that satisfy $d'(v, S) > D'/3$.

Phases. In the incremental setting on the other hand, distances can decrease so vertices can leave W. The incremental algorithm may have many phases, and at the beginning of each phase, we choose $w \in W$ uniformly at random. The beginning of a new phase is triggered when w leaves the set W.

Throughout the phase, we run out-$\mathcal{A}(w, D', \delta)$. Also, we will define a subset $S' \subseteq B_{out}(w, \frac{D'}{3})$ and for all $s' \in S'$, we run in-$\mathcal{A}(s', D', \delta)$. S' is initially empty and we independently add each vertex in $B_{out}(w, \frac{D'}{3})$ to S' with probability $\min\{1, \frac{\log^2 n}{\delta D'}\}$. In the incremental setting (but not the decremental setting), $B_{out}(w, \frac{D'}{3})$ can grow, and whenever a vertex u joins $B_{out}(w, \frac{D'}{3})$ we add u to S' with probability $\min\{1, \frac{\log^2 n}{\delta D'}\}$.

Reinitialization. We reinitialize the entire algorithm if at any point during the execution of the algorithm, any of the following occur: (1) $|B_{out}(w, \frac{D'}{3})| > n^{1-\alpha}$, (2) $|S'| > \frac{n^{1-\alpha}\log^4 n}{\delta D'}$, or (3) there is a vertex $v \in B_{out}(w, \frac{D'}{3})$ such that $d'(v, S') > \delta D'$. We will show in the analysis that with high probability we never reinitialize the algorithm.

Query. Following each update, the return value $\hat{D}$ is the maximum distance estimate found over all instantiations of out-$\mathcal{A}$ and in-$\mathcal{A}$. That is,
$\hat{D} = \max\{\max_{v\in V} d'(w, v), \max_{v\in V, s\in S\cup S'} d'(v, s)\}$.

To maintain this value, we maintain the following heaps. For every vertex v that we run out-$\mathcal{A}$ (resp., out-$\mathcal{A}$) from, we keep a max-heap $\mathcal{H}(v)$ that stores for each other vertex u the estimate $d'(v, u)$ (resp., $d'(u, v)$). Let $\hat{d}_{out}(v)$ be the value that $\mathcal{H}(v)$ outputs. Additionally we keep a max-heap $\mathcal{H}$ which stores each $\hat{d}_{out}(v)$.

4 $(2-\epsilon)$-approximation requires linear update time

In this section we give a linear lower bound per update for $(2-\epsilon)$-approximation for diameter, radius, and fixed-vertex eccentricity of undirected graphs. We defer the remaining proofs to the full version [5].

▶ **Theorem 13.** *Let t, ϵ, and ϵ' be positive constants. SETH implies that there exists no fully dynamic algorithm for $(2-\epsilon)$-approximate Diameter, Radius, or fixed-vertex Eccentricity on undirected, unweighted graphs with n vertices and $\tilde{O}(n)$ edges, which has preprocessing time $p(n) = O(n^t)$, amortized update time $u(n) = O(n^{1-\epsilon'})$, and amortized query time $q(n) = O(n^{1-\epsilon'})$. The same holds for the incremental/ decremental settings, for worst-case update and query times.*

Proof of Theorem 13.

Initialization

Let $a = \lceil \frac{2-\epsilon}{2\epsilon} \rceil + 1$ and $\delta = \frac{1-\epsilon'}{t}$. We begin with an instance of 2-OV with vector sets U and V of vectors, with $|U| = N^{\delta}$ and $|V| = N^{1-\delta}$. We create a graph G as follows. Add a node s and for each coordinate c, create two paths of length $2a$ beginning at s, and denote the endpoints of the paths by c_{left} and c_{right}.

Next, create two paths of length a for each vector $u \in U$. Denote the endpoints of one path by u^0_{left} and u^a_{left}, and the endpoints of the other by u^0_{right} and u^a_{right}. Finally, we encode each vector $u \in U$ in the graph by connecting u^a_{left} to c_{left} with a path of length a if $u[c] = 1$, and doing the same on the right side of G. If u has no coordinates equal to 1, then we may report that there is an orthogonal pair and halt; thus there will be no disconnected nodes in G.

Dynamic stages

We proceed in $N^{1-\delta}$ stages, one for each element $v \in V$. For the current v, for each coordinate c where $v[c] = 1$, we add edges (s, c_{left}) and (s, c_{right}). We then query the diameter or radius of G or the eccentricity of s. We will show that the eccentricity of s is always equal to the radius, and that if the diameter is least $8a$ or the radius is at least $4a$, then there is an orthogonal pair u, v; otherwise, the diameter is at most $4a+2$ or the radius is at most $2a+1$. We have set a so that a $(2-\epsilon)$-approximation algorithm for diameter, radius or eccentricity of s distinguishes between these two cases and thus detects an orthogonal pair. If the query

does not detect an orthogonal pair, we undo the edge additions for the stage and continue to the next v. We can modify the stage to be decremental by beginning with edges from s to all nodes c_{left} and c_{right}, and removing the excess edges each stage.

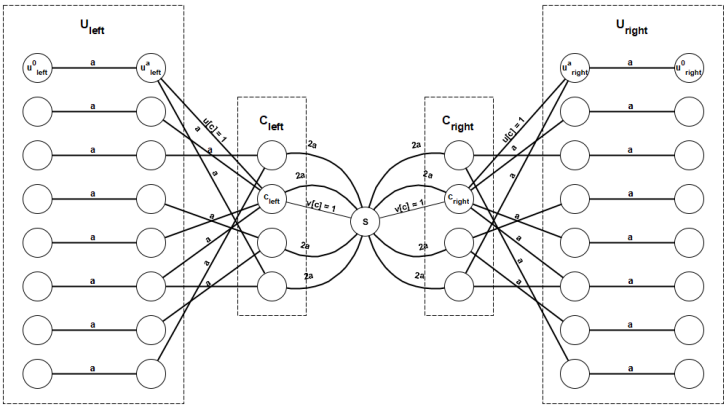


Figure 1 Sketch of Theorem 13 Construction. Bold edges represent paths, whose labels denote their length.

Correctness

We claim that the node s is always the center of G, so the eccentricity of s is always the radius of G. Let x_{right} be the node farthest from s on the right side of G. Since the graph is symmetrical, the counterpart x_{left} of x_{right} is such that $d(s, x_{right}) = d(s, x_{left})$. Any node y to the left of s must pass through s to reach x_{right}, so y has a higher eccentricity than s because it is farther from the node farthest from s. Symmetrically, any node y to the right of s must pass through s to reach x_{left}, so y has a higher eccentricity than s because it is farther from the node farthest from s.

If for the current stage, for all u, $u \cdot v \neq 0$, then for each u there must be some coordinate c such that $u[c] = v[c] = 1$. Then there is a path of length 1 from s to c_{left}, and a path of length a from c_{left} to u^a_{left}, for all u. The same is true on the right side. Then since all nodes except s are of distance at most a from a node u'^a_{left} or u'^a_{right} for some $u' \in U$, all nodes are accessible in at most $2a + 1$ steps from s. This means that the radius and eccentricity of s is $2a + 1$, and the diameter is at most $4a + 2$.

If for the current stage there is some u such that $u \cdot v = 0$, then there is no direct path from s to u^0 on either side via a vector coordinate c and u^a. A path via a different u'_a would be of length at least $4a+1$, because returning to a c' where $u[c'] = 1$ would cost an additional $2a$ from the direct path. The shortest path would thus be along the length-$2a$ path from s to c', giving $d(s, u^0) = 4a$. The radius and eccentricity of s must be at least $4a$ and diameter must be at least $8a$, because $d(u^0_{left}, u^0_{right}) = d(s, u^0_{left}) + d(s, u^0_{right}) = 4a + 4a = 8a$.

Running time

We assume for the sake of contradiction that the algorithm of Theorem 13 exists. Let $n = N^\delta$ be the size of G. We have that $u(n) = q(n) = O((N^\delta)^{1-\epsilon'})$. After initialization and $|V| = N^{1-\delta}$ stages, the total update and query time is then at most $\tilde{O}(N^{1-\delta\epsilon'})$. The preprocessing time $p(n)$ for the algorithm on G is $O((N^\delta)^t) = O(N^{1-\epsilon'})$. Thus the total time of the algorithm is $\tilde{O}(N^{1-\epsilon'} + N^{1-\delta\epsilon'})$. This contradicts SETH, because SETH implies that no algorithm exists for 2-OV in $O((|U| \cdot |V|)^{1-\epsilon''}) = O(N^{1-\epsilon''})$ time for any $\epsilon'' > 0$. ◀

References

1 Amir Abboud and Virginia Vassilevska Williams. Popular Conjectures Imply Strong Lower Bounds for Dynamic Problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 434–443, 2014.

2 Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and Fixed Parameter Subquadratic Algorithms for Radius and Diameter in Sparse Graphs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 377–391, 2016.

3 R. Albert, H. Jeong, and A.L. Barabasi. Diameter of the world wide web. *Nature*, 401:130–131, 1999.

4 N. Alon, R. Yuster, and U. Zwick. Finding and counting given length cycles. *Algorithmica*, 17:209–223, 1997.

5 Bertie Ancona, Monika Henzinger, Liam Roditty, Virginia Vassilevska Williams, and Nicole Wein. Algorithms and Hardness for Diameter in Dynamic Graphs. *arXiv preprint*, 2018. arXiv:1811.12527.

6 Arturs Backurs, Liam Roditty, Gilad Segal, Virginia Vassilevska Williams, and Nicole Wein. Towards Tight Approximation Bounds for Graph Diameter and Eccentricities. In *Proceedings of STOC'18*, page to appear, 2018.

7 Massimo Cairo, Roberto Grossi, and Romeo Rizzi. New Bounds for Approximating Extremal Distances in Undirected Graphs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 363–376, 2016.

8 Shiri Chechik, Daniel H. Larkin, Liam Roditty, Grant Schoenebeck, Robert Endre Tarjan, and Virginia Vassilevska Williams. Better Approximation Algorithms for the Graph Diameter. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1041–1052, 2014.

9 Mina Dalirrooyfard, Thuy Duong Vuong, and Virginia Vassilevska Williams. Graph pattern detection: Hardness for all induced patterns and faster non-induced cycles. In *STOC*, page to appear, 2019.

10 Camil Demetrescu and Giuseppe F. Italiano. A New Approach to Dynamic All Pairs Shortest Paths. *Journal of the ACM*, 51(6):968–992, 2004. Announced at STOC'03. doi:10.1145/1039488.1039492.

11 Shimon Even and Yossi Shiloach. An On-Line Edge-Deletion Problem. *Journal of the ACM*, 28(1):1–4, 1981. doi:10.1145/322234.322235.

12 Monika Henzinger and Valerie King. Fully Dynamic Biconnectivity and Transitive Closure. In *Symposium on Foundations of Computer Science (FOCS)*, pages 664–672, 1995. doi:10.1109/SFCS.1995.492668.

13 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Decremental Single-Source Shortest Paths on Undirected Graphs in Near-Linear Total Update Time. In *Symposium on Foundations of Computer Science (FOCS)*, pages 146–155, 2014. doi:10.1109/FOCS.2014.24.

14 Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and Strengthening Hardness for Dynamic Problems via the Online Matrix-Vector Multiplication Conjecture. In *Symposium on Theory of Computing (STOC)*, pages 21–30, 2015. doi:10.1145/2746539.2746609.

15 Monika Henzinger, Andrea Lincoln, Stefan Neumann, and Virginia Vassilevska Williams. Conditional Hardness for Sensitivity Problems. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:31, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITCS.2017.26.

16 R. Impagliazzo, R. Paturi, and F. Zane. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

17 François Le Gall. Powers of tensors and fast matrix multiplication. In *International Symposium on Symbolic and Algebraic Computation, ISSAC '14, Kobe, Japan, July 23-25, 2014*, pages 296–303, 2014.

18 Moshe Lewenstein, Seth Pettie, and Virginia Vassilevska Williams. Open Problems from Dagstuhl Seminar 16451: Structure and Hardness in P, 2016.

19 S. Pettie. A new approach to all-pairs shortest paths on real-weighted graphs. *Theor. Comput. Sci.*, 312(1):47–74, 2004.

20 Seth Pettie and Vijaya Ramachandran. A Shortest Path Algorithm for Real-Weighted Undirected Graphs. *SIAM J. Comput.*, 34(6):1398–1431, 2005.

21 Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, STOC '13, pages 515–524, 2013.

22 R. Seidel. On the All-Pairs-Shortest-Path Problem in Unweighted Undirected Graphs. *J. Comput. Syst. Sci.*, 51(3):400–403, 1995.

23 Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 887–898. ACM, 2012.

24 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians*, page to appear, 2018.

25 R. Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2–3):357–365, 2005.

26 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 664–673, 2014.

27 R. Yuster and U. Zwick. Detecting short directed cycles using rectangular matrix multiplication and dynamic programming. In *Proc. SODA*, pages 247–253, 2004.

28 Uri Zwick. All Pairs Shortest Paths using Bridging Sets and Rectangular Matrix Multiplication. *Journal of the ACM*, 49(3):289–317, 2002. Announced at FOCS'98. doi:10.1145/567112.567114.

Log Diameter Rounds Algorithms for 2-Vertex and 2-Edge Connectivity

Alexandr Andoni
Columbia University, New York City, NY, USA
andoni@cs.columbia.edu

Clifford Stein
Columbia University, New York City, NY, USA
cliff@cs.columbia.edu

Peilin Zhong
Columbia University, New York City, NY, USA
pz2225@columbia.edu

Abstract

Many modern parallel systems, such as MapReduce, Hadoop and Spark, can be modeled well by the MPC model. The MPC model captures well coarse-grained computation on large data – data is distributed to processors, each of which has a sublinear (in the input data) amount of memory and we alternate between rounds of computation and rounds of communication, where each machine can communicate an amount of data as large as the size of its memory. This model is stronger than the classical PRAM model, and it is an intriguing question to design algorithms whose running time is smaller than in the PRAM model.

In this paper, we study two fundamental problems, 2-edge connectivity and 2-vertex connectivity (biconnectivity). PRAM algorithms which run in $O(\log n)$ time have been known for many years. We give algorithms using roughly log diameter rounds in the MPC model. Our main results are, for an n-vertex, m-edge graph of diameter D and bi-diameter D', 1) a $O(\log D \log\log_{m/n} n)$ parallel time 2-edge connectivity algorithm, 2) a $O(\log D \log^2 \log_{m/n} n + \log D' \log\log_{m/n} n)$ parallel time biconnectivity algorithm, where the bi-diameter D' is the largest cycle length over all the vertex pairs in the same biconnected component. Our results are fully scalable, meaning that the memory per processor can be $O(n^\delta)$ for arbitrary constant $\delta > 0$, and the total memory used is linear in the problem size. Our 2-edge connectivity algorithm achieves the same parallel time as the connectivity algorithm of [4]. We also show an $\Omega(\log D')$ conditional lower bound for the biconnectivity problem.

2012 ACM Subject Classification Theory of computation → MapReduce algorithms; Mathematics of computing → Paths and connectivity problems

Keywords and phrases parallel algorithms, biconnectivity, 2-edge connectivity, the MPC model

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.14

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1905.00850`.

Funding *Alexandr Andoni*: Research partly supported by NSF Grants (CCF-1617955 and CCF-1740833), Simons Foundation (#491119) and Google Research Award.
Clifford Stein: Research partly supported by NSF Grants CCF-1714818 and CCF-1822809.
Peilin Zhong: Research partly supported by NSF Grants (CCF-1703925, CCF-1421161, CCF-1714818, CCF-1617955 and CCF-1740833), Simons Foundation (#491119) and Google Research Award.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 14; pp. 14:1–14:16

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

1 Introduction

The success of modern parallel and distributed systems such as MapReduce [16, 17], Spark [41], Hadoop [39], Dryad [23], together with the need to solve problems on massive data, is driving the development of new algorithms which are more efficient and scalable in these large-scale systems. An important theoretical problem is to develop models which are good abstractions of these computational frameworks. The *Massively Parallel Computation (MPC)* model [25, 21, 11, 3, 9, 15, 4] captures the capabilities of these computational systems while keeping the description of the model itself simple. In the MPC model, there are machines (processors), each with $\Theta(N^\delta)$ local memory, where N denotes the size of the input and $\delta \in (0, 1)$. The computation proceeds in rounds, where each machine can perform unlimited local computation in a round and exchange $O(N^\delta)$ data at the end of the round. The parallel time of an algorithm is measured by the total number of computation-communication rounds. The MPC model is a variant of the Bulk Synchronous Parallel (BSP) model [38]. It is also a more powerful model than the PRAM since any PRAM algorithm can be simulated in the MPC model [25, 21] while some problem can be solved in a faster parallel time in the MPC model. For example, computing the XOR of N bits takes $O(1/\delta)$ parallel time in the MPC model but needs near-logarithmic parallel time on the most powerful CRCW PRAM [10].

A natural question to ask is: which problems can be solved in faster parallel time in the MPC model than on a PRAM? This question has been studied by a line of recent papers [25, 19, 29, 3, 1, 6, 22, 15, 7, 14, 13, 32, 20]. Most of these results studied the graph problems, which are the usual benchmarks of parallel/distributed models. Many graph problems such as graph connectivity [35, 33, 30], graph biconnectivity [37, 36], maximal matching [26], minimum spanning tree [27] and maximal independent set [31, 2] can be solved in the standard logarithmic time in the PRAM model, but these problems have been shown to have a better parallel time in the MPC model.

In addition, we hope to develop *fully scalable* algorithms for the graph problems, i.e., the algorithm should work for any constant $\delta > 0$. The previous literatures show that a graph problem in the MPC model with large local memory size may be much easier than the same problem in the MPC model but with a smaller local memory size. In particular, when the local memory size per machine is close to the number of vertices n, many graph problems have efficient algorithms. For example, if the local memory size per machine is $n/\log^{O(1)} n$, the connectivity problem [7] and the approximate matching problem [5] can be solved in $O(\log \log n)$ parallel time. If the local memory size per machine is $\Omega(n)$, then the MPC model meets the congested clique model [12]. In this setting, the connectivity problem and the minimum spanning tree problem can be solved in $O(1)$ parallel time [24]. If the local memory size per machine is $n^{1+\Omega(1)}$, many graph problems such as maximal matching, approximate weighted matchings, approximate vertex and edge covers, minimum cuts, and the biconnectivity problem can be solved in $O(1)$ parallel time [29, 8]. The landscape of graph algorithms in the MPC model with small local memory is more nuanced and challenging for algorithm designers. If the local memory size per machine is $n^{1-\Omega(1)}$, then the best connectivity algorithm takes parallel time $O(\log D \log \log n)$ where D is the diameter of the graph [4], and the best approximate maximum matching algorithm takes parallel time $\widetilde{O}(\sqrt{\log n})$ [32].

Therefore, the main open question is: which kind of the graph problems can have faster fully scalable MPC algorithms than the standard logarithmic PRAM algorithms?

Two fundamental graph problems in graph theory are 2-edge connectivity and 2-vertex connectivity (biconnectivity). In this work, we studied these two problems in the MPC model. Consider an n-vertex, m-edge undirected graph G. A bridge of G is an edge whose removal

increases the number of connected components of G. In the 2-edge connectivity problem, the goal is to find all the bridges of G. For any two different edges e, e' of G, e, e' are in the same biconnected component (block) of G if and only if there is a simple cycle which contains both e, e'. If we define a relation R such that eRe' if and only if $e = e'$ or e, e' are contained by a simple cycle, then R is an equivalence relation [18]. Thus, a biconnected component is an induced graph of an equivalence class of R. In the biconnectivity problem, the goal is to output all the biconnected components of G. We proposed faster, fully scalable algorithms for the both 2-edge connectivity problem and the biconnectivity problem by parameterizing the running time as a function of the *diameter* and the *bi-diameter* of the graph. The diameter D of G is the largest diameter of its connected components. The definition of bi-diameter is a natural generalization of the definition of diameter. If vertices u, v are in the same biconnected component, then the cycle length of (u, v) is defined as the minimum length of a simple cycle which contains both u and v. The bi-diameter D' of G is the largest cycle length over all the vertex pairs (u, v) where both u and v are in the same biconnected component. Our main results are 1) a fully scalable $O(\log D \log \log_{m/n} n)$ parallel time 2-edge connectivity algorithm, 2) a fully scalable $O(\log D \log^2 \log_{m/n} n + \log D' \log \log_{m/n} n)$ parallel time biconnectivity algorithm. Our 2-edge connectivity algorithm achieves the same parallel time as the connectivity algorithm of [4]. We also show an $\Omega(\log D')$ conditional lower bound for the biconnectivity problem.

1.1 The Model

Our model of computation is the Massively Parallel Computation (MPC) model [25, 21, 11].

Consider two non-negative parameters $\gamma \geq 0, \delta > 0$. In the (γ, δ)-MPC model [4], there are p machines (processors) each with local memory size s, where $p \cdot s = \Theta(N^{1+\gamma}), s = \Theta(N^{\delta})$ and N denotes the size of the input data. Thus, the space per machine is sublinear in N, and the total space is only an $O(N^{\gamma})$ factor more than the input size. In particular, if $\gamma = 0$, the total space available in the system is linear in the input size N. The space size is measured by words each containing $\Theta(\log(s \cdot p))$ bits. Before the computation starts, the input data is distributed on $\Theta(N/s)$ input machines. The computation proceeds in rounds. In each round, each machine can perform local computation on its local data, and send messages to other machines at the end of the round. In a round, the total size of messages sent/received by a machine should be bounded by its local memory size $s = \Theta(N^{\delta})$. For example, a machine can send s size 1 messages to s machines or send a size s message to 1 machine in a single round. However, it cannot broadcast a size s message to every machine. In the next round, each machine only holds the received messages in its local memory. At the end of the computation, the output data is distributed on the output machines. An algorithm in this model is called a (γ, δ)-MPC algorithm. The parallel time of an algorithm is the total number of rounds needed to finish its computation. In this paper, we consider δ an arbitrary constant in $(0, 1)$.

1.2 Our Results

Our main results are efficient MPC algorithms for 2-edge connectivity and biconnectivity problems. In our algorithms, one important subroutine is computing the Depth-First-Search (DFS) sequence [4] which is a variant of the Euler tour representation proposed by [37, 36] in 1984. We show how to efficiently compute the DFS sequence in the MPC model with linear total space. Conditioned on the hardness of the connectivity problem in the MPC model, we prove a hardness result on the biconnectivity problem.

For 2-edge connectivity and biconnectivity, the input is an undirected graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges. $N = n + m$ denotes the size of the representation of G, D denotes the diameter of G, and D' denotes the bi-diameter of G. We state our results in the following.

Biconnectivity. In the biconnectivity problem, we want to find all the biconnected components (blocks) of the input graph G. Since the biconnected components of G define a partition on E, we just need to color each edge, i.e., at the end of the computation, $\forall e \in E$, there is a unique tuple (x, c) with $x = e$ stored on an output machine, where c is called the color of e, such that the edges e_1, e_2 are in the same biconnected components if and only if they have the same color.

▶ **Theorem 1** (Biconnectivity in MPC). *For any $\gamma \in [0, 2]$ and any constant $\delta \in (0, 1)$, there is a randomized (γ, δ)-MPC algorithm which outputs all the biconnected components of the graph G in $O\left(\log D \cdot \log^2 \frac{\log n}{\log(N^{1+\gamma}/n)} + \log D' \cdot \log \frac{\log n}{\log(N^{1+\gamma}/n)}\right)$ parallel time. The success probability is at least 0.95. If the algorithm fails, then it returns FAIL.*

The worst case is when the input graph is sparse and the total space available is linear in the input size, i.e., $N = n + m = O(n)$ and $\gamma = 0$. In this case, the parallel running time of our algorithm is $O(\log D \cdot \log^2 \log n + \log D' \cdot \log \log n)$. If the graph is slightly denser ($m = n^{1+c}$ for some constant $c > 0$), or the total space is slightly larger ($\gamma > 0$ is a constant), then we obtain $O(\log D + \log D')$ time.

A cut vertex (articulation point) in the graph G is a vertex whose removal increases the number of connected components of G. Since a vertex v is a cut vertex if and only if there are two edges e_1, e_2 which share the endpoint v and e_1, e_2 are not in the same biconnected component, our algorithm can also find all the cut vertices of G.

2-Edge connectivity. In the 2-edge connectivity problem, we want to output all the bridges of the input graph G. Since an edge is a bridge if and only if each of its endpoints is either a cut vertex or a vertex with degree 1, the 2-edge connectivity problem should be easier than the biconnectivity problem. We show how to solve 2-edge connectivity in the same parallel time as the algorithm proposed by [4] for solving connectivity.

▶ **Theorem 2** (2-Edge connectivity in MPC). *For any $\gamma \in [0, 2]$ and any constant $\delta \in (0, 1)$, there is a randomized (γ, δ)-MPC algorithm which outputs all the bridges of the graph G in $O\left(\log D \cdot \log \frac{\log n}{\log(N^{1+\gamma}/n)}\right)$ parallel time. The success probability is at least 0.97. If the algorithm fails, then it returns FAIL.*

DFS sequence. A rooted tree with a vertex set V can be represented by $n = |V|$ pairs $(v_1, \text{par}(v_1)), (v_2, \text{par}(v_2)), \cdots, (v_n, \text{par}(v_n))$ where $\text{par} : V \to V$ is a set of parent pointers, i.e., for a non-root vertex v, $\text{par}(v)$ denotes the parent of v, and for the root vertex v, $\text{par}(v) = v$. We show an algorithm which can compute the DFS sequence (Definition 6) of the rooted tree in the MPC model with linear total space.

▶ **Theorem 3** (DFS sequence of a tree in MPC). *Given a rooted tree represented by a set of parent pointers $\text{par} : V \to V$, there is a randomized $(0, \delta)$-MPC algorithm which outputs the DFS sequence in $O(\log D)$ parallel time, where $\delta \in (0, 1)$ is an arbitrary constant, D is the depth of the tree. The success probability is at least 0.99. If the algorithm fails, then it returns FAIL.*

Conditional hardness for biconnectivity. A conjectured hardness for the connectivity problem is the *one cycle vs. two cycles* conjecture: for any $\gamma \geq 0$ and any constant $\delta \in (0,1)$, any (γ,δ)-MPC algorithm requires $\Omega(\log n)$ parallel time to determine whether the input n-vertex graph is a single cycle or contains two disjoint length $n/2$ cycles. This conjectured hardness result is widely used in the MPC literature [25, 11, 28, 34, 40]. Under this conjecture, we show that $\Omega(\log D')$ parallel time is necessary for the biconnectivity problem, and this is true even when $D = O(1)$, i.e., the diameter of the graph is a constant.

▶ **Theorem 4** (Hardness of biconnectivity in MPC). *For any $\gamma \geq 0$ and any constant $\delta \in (0,1)$, unless there is a (γ,δ)-MPC algorithm which can distinguish the following two instances: 1) a single cycle with n vertices, 2) two disjoint cycles each contains $n/2$ vertices, in $o(\log n)$ parallel time, any (γ,δ)-MPC algorithm requires $\Omega(\log D')$ parallel time for testing whether a graph G with a constant diameter is biconnected.*

1.3 Our Techniques

Biconnectivity. At a high level our biconnectivity algorithm is based on a framework proposed by [36]. The main idea is to construct a new graph and reduce the problem of finding biconnected components of G to the problem of finding connected components of the new graph G'. At first glance, it should be efficiently solved by the connectivity algorithm [4]. However, there are two main issues: 1) since the parallel time of the MPC connectivity algorithm of [4] depends on the diameter of the input graph, we need to make the diameter of G' small, 2) we need to construct G' efficiently. Let us first consider the first issue, and we will discuss the second issue later.

We give an analysis of the diameter of $G' = (V', E')$ constructed by [36]. Without loss of generality, we can suppose the input $G = (V, E)$ is connected. Each vertex in G' corresponds to an edge of G. Let T be an arbitrary spanning tree of G with depth d. Each non-tree edge e can define a simple cycle C_e which contains the edge e and the unique path between the endpoints of e in the tree T. Thus, the length of C_e is at most $2d + 1$. If there is a such cycle containing any two tree edges $(u, v), (v, w)$, vertices $(u, v), (v, w)$ are connected in G'. For each non-tree edge e, we connect the vertex e to the vertex e' in graph G' where e' is an arbitrary tree edge in the cycle C_e. By the construction of G', any e, e' from the same connected components of G' should be in the same biconnected components of G. Now consider arbitrary two edges e, e' in the same biconnected component of G. There must be a simple cycle C which contains both edges e, e' in G. Since all the simple cycles defined by the non-tree edges are a cycle basis of G [18], the edge set of C can be represented by the xor sum of all the edge sets of k basis cycles $C_1, C_2, \cdots, C_k$ where C_i is a simple cycle defined by a non-tree edge e_i on the cycle C. k is upper bounded by the bi-diameter of G. Furthermore, we can assume C_i intersects C_{i+1}. There should be a path between e, e' in G', and the length of the path is at most $\sum_{i=1}^{k} |C_i| \leq O(k \cdot d)$. So, the diameter of G' is upper bounded by $O(k \cdot d)$. Thus, according to [4], we can find the connected components of G' in $\sim (\log k + \log d)$ parallel time, where d and k are upper bounded by the diameter and the bi-diameter of G respectively.

Now let us consider how to construct G' efficiently. The bottleneck is to determine whether the tree edges $(u, v), (v, w)$ should be connected in G' or not. Suppose w is the parent of v and v is the parent of u. The vertex (u, v) should connect to the vertex (v, w) in G' if and only if there is a non-tree edge that connects a vertex x in the subtree of u and a vertex y which is on the outside of the subtree of v. For each vertex x, let $\mathrm{lev}(x)$ be the minimum depth of the least common ancestor (LCA) of (x, y) over all the non-tree edges

(x, y). Then (u, v) should be connected to (v, w) in G' if and only if there is a vertex x in the subtree of u in G such that $\text{lev}(x)$ is smaller than the depth of v. Since the vertices in a subtree should appear consecutively in the DFS sequence, this question can be solved by some range queries over the DFS sequence. Next, we will discuss how to compute the DFS sequence of a tree.

DFS sequence. The DFS sequence of a tree is a variant of the Euler tour representation of the tree. For an n-vertex tree T, [36] gives an $O(\log n)$ parallel time PRAM algorithm for the Euler tour representation of T. However, since their construction method will destroy the tree structure, it is hard to get a faster MPC algorithm based on this framework. Instead, we follow the leaf sampling framework proposed by [4]. Although the DFS sequence algorithm proposed by [4] takes $O(\log d)$ time where d is the depth of T, it needs $\Omega(n \log d)$ total space. The bottleneck is the subroutine which needs to solve the least common ancestors problem and generate multiple path sequences. The previous algorithm uses the doubling algorithm for the subroutine, i,e., for each vertex v, they store the 2^i-th ancestor of v for every $i \in [\lceil \log d \rceil]$. This is the reason why [4] cannot achieve the linear total space. We show how to compress the tree T into a new tree T' which only contains at most $n/\lceil \log d \rceil$ vertices. We argue that applying the doubling algorithm on T' is sufficient for us to find the DFS sequence of T.

2-Edge connectivity. Without loss of generality, we can assume the input graph G is connected. Consider a rooted spanning tree T and an edge $e = (u, v)$ in G. Suppose the depth of u is at least the depth of v in T, i.e., v cannot be a child of u. The edge e is not a bridge if and only if either e is a non-tree edge or there is a non-tree edge (x, y) connecting the subtree of u and a vertex on the outside of the subtree of u. Similarly, the second case can be solved by some range queries over the DFS sequence of T.

Conditional hardness for biconnectivity. We want to reduce the connectivity problem to the biconnectivity problem. For an undirected graph G, if we add an additional vertex v^* and connects v^* to every vertex of G, then the diameter of the resulting graph G' is at most 2 and each biconnected components of G' corresponds to a connected component of G. Furthermore, the bi-diameter of G' is upper bounded by the diameter of G plus 2. Therefore, if the parallel time of an algorithm $\mathcal{A}'$ for finding the biconnected components of G' depends on the bi-diameter of G', there exists an algorithm $\mathcal{A}$ which can find all the connected components of G in the parallel time which has the same dependence on the diameter of G.

1.4 A Roadmap

Section 2 introduces the notation and some useful definitions. Section 3 describes the offline algorithms for 2-edge connectivity and biconnectivity. It also includes some crucial properties of the algorithms. In Section 4, we show an linear space offline algorithm to find the DFS sequence of a tree. All of these offline algorithms can be implemented in the MPC model efficiently. Section 5 contains the conditional hardness result for the biconnectivity problem in the MPC model. For the MPC implementations and all the missing technical proofs, we refer readers to the full version of the paper.

2 Preliminaries

2.1 Notation

We follow the notation of [4]. $[n]$ denotes the set of integers $\{1, 2, \cdots, n\}$.

Diameter and bi-diameter. Consider an undirected graph G with a vertex set V and an edge set E. For any two vertices u, v, we use $\text{dist}_G(u, v)$ to denote the distance between u and v in graph G. If u, v are not in the same (connected) component of G, then $\text{dist}_G(u, v) = \infty$. The diameter $\text{diam}(G)$ of G is the largest diameter of its connected components, i.e., $\text{diam}(G) = \max_{u,v \in V: \text{dist}_G(u,v) \neq \infty} \text{dist}_G(u, v)$. $(v_1, v_2, \cdots, v_k) \in V^k$ is a cycle of length $k - 1$ if $v_1 = v_k$ and $\forall i \in [k - 1], (v_i, v_{i+1}) \in E$. We say a cycle $(v_1, v_2, \cdots, v_k)$ is simple if $k \geq 4$ and each vertex only appears once in the cycle except v_1 (v_k). Consider two different vertices $u, v \in V$. We use $\text{cyclen}_G(u, v)$ to denote the minimum length of a simple cycle which contains both vertices u and v. If there is no simple cycle which contains both u and v, $\text{cyclen}_G(u, v) = \infty$. $\text{cyclen}_G(u, u)$ is defined as 0. The bi-diameter of G, $\text{bi-diam}(G)$, is defined as $\max_{u,v \in V: \text{cyclen}_G(u,v) \neq \infty} \text{cyclen}_G(u, v)$.

Representation of a rooted forest. Let V denote a set of vertices. We represent a rooted forest in the same manner as [4]. Consider a mapping $\text{par} : V \to V$. For $i \in \mathbb{N}_{>0}$ and $v \in V$, we define $\text{par}^{(i)}(v)$ as $\text{par}(\text{par}^{(i-1)}(v))$, and $\text{par}^{(0)}(v)$ is defined as v itself. If $\forall v \in V, \exists i > 0$ such that $\text{par}^{(i)}(v) = \text{par}^{(i+1)}(v)$, then we call par a set of parent pointers on V. For $v \in V$, if $\text{par}(v) = v$, then we say v is a root of par. Notice that par actually can represent a rooted forest, thus par can have more than one root. The depth of $v \in V$, $\text{dep}_{\text{par}(v)}$ is the smallest $i \in \mathbb{N}$ such that $\text{par}^{(i)}(v)$ is the same as $\text{par}^{(i+1)}(v)$. The root of $v \in V$, $\text{par}^{(\infty)}(v)$ is defined as $\text{par}^{(\text{dep}_{\text{par}}(v))}(v)$. The depth of par, $\text{dep}(\text{par})$ is defined as $\max_{v \in V} \text{dep}_{\text{par}}(v)$.

Ancestor and path. For two vertices $u, v \in V$, if $\exists i \in \mathbb{N}$ such that $u = \text{par}^{(i)}(v)$, then u is an ancestor of v (in par). If u is an ancestor of v, then the path $P(v, u)$ (in par) from v to u is a sequence $(v, \text{par}(v), \text{par}^{(2)}(v), \cdots, u)$ and the path $P(u, v)$ is the reverse of $P(v, u)$, i.e., $P(u, v) = (u, \cdots, \text{par}^{(2)}(v), \text{par}(v), v)$. If an ancestor u of v is also an ancestor of w, then u is a common ancestor of (v, w). Furthermore, if a common ancestor u of (v, w) satisfies $\text{dep}_{\text{par}}(u) \geq \text{dep}_{\text{par}}(x)$ for any common ancestor x of (v, w), then u is the lowest common ancestor (LCA) of (v, w).

Children and leaves. For any non-root vertex u of par, u is a child of $\text{par}(u)$. For any vertex $v \in V$, $\text{child}_{\text{par}}(v)$ denotes the set of all the children of v, i.e., $\text{child}_{\text{par}}(v) = \{u \in V \mid u \neq v, \text{par}(u) = v\}$. If u is the k^{th} smallest vertex in the set $\text{child}_{\text{par}}(v)$, then we define $\text{rank}_{\text{par}}(u) = k$, or in other words, u is the k^{th} child of v. If v is a root vertex of par, then $\text{rank}_{\text{par}}(v)$ is defined as 1. $\text{child}_{\text{par}}(v, k)$ denotes the k^{th} child of v. For simplicity, if par is clear in the context, we just use $\text{child}(v)$, $\text{rank}(v)$ and $\text{child}(v, k)$ to denote $\text{child}_{\text{par}}(v)$, $\text{rank}_{\text{par}}(v)$ and $\text{child}_{\text{par}}(v, k)$ for short. If $\text{child}(v) = \emptyset$, then v is a leaf of par. We denote $\text{leaves}(\text{par})$ as the set of all the leaves of par, i.e., $\text{leaves}(\text{par}) = \{v \mid \text{child}(v) = \emptyset\}$.

2.2 Depth-First-Search Sequence

The Euler tour representation of a tree is proposed by [37, 36]. It is a crucial building block in many graph algorithms including biconnectivity algorithms. The Depth-First-Search (DFS) sequence [4] of a rooted tree is a variant of the Euler tour representation. Let us first introduce some relevant concepts of the DFS sequence.

▶ **Definition 5** (Subtree [4]). *Consider a set of parent pointers* $\text{par} : V \to V$ *on a vertex set* V*. Let* v *be a vertex in* V*, and let* $V' = \{u \in V \mid v \text{ is an ancestor of } u\}$*.* $\text{par}' : V' \to V'$ *is a set of parent pointers on* V'*. If* $\forall u \in V' \setminus \{v\}$*,* $\text{par}'(u) = \text{par}(u)$ *and* $\text{par}'(v) = v$*, then* par′ *is a subtree of* v *in* par*. For* $u \in V'$*, we say* u *is in the subtree of* v*.*

The definition of the DFS sequence is the following:

▶ **Definition 6** (DFS sequence [4]). *Consider a set of parent pointers* $\text{par} : V \to V$ *on a vertex set* V*. Let* v *be a vertex in* V*. If* v *is a leaf in* par*, then the DFS sequence of the subtree of* v *is* (v)*. Otherwise, the DFS sequence of the subtree of* v *is defined recursively as*

$$(v, a_{1,1}, a_{1,2}, \cdots, a_{1,n_1}, v, a_{2,1}, a_{2,2}, \cdots, a_{2,n_2}, v, \cdots, a_{k,1}, a_{k,2}, \cdots, a_{k,n_k}, v),$$

where $k = |\text{child}(v)|$ *and* $\forall i \in [k]$*,* $(a_{i,1}, a_{i,2}, \cdots, a_{i,n_i})$ *is the DFS sequence of the subtree of* $\text{child}(v, i)$*, i.e., the* i^{th} *child of* v*.*

If $\text{par} : V \to V$ has a unique root v, then we define the DFS sequence of par as the DFS sequence of the subtree of v. By the definition of the DFS sequence, for any two consecutive elements a_i and a_{i+1} in the sequence, a_i is either a parent of a_{i+1} or a_i is a child of a_{i+1}. Furthermore, for any vertex v, if both elements a_i and a_j $(i < j)$ in the DFS sequence A are v, any element a_k between a_i and a_j (i.e., $i \leq k \leq j$) should be a vertex in the subtree of v.

3 2-Edge Connectivity and Biconnectivity

Consider a connected undirected graph G with a vertex set V and an edge set E. In the 2-edge connectivity problem, the goal is to find all the bridges of G, where an edge $e \in E$ is called a bridge if its removal disconnects G. In the biconnectivity problem, the goal is to partition the edges into several groups $E_1, E_2, \cdots, E_k$, i.e., $E = \bigcup_{i=1}^{k} E_i, \forall i \neq j, E_i \cap E_j = \emptyset$, such that $\forall e \neq e' \in E$, e and e' are in the same group if and only if there is a simple cycle in G which contains both e and e'. A subgraph induced by an edge group E_i is called a biconnected component (block). In other words, the goal of the biconnectivity problem is to find all the blocks of G.

In this section, we describe the algorithms for both the 2-edge connectivity problem and the biconnectivity problem in the offline setting.

3.1 2-Edge Connectivity

The 2-edge connectivity problem is much simpler than the biconnectivity problem. We first compute a spanning tree of the graph. Only a tree edge can be a bridge. Then for any non-root vertex v, if there is no non-tree edge which crosses between the subtree of v and the outside of the subtree of v, then the tree edge which connects v to its parent is a bridge.

▶ **Lemma 7** (2-Edge connectivity). *Consider an undirected graph* $G = (V, E)$*. Let* B *be the output of* Bridges(G)*. Then* B *is the set of all the bridges of* G*.*

3.2 Biconnectivity

In this section, we will show a biconnectivity algorithm. It is a modification of the algorithm proposed by [36]. The high level idea is to construct a new graph G' based on the input graph G, and reduce the biconnectivity problem of G to the connectivity problem of G'. Since the running time of the connectivity algorithm [4] depends on the diameter of the graph, we also give an analysis of the diameter of the graph G'.

Algorithm 1 2-Edge Connectivity Algorithm.

- **Input**:
  - A connected undirected graph $G = (V, E)$.
- **Output**:
  - A subset of edges $B \subseteq E$.
- **Finding bridges** (BRIDGES($G = (V, E)$)):
  1. Compute a rooted spanning tree of G. The spanning tree is represented by a set of parent pointers $\text{par} : V \to V$.
  2. Compute $\text{lev} : V \to \mathbb{Z}_{\geq 0}$: for each $v \in V$,
  $$\text{lev}(v) \leftarrow \min\left(\text{dep}_{\text{par}}(v), \min_{w \in V \setminus \{\text{par}(v)\} : (v,w) \in E} \text{dep}_{\text{par}}(\text{the LCA of } (v, w))\right).$$
  3. Compute the DFS sequence A of par.
  4. Initialize $B \leftarrow \emptyset$. For each non-root vertex v, let a_i, a_j be the first and the last appearance of v in A respectively. If $\min_{k:i \leq k \leq j} \text{lev}(a_k) \geq \text{dep}_{\text{par}}(v)$, $B \leftarrow B \cup \{(v, \text{par}(v))\}$. Output B.

Algorithm 2 Biconnectivity Algorithm.

- **Input**:
  - A connected undirected graph $G = (V, E)$.
- **Output**:
  - A coloring $\text{col} : E \to V$ of the edges.
- **Finding blocks** (BICONN($G = (V, E)$)):
  1. Compute a rooted spanning tree of G. The spanning tree is represented by a set of parent pointers $\text{par} : V \to V$.
  2. Compute $\text{lev} : V \to \mathbb{Z}_{\geq 0}$: for each $v \in V$,
  $$\text{lev}(v) \leftarrow \min\left(\text{dep}_{\text{par}}(v), \min_{w \in V \setminus \{\text{par}(v)\} : (v,w) \in E} \text{dep}_{\text{par}}(\text{the LCA of } (v, w))\right).$$
  3. Compute the DFS sequence A of par.
  4. Let r be the root of par. Initialize $V' \leftarrow V \setminus \{r\}, E' \leftarrow \emptyset$.
  5. For each $v \in V'$, let a_i, a_j be the first and the last appearance of v in A respectively. If $\min_{k \in \{i, i+1, \cdots, j\}} \text{lev}(a_k) < \text{dep}_{\text{par}}(\text{par}(v))$, $E' \leftarrow E' \cup \{(v, \text{par}(v))\}$.
  6. For each $(u, v) \in E$, if neither u nor v is the LCA of (u, v) in par, $E' \leftarrow E' \cup \{(u, v)\}$.
  7. Compute the connected components of $G' = (V', E')$. Let $\text{col}' : V' \to V'$ be the coloring of the vertices in V' such that $\forall u', v' \in V'$, u', v' are in the same connected component in $G' \Leftrightarrow \text{col}'(u') = \text{col}'(v')$.
  8. Initialize $\text{col} : E \to V$. For each $e = (u, v) \in E$, if $\text{dep}_{\text{par}}(u) \geq \text{dep}_{\text{par}}(v)$, set $\text{col}(e) \leftarrow \text{col}'(u)$; otherwise, set $\text{col}(e) \leftarrow \text{col}'(v)$. Output $\text{col} : E \to V$.

▶ **Lemma 8** (Biconnectivity). *Consider an undirected graph $G = (V, E)$. Let* $\mathrm{col} : E \to V$ *be the output of* $\textsc{Biconn}(G)$. *Then* $\forall e, e' \in E, e \neq e'$, col *satisfies* $\mathrm{col}(e) = \mathrm{col}(e') \Leftrightarrow$ *there is a simple cycle in G which contains both e and e'. Furthermore, the diameter of the graph G' constructed by* $\textsc{Biconn}(G)$ *is at most* $O(\mathrm{dep}(\mathrm{par}) \cdot \text{bi-diam}(G))$, *the number of vertices of G' is at most $|V|$, and the number of edges of G' is at most $|E|$.*

Algorithm 3 Leaf Sampling Algorithm for DFS Sequence.

- **Pre-determined**:
  - A threshold value s. //s will be the local memory size in the MPC model.
- **Input**:
  - A rooted tree represented by a set of parent pointers $\mathrm{par} : V \to V$ on a set V of n vertices (i.e., par has a unique root r).
- **Output**:
  - The DFS sequence of the rooted tree represented by par.
- **Leaf sampling algorithm** ($\textsc{LeafSampling}(s, \mathrm{par} : V \to V)$):

1. If $n \leq s$, return the DFS sequence of par directly.
2. Set $t \leftarrow \Theta(s^{1/3} \log n)$, $L \leftarrow \mathrm{leaves}(\mathrm{par})$.
3. Each $v \in L$ is independently chosen with probability $p = \min(1, t/|L|)$, and let $S = \{l_1, l_2, \cdots, l_k\}$ be the set of samples. If $|S|^2 > s$, output FAIL.
4. For every pair of sampled leaves $x, y \in S$ with $x \neq y$, find the least common ancestor $p_{x,y}$ of (x, y), and set $p_{xy,x}$, $p_{xy,y}$ to be two children of $p_{x,y}$ such that $p_{xy,x}$ is an ancestor of x and $p_{xy,y}$ is an ancestor of y.
5. Sort $l_1, l_2, \cdots, l_k \in S$ such that $\forall i < j \in [k]$, $\mathrm{rank}(p_{l_i l_j, l_i}) < \mathrm{rank}(p_{l_i l_j, l_j})$.
6. Find the paths $A'_1 = P(r, l_1), A'_2 = P(\mathrm{par}(l_1), p_{l_1,l_2}), A'_3 = P(p_{l_1 l_2, l_2}, l_2), \cdots, A'_{2k-2} = P(\mathrm{par}(l_{k-1}), p_{l_{k-1} l_k}), A'_{2k-1} = P(p_{l_{k-1} l_k, l_k}, l_k), A'_{2k} = P(l_{2k}, r)$, i.e., the paths: $r \to l_1 \to$ the LCA of $(l_1, l_2) \to l_2 \to \cdots \to l_{k-1} \to$ the LCA of $(l_{k-1}, l_k) \to l_k \to r$.
7. Set $A' \leftarrow A'_1 A'_2 \cdots A'_{2k}$, i.e., A' is the concatenation of $A'_1, A'_2, \cdots, A'_{2k}$.
8. For each element a'_i in the i^{th} $(i > 1)$ position of the sequence A',
   - if the vertex a'_i is a leaf, keep a'_i as a single copy;
   - Otherwise,
     - if $a'_{i-1} = \mathrm{par}(a'_i)$, i.e., i is the first position that the vertex a'_i appears in A', split a'_i into $\mathrm{rank}(a'_{i+1})$ copies; //a'_{i+1} is a child of a'_i.
     - if $a'_{i-1}, a'_{i+1} \in \mathrm{child}(a'_i)$, split a'_i into $\mathrm{rank}(a'_{i+1}) - \mathrm{rank}(a'_{i-1})$ copies;
     - if $a'_{i+1} = \mathrm{par}(a'_i)$, i.e., i is the last position that the vertex a'_i appears in A', split a'_i into $|\mathrm{child}(a'_i)| - \mathrm{rank}(a'_{i-1})$ copies. //a'_{i-1} is a child of a'_i.

   Let A'' be the result sequence.
9. For each $v \in V$, if $\mathrm{par}(v)$ appears in A'' but v does not appear in A'', recursively find the DFS sequence of the subtree of v, and insert the such sequence into the position after the $\mathrm{rank}(v)^{\text{th}}$ appearance of $\mathrm{par}(v)$ in A''. Output the final result sequence A.

4 An Offline DFS Sequence Algorithm in Linear Space

In Section 4.1, we will review an algorithmic framework proposed by [4] for the DFS sequence. In Section 4.2, 4.3, 4.4, we will discuss the subroutines needed for our DFS sequence algorithm in the offline setting.

4.1 DFS Sequence via Leaf Sampling

In the following, we review the leaf sampling algorithmic framework proposed by [4] for finding the DFS sequence of a rooted tree.

▶ **Theorem 9** (Leaf sampling algorithm [4]). *Consider a set of parent pointers* $\text{par} : V \to V$ *on a set* V *of* n *vertices. Suppose* par *has a unique root. For any* $\gamma \geq 0$ *and any constant* $\delta \in (0, 1)$*, if both of step 4 and step 6 in* LeafSampling(n^δ, par) *can be implemented in the* (γ, δ)*-MPC model with* $O(\log(\text{dep}(\text{par})))$ *parallel time, then the leaf sampling algorithm with parameter* $s = n^\delta$ *on input* $\text{par} : V \to V$ *can be implemented in the* (γ, δ)*-MPC model. Furthermore, with probability at least* 0.99*,* LeafSampling(n^δ, par) *can output the DFS sequence of* par *in* $O(\log(\text{dep}(\text{par})))$ *parallel time. If the algorithm fails, then it returns FAIL.*

By Theorem 9, we only need to give a linear total space MPC algorithm for the LCA problem and the path generation problem to design an efficient DFS sequence algorithm in the $(0, \delta)$-MPC model.

In [4], they proposed to use doubling algorithms to compute the LCA and generate the paths. Since they need to store the every 2^i-th ancestor for each vertex, the total space needed is $\Theta(n \cdot \log(\text{the depth of the tree}))$. We show that we only need to apply the doubling algorithm for a compressed tree, instead of applying it for the original tree.

Algorithm 4 Construction of a Compressed Rooted Tree.

- **Input**:
  - A rooted tree represented by a set of parent pointers $\text{par} : V \to V$ on a set V of n vertices (par has a unique root r).
- **Output**:
  - A vertex set $V' \subseteq V$, a set of parent pointers $\text{par}' : V' \to V'$ on V'.
- **Tree compression** (Compress$(\text{par} : V \to V)$):
  1. Compute the depth of par, the depth of each vertex and set $d \leftarrow \text{dep}(\text{par})$, $t \leftarrow \lceil \log d \rceil$.
  2. $V' \leftarrow \{v \in V \mid \text{dep}_{\text{par}}(v) \mod t = 0, \text{dep}_{\text{par}}(v) + t \leq d\}$.
  3. Initialize $\text{par}' : V' \to V'$. For each $v \in V'$, $\text{par}'(v) \leftarrow \text{par}^{(t)}(v)$.
  4. Output V', par′.

4.2 Compressed Rooted Tree

Given a set of parent pointers $\text{par} : V \to V$, we will show how to compress the rooted tree represented by par.

▶ **Lemma 10** (Properties of a compressed rooted tree). *Let* $\text{par} : V \to V$ *be a set of parent pointers on a vertex set* V *with* $|V| > 1$*, and* par *has a unique root. Let* $t = \lceil \log(\text{dep}(\text{par})) \rceil$ *and let* $(V', \text{par}') =$Compress(par)*. Then it has the following properties:*

1. $|V'| \leq |V| / \log(\text{dep}(\text{par}))$.
2. $\forall v \in V', i \in \mathbb{N}$, $\text{par}'^{(i)}(v) = \text{par}^{(i \cdot t)}(v) \in V'$.
3. $\forall v \in V$, $\exists i \in \{0, 1, \cdots, 2t\}$, *such that* $\text{par}^{(i)}(v) \in V'$.

4.3 Least Common Ancestor

Given a rooted tree represented by a set of parent pointers $\text{par} : V \to V$ on a vertex set V, and a set of q queries $Q = \{(u_1, v_1), (u_2, v_2), \cdots, (u_q, v_q)\}$ where $\forall i \in [q], u_i \neq v_i, u_i, v_i \in \text{leaves}(\text{par})$, we show a space efficient algorithm which can output the LCA of each queried

Algorithm 5 Lowest Common Ancestor.

- **Input**:
  - A rooted tree represented by a set of parent pointers $\text{par} : V \to V$ on a set V of n vertices (par has a unique root r), and a set of q queries $Q = \{(u_1, v_1), (u_2, v_2), \cdots, (u_q, v_q)\}$ where $\forall i \in [q], u_i \neq v_i, u_i, v_i \in \text{leaves(par)}$.
- **Output**:
  - $\text{lca} : Q \to V \times V \times V$.
- **Finding LCA** ($\text{LCA}(\text{par} : V \to V, Q)$):
  1. $(V', \text{par}') \leftarrow$COMPRESS(par). //(see Lemma 10).
  2. Set $d \leftarrow \text{dep(par)}, t \leftarrow \lceil \log d \rceil$ and compute mappings $g_0, g_1, \cdots g_t : V' \to V'$ such that $\forall v \in V', j \in \{0, 1, \cdots, t\}$, $g_j(v) = \text{par}'^{(2^j)}(v)$.
  3. For each query $(u_i, v_i) \in Q$: //Suppose $\text{dep}_{\text{par}}(u_i) \geq \text{dep}_{\text{par}}(v_i)$.
     - **a.** If $\text{dep}_{\text{par}}(u_i) > \text{dep}_{\text{par}}(v_i) + 2t$, find an ancestor $\widehat{u}_i$ of u_i in par such that $\text{dep}_{\text{par}}(\widehat{u}_i) \leq \text{dep}_{\text{par}}(v_i) + 2t$ and $\text{dep}_{\text{par}}(\widehat{u}_i) \geq \text{dep}_{\text{par}}(v_i)$. Otherwise, $\widehat{u}_i \leftarrow u_i$.
     - **b.** If $\exists j \in [4t]$ $\text{par}^{(j)}(\widehat{u}_i)$ is the LCA of $(\widehat{u}_i, v_i)$ in par, set $\text{lca}(u_i, v_i) = (\text{par}^{(j)}(\widehat{u}_i), x, y)$ where x, y are children of $\text{par}^{(j)}(\widehat{u}_i)$ and x, y are ancestors of $\widehat{u}_i, v_i$ respectively. The query of (u_i, v_i) is finished.
     - **c.** Find an ancestor u_i' of $\widehat{u}_i$ in par such that u_i' is the closest vertex to $\widehat{u}_i$ in V', i.e., $\text{dep}_{\text{par}}(\widehat{u}_i) - \text{dep}_{\text{par}}(u_i')$ is minimized. Similarly, find an ancestor v_i' of v_i in par such that v_i' is the closest vertex to v_i in V', i.e., $\text{dep}_{\text{par}}(v_i) - \text{dep}_{\text{par}}(v_i')$ is minimized.
     - **d.** Find $u_i'' \neq v_i'' \in V'$ such that they are ancestors of u_i' and v_i' respectively, and $\text{par}'(u_i'') = \text{par}'(v_i'')$ is the LCA of (u_i', v_i') in par'.
     - **e.** Find the smallest $j \in [2t]$ such that $\text{par}^{(j)}(u_i'') = \text{par}^{(j)}(v_i'')$. Set $\text{lca}(u_i, v_i) = (\text{par}^{(j)}(u_i''), \text{par}^{(j-1)}(u_i''), \text{par}^{(j-1)}(v_i''))$.

pair of vertices. Notice that the assumption that queries only contain leaves is without loss of generality: we can attach an additional child vertex v to each non-leaf vertex u. Thus, v is a leaf vertex. When a query contains u, we can use v to replace u in the query, and the result will not change.

Before we analyze the algorithm $\text{LCA}(\text{par}, Q)$, let us discuss some details of the algorithm.

1. We pre-compute $\text{dep}_{\text{par}}(v)$ and $\text{dep}_{\text{par}'}(u)$ for every $v \in V$ and $u \in V'$.
2. To implement step 3a, we firstly check whether $\text{dep}_{\text{par}}(u_i) > \text{dep}_{\text{par}}(v_i) + 2t$. If it is not true, we can set $\widehat{u}_i$ to be u_i directly. Otherwise, according to Lemma 10, there is a $j \in \{0, 1, \cdots, 2t\}$ such that $\text{par}^{(j)}(u_i) \in V'$. Since $\text{dep}_{\text{par}}(u_i) > \text{dep}_{\text{par}}(v_i) + 2t$, $\text{dep}_{\text{par}}(\text{par}^{(j)}(u_i)) > \text{dep}_{\text{par}}(v_i)$. We initialize $\widehat{u}_i$ to be $\text{par}^{(j)}(u_i) \in V'$. For $k = t \to 0$, if $\text{dep}_{\text{par}}(g_k(\widehat{u}_i)) > \text{dep}_{\text{par}}(v_i)$ (i.e., $\text{dep}_{\text{par}}(\text{par}'^{(2^k)}(\widehat{u}_i)) > \text{dep}_{\text{par}}(v_i)$), we set $\widehat{u}_i \leftarrow g_k(\widehat{u}_i) = \text{par}'^{(2^k)}(\widehat{u}_i)$. Due to Lemma 10 again, the final $\widehat{u}_i$ must satisfy $\text{dep}_{\text{par}}(\widehat{u}_i) \geq \text{dep}_{\text{par}}(v_i)$ and $\text{dep}_{\text{par}}(\widehat{u}_i) \leq \text{dep}_{\text{par}}(v_i) + 2t$. This step takes time $O(t)$.

▶ **Lemma 11** (LCA algorithm). *Let* $\text{par} : V \to V$ *be a set of parent pointers on a vertex set* V. par *has a unique root. Let* $Q = \{(u_1, v_1), (u_2, v_2), \cdots, (u_q, v_q)\}$ *be a set of* q *pairs of vertices where* $\forall i \in [q], u_i \neq v_i, u_i, v_i \in \text{leaves(par)}$. *Let* $\text{lca} : Q \to V \times V \times V$ *be the output of* $\text{LCA}(\text{par}, Q)$. *For* $(u_i, v_i) \in Q$, $(p_i, p_{i,u_i}, p_{i,v_i}) = \text{lca}(u_i, v_i)$ *satisfies that* p_i *is the LCA of* (u_i, v_i), p_{i,u_i}, p_{i,v_i} *are ancestors of* u_i, v_i *respectively, and* p_{i,u_i}, p_{i,v_i} *are children of* p_i. *Furthermore, the space used by the algorithm is at most* $O(|Q| + |V|)$.

4.4 Multi-Paths Generation

Consider a rooted tree represented by a set of parent pointers $\text{par} : V \to V$ on a vertex set V and a set of q vertex-ancestor pairs $Q = \{(u_1, v_1), (u_2, v_2), \cdots, (u_q, v_q)\}$ where $\forall i \in [q]$, v_i is an ancestor of u_i. We show a space efficient algorithm MultiPaths(par, Q) which can generate all the paths $P(u_1, v_1), P(u_2, v_2), \cdots, P(u_q, v_q)$.

Algorithm 6 Multi-Paths Generation.

- **Input**:
  - A rooted tree represented by a set of parent pointers $\text{par} : V \to V$ on a set V of n vertices (par has a unique root r), and a set of q vertex-ancestor pairs $Q = \{(u_1, v_1), (u_2, v_2), \cdots, (u_q, v_q)\}$ where $\forall i \in [q], v_i$ is an ancestor of u_i.
- **Output**:
  - $P_1, P_2, \cdots, P_q$.
- **Generating multiple path sequences** (MultiPaths$(\text{par} : V \to V, Q)$):
  1. $(V', \text{par}') \leftarrow$Compress(par). //(see Lemma 10).
  2. Set $d \leftarrow \text{dep}(\text{par}), t \leftarrow \lceil \log d \rceil$ and compute mappings $g_0, g_1, \cdots g_t : V' \to V'$ such that $\forall v \in V', j \in \{0, 1, \cdots, t\}$, $g_j(v) = \text{par}'^{(2^j)}(v)$.
  3. For each vertex-ancestor pair $(u_i, v_i) \in Q$:
     a. If $\text{dep}_{\text{par}}(u_i) - \text{dep}_{\text{par}}(v_i) \leq 2t$, generate the path sequence $P_i = (u_i, \text{par}^{(1)}(u_i), \text{par}^{(2)}(u_i), \cdots, v_i)$ directly.
     b. Otherwise, find the minimum $j \in [2t]$ such that $\text{par}^{(j)}(u_i) \in V'$. Set $u'_i \leftarrow \text{par}^{(j)}(u_i)$. Find an ancestor v'_i of u'_i in par$'$ such that $\text{dep}_{\text{par}}(v'_i) \geq \text{dep}_{\text{par}}(v_i)$ and $\text{dep}_{\text{par}}(v'_i) - 2t \leq \text{dep}_{\text{par}}(v_i)$.
     c. Generate the path $P'(u'_i, v'_i)$ in par$'$.
     d. Initialize a sequence A as the concatenation of (u_i), $P'(u'_i, v'_i)$ and (v_i).
     e. Repeat: for each element a_i in A, if a_i is not the last element and $a_{i+1} \neq \text{par}(a_i)$, insert $\text{par}(a_i)$ between a_i and a_{i+1}; until A does not change. Output the final sequence A as the path sequence P_i.

Before we analyze the correctness of the algorithm, let us discuss some details.

1. In step 3a, if the length of the path is at most $2t$, then we can generate the path in $O(t)$ rounds. In the j-th round, we can find the vertex $\text{par}^{(j)}(u_i) = \text{par}(\text{par}^{(j-1)}(u_i))$.
2. In step 3b, we want to find v'_i. We initialize v'_i as u'_i. For $k = t \to 0$, if $\text{dep}_{\text{par}}(g_k(v'_i)) > \text{dep}_{\text{par}}(v_i)$ (i.e., $\text{dep}_{\text{par}}(\text{par}'^{(2^k)}(v'_i)) > \text{dep}_{\text{par}}(v_i)$), we set $v'_i \leftarrow g_k(v'_i) = \text{par}'^{(2^k)}(v'_i)$.

▶ **Lemma 12** (Generation of multiple paths). *Let* $\text{par} : V \to V$ *be a set of parent pointers on a vertex set* V. par *has a unique root. Let* $Q = \{(u_1, v_1), (u_2, v_2), \cdots, (u_q, v_q)\} \subseteq V \times V$ *be a set of pairs of vertices where* $\forall j \in [q]$, v_j *is an ancestor of* u_j *in* par. *Let* $P_1, P_2, \cdots, P_q$ *be the output of* MultiPaths(par, Q). *Then* $\forall j \in [q], P_j = P(u_j, v_j)$, *i.e.,* P_j *is a sequence which denotes a path from* u_j *to* v_j *in* par. *Furthermore, the space used by the algorithm is at most* $O(|V| + \sum_{j \in [q]} |P_j|)$.

5 Hardness of Biconnectivity in MPC

There is a conjectured hardness which is widely used in the MPC literature [25, 11, 28, 34, 40].

▷ Conjecture 1 (1-cycle vs. 2-cycles). For any $\gamma \geq 0$ and any constant $\delta \in (0, 1)$, distinguishing the following two instances in the (γ, δ)-MPC model requires $\Omega(\log n)$ parallel time:

1. a single cycle contains n vertices,
2. two disjoint cycles, each contains $n/2$ vertices.

Under the above conjecture, we show that $\Omega(\log \text{bi-diam}(G))$ parallel time is necessary to compute the biconnected components of G. This claim is true even for the constant diameter graph G, i.e., $\text{diam}(G) = O(1)$.

▶ **Theorem 13** (Hardness of biconnectivity in MPC). *For any $\gamma \geq 0$ and any constant $\delta \in (0, 1)$, unless the one cycle vs. two cycles conjecture (Conjecture 1) is false, any (γ, δ)-MPC algorithm requires $\Omega(\log \text{bi-diam}(G))$ parallel time for testing whether a graph G with a constant diameter is biconnected.*

Proof. For $\gamma \geq 0$ and an arbitrary constant $\delta \in (0, 1)$, suppose there is a (γ, δ)-MPC algorithm $\mathcal{A}$ which can determine whether an arbitrary constant diameter graph G is biconnected in $o(\log \text{bi-diam}(G))$ parallel time. Then we give a (γ, δ)-MPC algorithm for solving one cycle vs. two cycles problem as the following:

1. For a one cycle vs. two cycles instance n-vertex graph $G' = (V', E')$, construct a new graph $G = (V, E)$: $V = V' \cup \{v^*\}, E = E' \cup \{(v, v^*) \mid v \in V'\}$.
2. Run $\mathcal{A}$ on G. If G is not biconnected, G' has two cycles. Otherwise G' is a single cycle.

It is easy to see that the diameter of G is 2. If G' is a single cycle, then G is biconnected and $\text{bi-diam}(G) = \Theta(n)$. If G' contains two cycles, then G contains two biconnected components and $\text{bi-diam}(G) = \Theta(n)$.

The first step of the above algorithm takes $O(1)$ parallel time and only requires linear total space. The graph G has $n + 1$ vertices and $2n$ edges. Thus, the above algorithm is also a (γ, δ)-MPC algorithm. The parallel time of the above algorithm is the same as the time needed for running $\mathcal{A}$ on G which is $o(\log \text{bi-diam}(G)) = o(\log n)$. Thus the existence of the algorithm $\mathcal{A}$ implies that the one cycle vs. two cycles conjecture (Conjecture 1) is false. ◀

References

1 Kook Jin Ahn and Sudipto Guha. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. *ACM Transactions on Parallel Computing (TOPC)*, 4(4):17, 2018.

2 Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of algorithms*, 7(4):567–583, 1986.

3 Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 574–583. ACM, 2014.

4 Alexandr Andoni, Zhao Song, Clifford Stein, Zhengyu Wang, and Peilin Zhong. Parallel Graph Connectivity in Log Diameter Rounds, 2018. In *FOCS 2018*. `arXiv:1805.03055`.

5 Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab Mirrokni, and Cliff Stein. Coresets meet EDCS: algorithms for matching and vertex cover on massive graphs. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1616–1635. SIAM, 2019.

6 Sepehr Assadi and Sanjeev Khanna. Randomized composable coresets for matching and vertex cover. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 3–12. ACM, 2017.

7 Sepehr Assadi, Xiaorui Sun, and Omri Weinstein. Massively Parallel Algorithms for Finding Well-Connected Components in Sparse Graphs. *arXiv preprint*, 2018. `arXiv:1805.02974`.

8 Giorgio Ausiello, Donatella Firmani, Luigi Laura, and Emanuele Paracone. Large-scale graph biconnectivity in MapReduce. *Department of Computer and System Sciences Antonio Ruberti Technical Reports*, 4(4), 2012.

9 Boaz Barak, Jonathan A Kelner, and David Steurer. Dictionary learning and tensor decomposition via the sum-of-squares method. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing (STOC)*, pages 143–151. ACM, 2015. `arXiv:1407.1543`.

10 Paul Beame and Johan Hastad. Optimal bounds for decision problems on the CRCW PRAM. *Journal of the ACM (JACM)*, 36(3):643–670, 1989.

11 Paul Beame, Paraschos Koutris, and Dan Suciu. Communication steps for parallel query processing. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI symposium on Principles of database systems*, pages 273–284. ACM, 2013.

12 Soheil Behnezhad, Mahsa Derakhshan, and MohammadTaghi Hajiaghayi. Brief announcement: Semi-mapreduce meets congested clique. *arXiv preprint*, 2018. `arXiv:1802.10297`.

13 Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, and Richard M Karp. Massively parallel symmetry breaking on sparse graphs: MIS and maximal matching. *arXiv preprint*, 2018. `arXiv:1807.06701`.

14 Sebastian Brandt, Manuela Fischer, and Jara Uitto. Matching and MIS for Uniformly Sparse Graphs in the Low-Memory MPC Model. *arXiv preprint*, 2018. `arXiv:1807.05374`.

15 Artur Czumaj, Jakub Łącki, Aleksander Mądry, Slobodan Mitrović, Krzysztof Onak, and Piotr Sankowski. Round compression for parallel matching algorithms. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 471–484. ACM, 2018.

16 Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *To appear in OSDI*, page 1, 2004.

17 Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

18 Reinhard Diestel. *Graph theory*. Springer Publishing Company, Incorporated, 2018.

19 Alina Ene, Sungjin Im, and Benjamin Moseley. Fast clustering using MapReduce. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 681–689. ACM, 2011.

20 Manuela Fischer, Mohsen Ghaffari, and Jara Uitto. Simple Graph Coloring Algorithms for Congested Clique and Massively Parallel Computation. *arXiv preprint*, 2018. `arXiv:1808.08419`.

21 Michael T Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, Searching, and Simulation in the MapReduce Framework. In *ISAAC*, volume 7074, pages 374–383. Springer, 2011.

22 Sungjin Im, Benjamin Moseley, and Xiaorui Sun. Efficient massively parallel methods for dynamic programming. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 798–811. ACM, 2017.

23 Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. In *ACM SIGOPS operating systems review*, volume 41 (3), pages 59–72. ACM, 2007.

24 Tomasz Jurdziński and Krzysztof Nowicki. MST in O(1) rounds of congested clique. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2620–2632. SIAM, 2018.

25 Howard Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for MapReduce. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 938–948. Society for Industrial and Applied Mathematics, 2010.

26 Richard M Karp, Eli Upfal, and Avi Wigderson. Constructing a perfect matching is in random NC. *Combinatorica*, 6(1):35–48, 1986.

27 Valerie King, Chung Keung Poon, Vijaya Ramachandran, and Santanu Sinha. An optimal EREW PRAM algorithm for minimum spanning tree verification. *Information Processing Letters*, 62(3):153–159, 1997.

28 Raimondas Kiveris, Silvio Lattanzi, Vahab Mirrokni, Vibhor Rastogi, and Sergei Vassilvitskii. Connected components in mapreduce and beyond. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 1–13. ACM, 2014.

29 Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, and Sergei Vassilvitskii. Filtering: a method for solving graph problems in mapreduce. In *Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures*, pages 85–94. ACM, 2011.

30 Sixue Liu and Robert E Tarjan. Simple Concurrent Labeling Algorithms for Connected Components. *arXiv preprint*, 2018. `arXiv:1812.06177`.

31 Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM journal on computing*, 15(4):1036–1053, 1986.

32 Krzysztof Onak. Round compression for parallel graph algorithms in strongly sublinear space. *arXiv preprint*, 2018. `arXiv:1807.08745`.

33 John H Reif. Optimal Parallel Algorithms for Interger Sorting and Graph Connectivity. Technical report, HARVARD UNIV CAMBRIDGE MA AIKEN COMPUTATION LAB, 1985.

34 Tim Roughgarden, Sergei Vassilvitskii, and Joshua R Wang. Shuffles and circuits:(on lower bounds for modern parallel computation). In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 1–12. ACM, 2016.

35 Yossi Shiloach and Uzi Vishkin. An O (log n) parallel connectivity algorithm. Technical report, Computer Science Department, Technion, 1980.

36 Robert E Tarjan and Uzi Vishkin. An efficient parallel biconnectivity algorithm. *SIAM Journal on Computing*, 14(4):862–874, 1985.

37 Robert Endre Tarjan and Uzi Vishkin. Finding biconnected componemts and computing tree functions in logarithmic parallel time. In *25th Annual Symposium onFoundations of Computer Science, 1984.*, pages 12–20. IEEE, 1984.

38 Leslie G Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990.

39 Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing (STOC)*, pages 887–898. ACM, 2012.

40 Grigory Yaroslavtsev and Adithya Vadapalli. Massively Parallel Algorithms and Hardness for Single-Linkage Clustering under Lp Distances. In *International Conference on Machine Learning*, pages 5596–5605, 2018.

41 Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.

Two Party Distribution Testing: Communication and Security

Alexandr Andoni
Columbia University, New York City, NY, USA

Tal Malkin
Columbia University, New York City, NY, USA

Negev Shekel Nosatzki
Columbia University, New York City, NY, USA

Abstract

We study the problem of discrete distribution testing in the *two-party setting*. For example, in the standard closeness testing problem, Alice and Bob each have t samples from, respectively, distributions a and b over $[n]$, and they need to test whether $a = b$ or a, b are ϵ-far (in the ℓ_1 distance). This is in contrast to the well-studied one-party case, where the tester has unrestricted access to samples of both distributions. Despite being a natural constraint in applications, the two-party setting has previously evaded attention.

We address two fundamental aspects of the two-party setting: 1) what is the communication complexity, and 2) can it be accomplished securely, without Alice and Bob learning extra information about each other's input. Besides closeness testing, we also study the independence testing problem, where Alice and Bob have t samples from distributions a and b respectively, which may be correlated; the question is whether a, b are independent or ϵ-far from being independent. Our contribution is three-fold: 1) We show how to gain **communication efficiency** given more samples, beyond the information-theoretic bound on t. The gain is polynomially better than what one would obtain via adapting one-party algorithms. 2) We prove *tightness* of our trade-off for the closeness testing, as well as that the independence testing requires tight $\Omega(\sqrt{m})$ communication for unbounded number of samples. These **lower bounds** are of independent interest as, to the best of our knowledge, these are the first 2-party communication lower bounds for testing problems, where the inputs are a set of *i.i.d. samples*. 3) We define the concept of **secure distribution testing**, and provide secure versions of the above protocols with an overhead that is only polynomial in the security parameter.

2012 ACM Subject Classification Mathematics of computing → Hypothesis testing and confidence interval computation

Keywords and phrases distribution testing, communication complexity, security

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.15

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1811.04065`.

Acknowledgements We thank Devanshi Nishit Vyas for her contribution to some of the initial work which led to this paper. We thank Clement Canonne for invaluable comments on an early draft of the manuscript. We thank Yuval Ishai for helpful discussions. Work supported in part by Simons Foundation (#491119), NSF grants CCF-1617955 and CCF-1740833.

1 Introduction

Distribution property testing is a sub-area of statistical hypothesis testing, which has enjoyed continuously growing interest in the theoretical computer science community, especially since the 2000 papers [36, 12]. One of the most basic problems is closeness testing, also known as the *homogeneity testing* ; see [37, 55, 58]. Here, given two distributions a, b and t

samples from each of them, distinguish between the cases where $a = b$ versus a and b are ϵ-far, which usually means $\|a - b\|_1 > \epsilon$.[1] For this specific problem, the extensive research led to algorithms with optimal sample complexity [12, 59, 13, 24, 31, 29], including when the number of samples from the two distributions is unequal [5, 16, 31]. Further research directions of interest include obtaining instance-optimal algorithms, which depend on further properties of the distributions a, b [3, 4, 31], quantum algorithms [20], as well as algorithms whose output is differentially-private [30, 21, 6, 8]. An even larger body of work studied numerous other related problems; see, e.g., surveys [35, 22, 53, 52].

Focusing on testing two distributions, such as in the closeness problem, a very natural aspect has, surprisingly, evaded attention so far: such a task would often be run by two players, each with access to their own distribution. Specifically, Alice has samples from distribution a, Bob has samples from distribution b, and they need to jointly solve a distribution testing problem on (a, b). This setting models many of the envisioned usage scenarios of distribution testing, where different parties wish to jointly perform a statistical hypothesis testing task on their distributions. For example, [55] describes the scenario where two distinct sensors need to test whether they sample from the same distribution ("noise") or not.

This 2-party setting raises the following standard theoretical challenges, neither of which has been previously studied in the context of distribution testing:

- What is the *communication complexity* of the testing problem? In particular, can we do better than the straightforward approach, where Alice sends her samples to Bob who then runs an offline algorithm? Can we prove matching lower bounds?
  This aspect parallels the quest for low memory or communication usage for hypothesis testing on a *single distribution*, initiated in the statistics community [25, 41] and [7, 40, 10]. In fact, very recent, independent work has considered this aspect for binary sources [54, 38].
- Is it possible to design a distribution testing protocol that is *secure*, i.e., where Alice and Bob learn nothing about each other's samples, besides testing result? This question is highly relevant in today's push for doing statistics in a privacy-respecting manner.

1.1 Our Contributions

In this paper, we initiate the study of testing problems in the two-party model, and design protocols which are both communication-efficient and secure. We do so for two basic problems on pairs of distributions (i.e., where the two-party setting is natural): 1) closeness testing, and 2) independence testing.

Our main finding is that, once the number of samples exceeds the information-theoretic minimum, we can obtain protocols with polynomially smaller communication than the naïve adaptation of existing algorithms. We complement our protocols with lower bounds on the communication complexity of such problems that are near-optimal for closeness testing, as well as for independence for an unbounded number of samples. Our upper and lower bounds on communication are novel even without any security considerations.

To argue security, we also put forth a definition for secure distribution testing in the multi-party model. Our definition differs from the standard secure computation setting due to two unique features of the considered setting. First, this is "testing" (a promise problem) and not "computing"; second, the function of interest is defined with respect to distributions, while the parties' inputs are samples. These features do not come into play if the distributions satisfy the promise (e.g., they are either identical or ϵ-far), in which case

[1] This is equivalent to saying that the total variation distance is more than $\epsilon/2$.

the security guarantee matches the standard cryptographic one (no information is leaked beyond the output). However, when the promise is not satisfied, we need to allow for some information on the parties' samples to be leaked by the protocol. Our definition permits leakage of at most one bit in this case, and leaks nothing when the promise is satisfied.

▶ **Definition 1** (Security Definition). *Let D be the set of input distributions over $\bigtimes_{i=1}^{d}[n_i]$, and let $g : D \to \{0,1\}$ be a partial boolean function, defined on $P \subseteq D$.*

Let π be a d-party protocol, and let k be a security parameter. We say that π is a t-sample secure distribution testing protocol *(for the testing task defined by g), if there exists a boolean function $f : \{\bigtimes_{i=1}^{d}[n_i]\}^t \to \{0,1\}$ such that the following holds:*

Correctness: *for any $p \in P$, $\Pr_{\zeta_1 \dots \zeta_t \sim_{i.i.d} p}[f(\zeta) = g(p)] = 1 - neg(k)$*

Security: *For any $\zeta \in \{\bigtimes_{i=1}^{d}[n_i]\}^t$, if we give each player $i \in [d]$ the input $\left(1^k, \zeta_1(i), \dots, \zeta_t(i)\right)$, then protocol π is a secure computation of the function $f(\zeta)$.*

We provide a detailed discussion of the above definition in the full version of this paper.

Closeness Testing. In the *2-party closeness testing* problem $2\text{PCT}_{n,t,\epsilon}$, Alice and Bob each have access to t samples from some distributions respectively a, b over alphabet $[n]$. Their goal is to distinguish between $a = b$ and $\|a - b\|_1 \geq \epsilon$ with probability $\geq 2/3$.

We first give a non-secure near-optimal communication protocol, and then show how to make it secure with only a small overhead (polynomial in the security parameter). Our secure version is based on the existence of a PRG that stretches from $\text{polylog}(m)$ bits to m bits, and of an OT protocol with polylog communication. Overall, we prove the following.

▶ **Theorem** (Closeness, Secure). *Fix a security parameter $k > 1$. Fix $n > 1$ and $\epsilon \in (0, 2)$, and let t be such that $t \geq C \cdot k \cdot \max\left(n^{2/3} \cdot \epsilon^{-4/3}, \sqrt{n} \cdot \epsilon^{-2}\right)$ for some (universal) constant $C > 0$. Then, assuming PRG and OT as above, there exists a* secure distribution testing protocol *for $2\text{PCT}_{n,t,\epsilon}$ which uses $\tilde{O}_k\left(\frac{n^2}{t^2\epsilon^4} + 1\right)$ communication.*

To contrast the communication bounds of our protocol to the classic 1-party setting, consider what happens in the extreme settings of the parameters s, t, for a fixed ϵ. When $t \approx \Theta(n^{2/3})$, the communication is $\tilde{O}(n^{2/3})$ as well, i.e., Alice may as well just send all the samples over to Bob. However the communication decreases as the players have more samples. This may not be surprising given the testing results with unequal number of samples [16, 31]: indeed, Alice can send $\approx \max\{n/\sqrt{t}, \sqrt{n}\}$ samples to Bob, and Bob can run the tester. In contrast, our protocol obtains a *polynomially smaller* complexity, $\approx n^2/t^2$, whenever $t \gg n^{2/3}$. Intuitively, considering the extreme of $t \gg n$, we can obtain near-constant communication: with so many samples, we can *learn* the distribution, and then use *sketching* tools [9, 42].

We prove a *near-tight* lower bound on the above trade-off (even without security considerations) in Section 4. We note that our lower bound differs from the common communication complexity lower bounds as the players' inputs are *i.i.d. samples* and not worst-case.

▶ **Theorem** (Closeness lower bound). *Any two-way communication protocol for $2\text{PCT}_{n,t,1/2}$ requires $\tilde{\Omega}\left(n^2/t^2\right)$ communication.*

Independence Testing. Our second problem is the independence testing problem in the 2-party model, denoted $2\text{PIT}_{n,m,t,\epsilon}$. Let $p = (a, b)$ be some joint distribution over $[n] \times [m]$, where $n \geq m$, and for $i \in [t]$, let ζ_i be a sample drawn from p. Now we provide Alice with the first coordinates of ζ_i's and Bob with the second coordinates. Alice and Bob's goal is to test whether distributions a and b are independent (p is a product distribution) or p is ϵ-far from any product distribution. We prove the following:

▶ **Theorem** (Independence, Secure). *Fix a security parameter $k > 1$. Fix $\epsilon \in (0, 2)$, $1 \leq m \leq n$, and let t be such that $t \geq C \cdot k \cdot \left(n^{2/3}m^{1/3}\epsilon^{-4/3} + \sqrt{nm}/\epsilon^2\right)$, for some (universal) constant C, and assuming OT, there is a secure distribution testing protocol for $2\text{PIT}_{n,m,t,\epsilon}$ using $\tilde{O}_k\left(\frac{n^2 \cdot m}{t^2\epsilon^4} + \frac{n \cdot m}{t\epsilon^4} + \frac{\sqrt{m}}{\epsilon^3}\right)$ bits of communication.*

We note that the lower bound on t from the above theorem is necessary as it is the information-theoretic bound, as proven in [31]. An important qualitative aspect of the communication complexity for 2PIT is that, when the number of samples $t \to \infty$, the protocol uses $\tilde{\Theta}_\epsilon(\sqrt{m})$ bits of communication. This is in contrast to 2PCT, where the communication becomes $\tilde{O}(1)$ for $t \to \infty$. Indeed, we show that $\Omega(\sqrt{m})$ is necessary for one-way protocols for 2PIT. Since our protocol can easily be converted to a one-way (non-secure) protocol, this lower bound is tight for one-way protocols. We conjecture that the bound from the above theorem is near-tight in n, m, t for two-way communication protocols, even without security.

▶ **Theorem** (Independence lower bound). *For $n, t \in \mathbf{N}$, any one-way protocol for $2\text{PIT}_{n,n,t,1}$ requires $\Omega(\sqrt{n})$ bits of communication.*

1.2 Related work

Our work bridges three separate areas and models: distribution testing, streaming/sketching, and secure computation. There's a large body of work in each of these areas. We mention work most relevant to us.

Testing and learning with memory or space constraints. Two-party communication model is tightly connected to the streaming and distributed models, which have received lots of focus in the context of testing and learning questions. As early as in 1960s, [25, 41] considered the hypothesis testing (of one distribution) in the streaming model, where samples are streamed over while keeping small extra space. More recently, much attention has been drawn to streaming (memory) lower bounds for learning problems, such as parity learning [50, 51, 46, 48, 33]. Another direction was to consider *stochastic streaming* problems [26], where the input is generated from a distribution. All these results apply to samples from *one distribution*, and show a (tight) trade-off between number of samples and space complexity.

Another recent avenue is to study such problems in the distributed model, where there are many symmetric players, each with a number of i.i.d. samples from the same distribution. For learning problems (e.g., parameter or density estimation), see, e.g., [19, 28, 27]. We note that since learning is a much harder problem, typically proving lower bounds is easier (e.g., as shown in [28], merely communicating the output requires $\Omega(n)$ communication). In contrast, for testing problems, the output is just one bit. For testing problems (of one distribution), see also the recent (independent) manuscript [2].

None of the lower bounds from the above papers are relevant here as they become vacuous for a 2-party setting. Indeed, when two players have two sets of samples from the *same* distribution, then purely doubling the sample set of a player trivializes the question (she can solve it without communication).

Finally, a very recent, independent works of [54, 38] consider a problem very similar to 2PIT: estimating the correlation of two binary sources ($n = m = 2$) in the two-party model.

Secure approximations. Our results on secure distribution testing can also be seen in the context of the area of secure computation of approximations. This is a framework introduced by [32], allowing to combine the benefits of approximation algorithms and secure computation.

This was considered in different settings [32, 39, 14, 43, 15, 44, 45], but the most relevant to us is private approximation of distance between two input vectors. In particular, for ℓ_2 distance, Alice and Bob each have a vector $a, b \in \mathbb{R}^n$ and want to estimate $\|a - b\|_2$, without revealing any information that does not follow from the ℓ_2 distance itself. For this problem, [43] show that secure protocols are possible with only poly-logarithmic communication complexity.We use some of their techniques in our secure protocols.

Approximation and testing have a similar flavor in that they both trade accuracy for efficiency, in different ways. The security goals are also similar (prevent leakage beyond the intended output). One important difference is that the intended output in secure testing is just the single bit of whether or not the test passed. Thus, for example, when approximating a distance function, even a secure protocol can leak any information that follows from the distance. In contrast, when testing for closeness, if the inputs are either identical or far, the protocol may only reveal this fact, but no other information about what the distance is.

Security and privacy of testing. While we are not aware of any work on secure testing, several recent papers address *differentially-private distribution testing* [30, 21, 6, 8]. Here the privacy guarantee relates to the value of the output after the computation is concluded, requiring it to be differentially-private with respect to the inputs. Our notion of security for distribution testing is different, in the same way that secure computation is different from differentially private computation. While differential privacy (DP) is concerned with what the intended output may leak about the inputs (even if the input came from a single party or the computation is done by a trusted curator), secure 2-party computation is concerned with how to compute an intended output without leaking any information beyond the output itself. The difference in goals is also reflected in the privacy guarantees, which are typically statistical in nature (for DP testing) and provide a non-negligible adversarial advantage. Secure testing protocols rely on cryptographic assumptions and provide negligible advantage.

Even more recently, a stringent model of Locally Differentially Private Testing was proposed [56, 1]. This model provides a stronger notion of differential privacy, where users send noisy samples to an untrusted curator, and the goal is to allow the curator to test the distribution of user inputs (for some property) without learning "too much" about the individual samples. For LDP, the main goal is to optimize the sample complexity as a function of the privacy guarantees. While this notion of privacy also incorporates some privacy of the individual inputs, it is much closer to DP than to our security notion. In addition, both DP and LDP do not provide sub-linear communication (in the sample size, as we achieve here). In fact, their goal is to allow $O(1)$ communication per sample, with minimal sample overhead. In contrast, our protocols provide security "nearly for free" while allowing for faster communication with more samples. Finally, in the case of independence testing, our work assumes samples are *distributed* between the parties who need to test the joint distribution, while in the above work, each data point contains full sample information.

1.3 Our Techniques

We now outline the techniques used to establish our main results. Since our overall contribution is painting a big picture of the 2-party complexity of distribution testing problems, we appeal to a number of diverse tools. First, we design communication-efficient protocols. Second, we argue optimality of our protocols by proving communication complexity lower bounds on the considered problems, which are near-tight in some of the parameter regimes. Third, we show how to transform our protocols into secure protocols, under standard cryptographic assumptions, without further loss in efficiency. All three of these contributions are independently first-of-a-kind, to the best of our knowledge.

Communication-efficient protocols. We start by reducing the testing problem under the ℓ_1 distance to the same problem under the ℓ_2 distance, using now-standard methods of [31, 24]. Here, our main challenge is actually testing under the ℓ_2 distance.

Closeness testing (2pCT) is technically the simpler problem, but it already illustrates some phenomena, how to leverage a larger number of samples to improve communication. To estimate the ℓ_2 distance between the 2 unknown distributions, we compute the ℓ_2 distance approximation between the given *samples* of these distributions. In order to approximate the latter in the 2-party setting, we use the ℓ_2 *sketching* tools [9]. The crux is to show that we can tolerate a cruder ℓ_2 approximation if we are given a larger sample size. Since the complexity of $(1+\alpha)$-approximating the ℓ_2 distance is $\Theta(1/\alpha^2)$, we obtain an improvement in communication that is quadratic in the number of samples.

Independence Testing (2pIT) is more challenging since any distance approximation would need to be established based on the distribution(s) implicitly defined via the *joint samples*, split between Alice and Bob, and hence our approximation techniques above are not sufficient. Instead, we develop a reduction from a large, $[n] \times [m]$, alphabet problem, to a smaller alphabet problem, which can be efficiently solved by communicating fewer samples. This is accomplished by sampling a rectangle of the joint alphabet, and showing that such a process, when combined with the *split-set* technique from [31], generates *sub-distributions* (defined later) which satisfy some nice properties. We then show one can test the original distribution $p = (a, b)$ over a "large" domain of size $[n]\times[m]$ for independence by distinguishing closeness of 2 simulated distributions $\hat{p}, \hat{q}$, defined on a smaller domain of size $[l] \times [m]$, where $l = \tilde{\Theta}_\epsilon(n^3m/t^3 + n^2m/t^2 + 1)$. We show it is possible for Alice and Bob to simulate joint samples from $\hat{p}$ and $\hat{q}$ using $O(1)$ communication per sample, after they have down-sampled letters from one of the marginals.

The trade-off on communication–vs–samples emerges from two compounding effects: 1) balancing the size of the target rectangle with the expected number of available samples over such rectangle; and 2) the additional advantage from a tighter bound on the ℓ_2 norm of $\hat{q}$. Each of the above independently generates linear improvement in communication with more samples. The latter advantage, however, is helpful only while $t = O(n)$, and therefore we benefit from quadratic improvement in that regime, and linear improvement thereafter.

Lower bounds on communication. We note that the lower bounds on communication of testing problems present a particular technical challenge: for testing problems, the inputs are i.i.d. samples from some distributions. This is more akin to the average-case complexity setup, as opposed to "worst case" complexity as is standard for communication lower bounds.

We manage to prove such testing lower bounds for the *Closeness Testing* problem (2pCT). While our lower bound is, at its core, a reduction from some "hard 2-party communication problem", our main contribution is dealing with the above challenge. One may observe that a "hard 2-party communication problem" is hard under a certain input distribution (by Yao's minimax theorem), and hence a reduction algorithm would also produce a hard distribution on the inputs to our problem. However, a priori, it is hard to ensure that the resulting input distribution resembles anything like a set of samples from distributions a, b. For example, the inputs may have statistical quirks that actually depend on whether it is a "close" or "ϵ-far" instance, which a reduction is not able to generate without knowing the output.

At a high level, the role of the "hard problem" is played by a variant of the well-known two-way Gap Hamming Distance (GHD) problem [23, 57]. The known GHD lower-bound variants are insufficient for us precisely because of the above challenge – we need a better control over the actual hard distribution.Therefore, we study the following *Exact GHD*

variant: given $x, y \in \{0,1\}^n$, with $\|x\|_1 = \|y\|_1 = n/2$, distinguish between $\|x-y\|_1 = n/2$ versus $\|x-y\|_1 \in [n/2+\beta, n/2+2\beta]$. We show there exists some $\beta \in [\Omega(\sqrt{n}), O(\sqrt{n \log n})]$ for which communication complexity must be $\tilde{\Omega}(n)$, by adapting the proof of [57].

Using one instance of *Exact GHD*, our reduction performs a careful embedding of this hard instance into the samples from distributions a, b, while patching the set of samples to look like i.i.d. samples from the two distributions. While we don't manage to get the output of the reduction to look precisely like i.i.d. samples from a, b, our reduction produces two sets of size $\mathrm{Poi}(t)$ whose distribution is within a small statistical distance from the distribution of two set of samples that would be drawn from two distributions a and b which are either "equal" (when $\|x-y\|_1 = n/2$) or "far" (when $\|x-y\|_1 \in [n/2+\beta, n/2+2\beta]$).

Note that our proof recovers the standard lower bound of $\tilde{\Omega}(n^{2/3})$ on samples necessary to solve closeness testing (in the vanilla setting), albeit not a tight bound [24, 31].

For *Independence testing* (2pIT), we focus on the lower bound for unbounded number of samples. We argue such a hardness result under one-way communication only. Our $\Omega(\sqrt{m})$ lower bound uses the Boolean Hidden Hypermatching (BHH) problem [60]. We conjecture our entire trade-off for the Independence problem is tight. The proof of this conjecture would have to overcome the challenge of lower bounds for statistical inputs.

Securing the communication protocols. Once low-communication insecure protocols have been designed, one may try to convert the protocols to secure ones using generic cryptographic techniques. The latter includes various techniques for secure computation ([61] and followup work), fully homomorphic encryption ([34] and followup work), or homomorphic secret sharing ([17, 18]). However, a naïve application of such techniques will blow up the communication to be at least linear in the input size, possibly requiring strong assumptions, a high computation complexity, or not being applicable to arbitrary computations. The constraint of low-overhead, among other considerations, requires design of custom protocols.

Our starting point is a technique that falls into the latter category: secure circuits with ROM [49], a technique that can transform an insecure 2-party protocol to a secure one with a minimal blow-up in communication, and uses a weak assumption only (OT). In order to obtain an efficient protocol, however, it only applies to computations expressible via a very small circuit, whose size is proportional to the target communication, with access to a larger read-only-memory (ROM) table. Thus, the main challenge becomes to design two-party testing protocols that fit this required format.

For *Closeness Testing* (2pCT), we begin with our low-communication non-secure protocol, and adapt it to be secure by designing a small circuit. One of the main difficulties in designing such a circuit is that, in the ℓ_1- to ℓ_2-testing reduction, Alice and Bob need to agree on an alphabet, which depends on their inputs, without compromising the inputs themselves. To bypass this, and other issues, we allow Alice and Bob to perform some off-line work and prepare some polynomial-size inputs (in ROM). First, we devise a method for Alice and Bob to generate a combined split set S (discussed later) by having each of Alice and Bob contribute sampled letters to S. Second, we securely estimate the ℓ_2 distance of Alice and Bob's original, un-splitted samples using techniques from [43]. Finally, we adjust our approximation by accounting for a few letters which differ from the original alphabet or which cannot be estimated efficiently. The main focus of our analysis goes into proving our construction adds only poly-logarithmic communication over the insecure protocol.

We describe the details of our secure protocols, as well as our results on independence testing, in the full version of this paper.

2 Preliminaries

Notation. Throughout this paper we denote distributions in small letters, and distribution samples in capital letters. Unless stated otherwise, any distribution is on alphabet $[n]$, and domain elements of $[n]$ are addressed as letters.

We also denote any multiplicative error arising from approximation as $1+\alpha$, and any error or distance of/between distributions as ϵ. Unless stated otherwise, distance and norms are referring to the Euclidean distance and ℓ_2 norms.

We denote Poisson Random variables with parameter $\lambda > 0$ as $\mathrm{Poi}(\lambda)$.

Split Distributions. We use the concept of split distributions from [31] to essentially reduce testing under ℓ_1 distance to testing under ℓ_2 distance.

▶ **Definition 2.** *Given a probability distribution p on $[n]$ and a multiset S of items from $[n]$, define the* split distribution p_S *on $[n+|S|]$ as follows. For $i \in [n]$, let a_i be equal to 1 plus the number of occurrences of i in S; note that $\sum_{i=1}^{n} a_i = n + |S|$. We associate the elements of $[n+|S|]$ to elements of the set $E = \{(i,j) : i \in [n], 1 \le j \le a_i\}$. Now the distribution p_S has support E and a random draw (i,j) from p_S is sampled by picking i randomly from p and j uniformly at random from $[a_i]$.*

Recall from [31] that split distributions are used to upper bound the ℓ_2 norm of an underlying distribution while maintaining its ℓ_1 distance to other distributions:

▶ **Fact 3** ([31]). *Let p and q be probability distributions on $[n]$, and S a given multiset of $[n]$. Then, (1) We can simulate a sample from p_S or q_S by taking a single sample from p or q, respectively; and (2) $\|p_S - q_S\|_1 = \|p - q\|_1$.*

▶ **Lemma 4** ([31]). *Let p be a distribution on $[n]$. Then: (i) For any multisets $S \subseteq S'$ of $[n]$, $\|p_{S'}\|_2 \le \|p_S\|_2$, and (ii) If S is obtained by taking $\mathrm{Poi}(m)$ samples from p, then $\mathbb{E}[\|p_S\|_2^2] \le 1/m$.*

3 Closeness Testing: Communication-Efficient Protocol

In this section we consider the closeness testing problem 2PCT, focusing on the 2-party communication complexity only. In the full version of this paper, we show how to modify the protocol to make it secure.

As mentioned in the introduction, one way to obtain a protocol is to use unequal-size closeness testing, where Alice has s samples and Bob has t samples: Alice just sends her s samples to Bob, and Bob invokes a standard algorithm for closeness testing. Using the optimal bounds from, say, [31], we get the following trade-off for fixed ϵ: $s = \tilde{O}(n/\sqrt{t})$, with the condition that $s, t \ge \sqrt{n}$. Here we obtain a *polynomially smaller* communication complexity, $s = \tilde{O}_\epsilon(n^2/t^2)$, whenever t is above the information-theoretic minimum on the number of samples. In Section 4, we show a nearly-matching lower bound.

3.1 Tool: approximation via occurrence vectors

Our protocol uses the framework introduced in [31], allowing us to focus on the ℓ_2 testing problem. For ℓ_2 testing, we show that we can approximate the ℓ_2 distance of two discrete distributions p, q by approximating the ℓ_2 distance of their respective sample occurrence vectors, defined as follows.

▶ **Definition 5.** *Given t samples of distribution p over $[n]$, we define the* occurrence vector *$X \in [t]^n$ such that X_i represent the count of occurrences of element $i \in [n]$ in the sample set.*

The following lemma bounds how well we need to estimate the ℓ_2 distance between occurrence vectors to distinguish between $p = q$ vs. $\|p - q\|_1 \geq \epsilon$. It shows that the more samples we have, the less accurate the ℓ_2 estimation needs to be. Using the framework from [31], for now it is enough to assume that the ℓ_2 norm of both p and q is bounded by $U < 1$.

▶ **Lemma 6.** *Let p, q be distributions over $[n]$ with $\|p\|_2, \|q\|_2 \leq U$ for some $U < 1$. There exists $t = O(U \cdot n \cdot \epsilon^{-2})$, and $\alpha = \Omega(U)$, such that given Δ which is $(1 \pm \alpha)$-factor approximation of $\|X - Y\|_2^2$ where X, Y represent the occurrence vectors of t samples drawn from p, q respectively, then, using Δ, it is possible to distinguish whether $p = q$ versus $\|p - q\|_1 > \epsilon$ with 0.8 probability.*

The actual distinguishing algorithm is simple: for fixed $\alpha = \Omega(U)$, we merely compare Δ to some fixed threshold τ (fixed in the proof below). The intuition is that for a given number of samples, we have some gap between the range of possible distances $\|X - Y\|_2^2$ for each of the cases. If the number of samples is close to the information-theoretic minimum [24], then the gap is minimal and we need to calculate almost exactly the distance, hence estimating the distance between occurrence vectors doesn't help. However, as the number of samples t increases, so does the gap between the ranges, allowing for a looser distance approximation.

Proof of Lemma 6. Given $t = O(U/\epsilon'^2)$ samples from each p, q, according to [24, Proposition 3.1], the estimator $Z = \sqrt{\sum_i (X_i - Y_i)^2 - X_i - Y_i}/t$ is a $\max\{\epsilon', \|p - q\|_2 /8\}$ additive approximation of $\|p - q\|_2$ with 0.9 probability. Setting $\epsilon' = \epsilon/(8\sqrt{n})$, we obtain: (1) $\|p - q\|_2 = 0 \Rightarrow \|X - Y\|_2^2 \leq \frac{\epsilon^2 t^2}{4n} + 2t$; and (2) $\|p - q\|_2 > \epsilon/\sqrt{n} \Rightarrow \|X - Y\|_2^2 \geq \frac{3\epsilon^2 t^2}{4n} + 2t$ Now, suppose Δ is such that $\frac{\Delta}{\|X-Y\|_2^2} \in (1 - \alpha, 1 + \alpha)$. If $\|p - q\|_1 = 0$, then $\|p - q\|_2 = 0$ and hence $\Delta \leq (1+\alpha)(\frac{\epsilon^2 t^2}{4n} + 2t) \leq t(\frac{\epsilon^2 t}{4n} + 2 + 2\alpha + \frac{\alpha\epsilon^2 t}{4n})$. On the other hand, if $\|p - q\|_1 > \epsilon$, then $\|p - q\|_2 > \epsilon/\sqrt{n}$ and hence $\Delta \geq (1 - \alpha)(\frac{3\epsilon^2 t^2}{4n} + 2t) \geq t(\frac{3\epsilon^2 t}{4n} + 2 - 2\alpha - \frac{3\alpha\epsilon^2 t}{4n})$.

We distinguish the two cases, by comparing Δ to $\tau = \frac{\epsilon^2 t^2}{2n} + 2t$: namely $p = q$ iff $\Delta \leq \tau$. Indeed we argue that $t(\frac{3\epsilon^2 t}{4n} + 2 - 2\alpha - \frac{3\alpha\epsilon^2 t}{4n}) - \tau \geq \tau - t(\frac{\epsilon^2 t}{4n} + 2 + 2\alpha + \frac{\alpha\epsilon^2 t}{4n})$. We have that $\frac{\epsilon^2 t}{4n} - 2\alpha - \frac{3\alpha\epsilon^2 t}{4n} \geq 0$, or $\alpha \leq \frac{\epsilon^2 t}{4n \cdot (2 + 3\epsilon^2 t/4n)}$. Since $t = O(Un\epsilon^{-2})$, the conclusion follows. ◀

3.2 Communication vs number of samples

We now provide a (non-secure) protocol for 2PCT with a trade-off between communication and number of samples.

▶ **Theorem 7** (Closeness, insecure)**.** *Fix $n > 1$ and $\epsilon \leq 2$. There exists some constant $C > 0$ such that for all $t \geq C \cdot \max\left(n^{2/3} \cdot \epsilon^{-4/3}, \sqrt{n} \cdot \epsilon^{-2}\right)$, the problem $2\text{PCT}_{n,t,\epsilon}$ can be solved using $\tilde{O}\left(\frac{n^2}{t^2\epsilon^4} + 1\right)$ bits of communication.*

The protocol uses Lemma 6 as the main algorithmic tool and proceeds as follows. Bob generates multi-set S using samples from b and sends S to Alice. Then, Alice and Bob each simulate samples from a_S and b_S respectively, and together approximate the ℓ_2 difference of the resulting occurrence vectors using sketching methods [9].

Proof of Theorem 7. We note that, according to Lemma 4, $\mathbb{E}[\|b_S\|_2^2] \leq t^2\epsilon^4/n^2$ and hence $\|b_S\|_2^2 = O(t^2\epsilon^4/n^2)$ with at least 90% probability. Furthermore, since $t = \Omega(\sqrt{n}/\epsilon^2)$, we have that $|S| = O(n)$ with high probability. From now on, we condition on these two events.

Non-Secure 2pCT(a, b, t)
Alice's input: t samples from a
Bob's input: t samples from b
1. Fix $\alpha = \Omega(t \cdot \epsilon^2/n)$.
2. Bob generates multi-set S using $\mathrm{Poi}(\frac{n^2}{t^2\epsilon^4})$ samples from b.
3. Bob sends S to Alice.
4. Alice and Bob recast their samples as being from distributions a_S, b_S (see Def. 2), and set A_S, B_S to be the respective occurrence vectors.
5. Alice and Bob each estimate $\|a_S\|_2$ and $\|b_S\|_2$ up to factor 2; if the two estimates are not within factor 4, output "ϵ-FAR";
6. Alice and Bob approximate $\Delta = \|A_S - B_S\|_2^2$ up to $(1+\alpha)$ factor, using, say, [9].
7. If Δ is less than $\tau = \frac{\epsilon^2 t^2}{2n} + 2t$ output "SAME", and, otherwise, output "ϵ-FAR".

If $\|a_S\|_2 \neq \Theta(\|b_S\|_2)$ then distributions are different and we output "ϵ-far" is step 5. Otherwise, we have that $\|a_S\|_2^2 = O(\|b_S\|_2^2) = O(t^2\epsilon^4/n^2)$. Hence we can use Lemma 6, where $U = O(t\epsilon^2/n)$ and $\alpha = \Omega(U)$, to claim the correctness of the protocol.

In terms of communication complexity, first, communicating S takes $|S| \log n = \tilde{O}(n^2/t^2\epsilon^4)$ bits with high probability. Second, estimating Δ up to approximation $1+\alpha$ takes $\tilde{O}(1/\alpha^2) = \tilde{O}(n^2/t^2\epsilon^4)$ bits, using standard ℓ_2 estimation algorithms [9, 47]. ◀

▶ Remark 8. Another application of this protocol is that it can be simulated by a *single party* to obtain a space bounded *streaming algorithm* with the same space/sample trade-offs. While we are not formalizing this argument in this paper, this can essentially be done by storing S and sketching $\|A_S - B_S\|_2^2$.

4 Closeness Testing: Communication Lower Bounds

We now prove that the protocol for 2pCT from Section 3 is near-tight, showing the following:

▶ **Theorem 9.** *Let a, b be some distributions over alphabet $[n]$, where Alice and Bob each receive $\mathrm{Poi}(t)$ samples from a, b respectively, for $t \leq n/\log^c n$ for some large enough $c > 1$. Then any (two-way) communication protocol Π that distinguishes between $a = b$ and $\|a - b\|_1 \geq 1/2$ requires $s = \tilde{\Omega}(n^2/t^2)$ communication.*

Intuitively, our proof formalizes the concept that in testing distributions for closeness, "collisions is all that matters", *even in the communication model*. This is similar to the intuition from the "canonical tester" from [59], which shows a similar principle when all the samples are accessible. Our result can be seen to extending it to saying that the canonical tester is still the best even if we have more-than-strictly-necessary number of samples that we could potentially compress in a communication protocol.

To prove the theorem, we rely on the following communication complexity lower bound, which is a variant of the Gap-Hamming-Distance (GHD) lower bound [23, 57]. Somewhat surprisingly, there does not seem to be a proof in the *one-way communication* model, which would be simpler than the two-way proof from the lemma below.

▶ **Lemma 10.** *Let $n \geq 1$ be even. There exists some $\beta = \beta(n) \in [\Theta(\sqrt{n}), \Theta(\sqrt{n \log n})]$, satisfying the following. Consider a two-way communication protocol $\mathcal{A}$ that, with probability at least 0.9, for $x, y \in \{0,1\}^n$ with $\|x\|_1 = \|y\|_1 = n/2$, can distinguish between the case when $\|x - y\|_1 = n/2$ versus $\|x - y\|_1 - n/2 \in [\beta, 2\beta]$. Then $\mathcal{A}$ must exchange at least $\Omega(\frac{n}{\log n \cdot \log\log n \cdot \log\log\log n})$ bits of communication.*

The proof of this lemma is presented in the full version of this paper.

Proof of Theorem 9. The idea is to reduce an instance of the GHD problem from Lemma 10 to an instance of closeness testing by carefully molding the input (x, y) into a couple of related occurrence vectors $(A, B) \in \mathbf{N}^n \times \mathbf{N}^n$ (recall that an occurrence vector precisely describes a set of samples).

Fix input vectors x, y, of length $m = \frac{n^2}{t^2 \log^3 n}$, to the above GHD problem. Let $\Delta = \beta(m) = \Omega(\sqrt{m})$, and $\delta = \frac{1}{2}(\|x - y\|_1 - m/2) \in \{0\} \cup [\Delta/2, \Delta]$. The case of $\delta = 0$ will correspond to "same" case (i.e. $a = b$), and $\delta \in [\Delta/2, \Delta]$ – to "far" case (i.e. $\|a - b\|_1 \in [1/2, 1]$).

Fix $d = n/10$ and $l = C \cdot t \cdot \log n$ (where C is some constant that we shall fix later), which have the following meaning: each distribution a, b has half mass over $[d]$ items uniformly (called *dense items*), and the other half on $[l]$ items uniformly (called *large items*). When $a = b$, these are the same items, and when $a \neq b$, the large items are the same while the dense items have supports with a large difference. In particular, the dense items are supported on sets S_A, S_B respectively, with $|S_A| = |S_B| = d$, and $S_A \cap S_B = d \cdot \frac{\Delta - \delta}{\Delta}$; we hence also have that $|S_A \setminus S_B| = d \cdot \frac{\delta}{\Delta}$.

Now for $i \geq 0$, let $D(i) = \Pr[\mathrm{Poi}(t/2d) = i]$, i.e., probability a dense number is sampled i times (when sampling $\mathrm{Poi}(t)$ items from one of the distributions). For simplicity, we write $D(i, j) = D(i) \cdot D(j)$. Similarly we define $L(i) = \Pr[\mathrm{Poi}(t/2l) = i]$ and $L(i, j) = L(i) \cdot L(j)$. We also set $k = \Theta(\log n)$, which should be thought of as an upper bound on the count of any fixed item (whp). The algorithm constructs the occurrence vectors A, B iteratively coordinate by coordinate. Let $m_c = m/4 - \Delta$.

2pCT Lower Bound Reduction
Input: (x, y) size m input bits for the Exact GHD problem
Output: (A, B) occurrence vectors for $\mathrm{Poi}(t)$ samples for the 2pCT Problem.

1. For each $i, j \in \{1, \dots k\}$, and for each $c \in [m]$ (corresponding to a coordinate of x, y), we take $z_c = \mathrm{Poi}(\frac{d}{\Delta} \cdot D(i, j))$, and generate z_c pairs $(i \cdot x_c, j \cdot y_c)$ (i.e., we set the corresponding coordinate of A or B to i or j iff $x_c = 1$ or $y_c = 1$ respectively);
2. For each $i \in \{1, \dots k\}$, generate $\mathrm{Poi}(d \cdot D(i, 0))$ pairs $(i, 0)$, and similarly-distributed number of pairs $(0, i)$;
3. For each $i, j \in \{1, \dots k\}$, generate $\mathrm{Poi}(l \cdot L(i, j) - m_c \cdot \frac{d}{\Delta} D(i, j))$ pairs (i, j);
4. For each $i \in \{1, \dots k\}$, generate $\mathrm{Poi}(l \cdot L(i, 0) - \frac{m}{4} \cdot \frac{d}{\Delta} \sum_{j=1}^{k} D(i, j))$ pairs $(i, 0)$, and similarly-distributed number of pairs $(0, i)$.
5. Generate the required number of $(0, 0)$ pairs so that A, B have length precisely n;
6. Permute the coordinates of A, B using a common randomly picked permutation over $[n]$.

In each of steps (1)-(5) above, we say we "generate a pair (i, j)" which corresponds to setting the next coordinate of A and B to i and j respectively. We only use the input vectors (x, y) in step (1). We note that all random variables are chosen using shared randomness.

We first claim that the above reduction is well-defined, and in particular all arguments of the Poisson variables are positive.

▷ Claim 11. All the Poisson random variables from above have positive argument.

Proof. We only need to prove this for steps 3 and 4 as the other ones are obvious. Indeed, for $i, j \geq 1$, we get that $l \cdot L(i, j) = t \log n \cdot (\Omega(1/\log n))^{i+j} = t/\log n \cdot (\Omega(1/\log n))^{i+j-2}$, whereas, $m_c \cdot \frac{d}{\Delta} D(i, j) = O(\sqrt{m} \cdot n \cdot (t/2d)^{i+j}) \leq \frac{n^2}{t \log^{1.5} n} \cdot O(t^2/n^2) \cdot (O(t/n))^{i+j-2} \leq \frac{t}{\log^{1.5} n} (O(t/n))^{i+j-2}$. Thus $l \cdot L(i, j) - m_c \cdot \frac{d}{\Delta} D(i, j) \geq 0$ for all $i, j \geq 1$.

Similarly, for step 5, for $i \geq 1$, we have, $l \cdot L(i,0) = \Omega(t \cdot (O(1/\log n))^{i-1})$, whereas, $m/4 \cdot \frac{d}{\Delta} \sum_{j \geq 1} D(i,j) \leq O(\sqrt{m} \cdot n \cdot \sum_{j \geq 1} (O(t/n))^{i+j}) \leq O(\frac{n^2}{t \log^{1.5} n} \cdot (O(t/n))^{i+1}) \leq O(\frac{t}{\log^{1.5} n} \cdot (O(t/n))^{i-1})$. We again have $l \cdot L(i,0) - m/4 \cdot \frac{d}{\Delta} \sum_{j \geq 1} D(i,j) \geq 0$ as required. ◁

We now prove the core of the reduction: that the distribution of (A, B) (denoted $\hat{\mathcal{D}}$) is close to the distribution $\mathcal{D}$ of occurrence vectors of $\mathrm{Poi}(t)$ i.i.d. samples from (a, b), such that $a = b$ if $\|x - y\|_1 = m/2$, and similarly, $\|a - b\|_1 \geq 1/2$ when $\|x - y\|_1 \geq m/2 + \beta$. We will prove that, for distribution of (co-)occurrences of large items is nearly same in the two instances; and similarly for the dense items. We partition the coordinates of (x, y) in the following four groups, each corresponding to either occurrences of dense or large items:

- large: $m_c = m/4 - \Delta$ coordinates for each of $(1,1)$ and $(0,0)$ coordinate pairs (i.e., coordinates $i \in [m]$ where $(x_i, y_i) = (1,1)$ or $(x_i, y_i) = (0,0)$);
- large: $m/4$ coordinates for each of $(1,0)$ and $(0,1)$ pairs;
- dense: $\Delta - \delta$ coordinates for each of $(1,1)$ and $(0,0)$ pairs;
- dense: δ coordinates for each of $(1,0)$ and $(0,1)$ pairs.

Note that this accounts for all coordinates for a pair x, y such that $\|x - y\|_1 = m/2 + 2\delta$.

Next, we compare the distributions of occurrences for large and dense items in the generated vectors (A, B) as opposed to occurrences of items coming from distributions a, b defined above. In particular, we consider the distribution of counts $c_{i,j}$, where $i + j > 0$, where $c_{i,j}$ is the number of large items which where sampled i times on Alice's side and j times on the Bob's side; we will refer to them as (i, j) occurrence pairs.

We denote by $\hat{\mathcal{D}}^L, \hat{\mathcal{D}}^D$ the distribution of, respectively, large and dense $\{c_{i,j}\}_{i+j>0}$ occurrence pairs in (A, B). Similarly, we denote $\mathcal{D}^L, \mathcal{D}^D$ the distribution of, respectively, large and dense $\{c_{i,j}\}_{i+j>0}$ occurrence pairs randomly drawn from (a, b). Note that $\mathcal{D}^L, \mathcal{D}^D$ are multinomial distributions, formally defined as follows:

▶ **Definition 12.** *Fix $n, k \geq 1$, vector $\vec{p} \in \mathbb{R}_+^k$, where $\sum_{i=1}^k p_i \leq 1$. The k-dimensional random variable $(M_1, \ldots M_k) = \mathrm{Mult}_{-0}(n; \vec{p})$ is obtained by drawing a Multinomial r.v. with parameters n and probability vector $(1 - \sum_{i=1}^k p_i, \vec{p})$, and dropping the first coordinate.*

In particular, $\mathcal{D}^L = \mathrm{Mult}_{-0}(l; \vec{p}_L)$ where $\vec{p}_L = (L(i,j))_{i,j \geq 0; i+j>0}$; $\mathcal{D}^D$ will be clarified later. We now deduce $\hat{\mathcal{D}}^L$ and $\hat{\mathcal{D}}^D$. Below we use the fact that the sum of Poisson random variables is also Poisson.

▷ Claim 13. $\hat{\mathcal{D}}^L$ is distributed as $\mathrm{Poi}(l \cdot \vec{p}_L)$. Also $\hat{\mathcal{D}}^D$ is distributed as $\mathrm{Poi}(\frac{\Delta - \delta}{\Delta} \cdot d \cdot \vec{p}_D) + \mathrm{Poi}(\frac{\delta}{\Delta} \cdot d \cdot \vec{p}_{D^0})$ where $\vec{p}_D = (D(i,j))_{i,j \geq 0; i+j>0}$ and $\vec{p}_{D^0} = (D(i) \cdot \mathbb{1}[j=0] + D(j) \cdot \mathbb{1}[i=0])_{i,j \geq 0; i+j>0}$.

Proof. For $i, j \in \{1, \ldots k\}$, $\hat{\mathcal{D}}^L_{i,j}$ is distributed as $\mathrm{Poi}(l \cdot L(i,j))$, which is composed of $\mathrm{Poi}(m_c \cdot \frac{d}{\Delta} D(i,j))$ (from the first step: there are m_c coordinate pairs $(1,1)$), plus $\mathrm{Poi}(l \cdot L(i,j) - m_c \cdot \frac{d}{\Delta} \cdot D(i,j))$ (from the third step).

Similarly, $\hat{\mathcal{D}}^L_{i,0}$ (and by symmetric argument also $\hat{\mathcal{D}}^L_{0,i}$) is distributed as $\mathrm{Poi}(l \cdot L(i,0))$, composed of $\mathrm{Poi}(m/4 \cdot \frac{d}{\Delta} \sum_{j=1}^k D(i,j))$ (from the first step: there are $m/4$ coordinate pairs $(1,0)$), plus $\mathrm{Poi}(l \cdot L(i,0) - m/4 \cdot \frac{d}{\Delta} \sum_{j=1}^k D(i,j))$ (from step 4).

For $i, j \geq 1$, $\hat{\mathcal{D}}^D_{i,j}$ are distributed as $\mathrm{Poi}((\Delta - \delta) \cdot \frac{d}{\Delta} D(i,j))$ since there are $\Delta - \delta$ coordinate pairs $(1,1)$. For $\hat{\mathcal{D}}^D_{i,0}$ (and similarly $\hat{\mathcal{D}}^L_{0,i}$), the distribution is $\mathrm{Poi}(\delta \cdot \frac{d}{\Delta} \sum_{j=1}^k D(i,j))$ (from the first step: there are δ coordinate pairs $(1,0)$), plus $\mathrm{Poi}(d \cdot D(i,0))$ (from the second step). This amounts to $\mathrm{Poi}(d \cdot \frac{\delta}{\Delta} \sum_{j=1}^k D(i,j) + d \cdot D(i,0)) = \mathrm{Poi}(d \frac{\Delta - \delta}{\Delta} \cdot D(i,0) + d \frac{\delta}{\Delta} \cdot (D(i,0) + \sum_{j=1}^k D(i,j))) = \mathrm{Poi}(d \cdot \frac{\Delta - \delta}{\Delta} \cdot D(i,0) + d \cdot \frac{\delta}{\Delta} \cdot D(i))$. ◁

We prove $\left\|\hat{\mathcal{D}} - \mathcal{D}\right\|_{TV} \leq 0.01 + o(1)$ by showing (i) $\left\|\hat{\mathcal{D}}^L - \mathcal{D}^L\right\|_{TV} \leq 0.01 + o(1)$ and (ii) $\left\|\hat{\mathcal{D}}^D - \mathcal{D}^D\right\|_{TV} = o(1)$. We compare $\hat{\mathcal{D}}^L$ and $\hat{\mathcal{D}}^D$ versus $\mathcal{D}^L$ and $\mathcal{D}^D$ using the following estimate on the TV distance between Multinomial and Poisson random variables. Note that the identity of items is not important, as the items are randomly permuted inside the domain, for both A, B as well as in distributions a, b.

▶ **Theorem 14** ([11]). *Let $n, k \geq 1$, as well as a vector $\vec{p} \in \mathbb{R}_+^k$, where $p = \sum_{i=1}^k p_i \leq 1$. Consider the random variable $(M_1, \ldots M_k)$ drawn from the Multinomial $\mathrm{Mult}_{-0}(n; \vec{p})$. Also consider the Poisson random variable $P = (P_1, \ldots, P_k)$ where $P_i \sim \mathrm{Poi}(np_i)$. Then the variables $M = (M_1, \ldots M_k)$ and $(P_1, \ldots P_k)$ are at a statistical distance of $O(p \log n)$.*

By Theorem 14, the TV-distance between $\mathcal{D}^L = \mathrm{Mult}_{-0}(l; \vec{p}_L)$ and $\hat{\mathcal{D}}^L = \mathrm{Poi}(l \cdot \vec{p}_L)$ is bounded by: $O(\log n) \cdot \sum_{i,j \geq 0; i+j>0} L(i,j) \leq O(\log n) \cdot \sum_{i,j \geq 0; i+j>0} (t/2l)^{i+j} \leq O(1)/C \leq 0.01$ (for sufficiently large constant C).

For (ii), we note $\mathcal{D}^D$ can be thought of as two distributions, corresponding to: (1) items in $S_A \cap S_B$, (2) items in $S_A \triangle S_B$. The occurrence counts for (1) are distributed as a Multinomial M^D with parameters $|S_A \cap S_B| = d \cdot \frac{\Delta - \delta}{\Delta}$ and probability vector $\vec{p}_D = (D(i,j))_{i,j \geq 0; i+j>0}$. By Theorem 14, the TV-distance between M^D and the distribution $\mathrm{Poi}(d \cdot \frac{\Delta - \delta}{\Delta} \cdot \vec{p}_D)$ is bounded by: $O(\log n) \cdot \sum_{i,j \geq 0; i+j>0} D(i,j) \leq O(\log n) \cdot \sum_{i,j \geq 0; i+j>0} (t/2d)^{i+j} \leq O(1/\log n)$.

For (2), we can only have $(i, 0)$ and $(0, j)$ pairs, and the occurrence counts are distributed as a Multinomial $M^{D^0} = \mathrm{Mult}_{-0}(|S_A \triangle S_B|/2; \vec{p}_{D^0})$. By Theorem 14, the TV-distance between M^{D^0} and $\mathrm{Poi}(\frac{\delta}{\Delta} \cdot d \cdot \vec{p}_{D^0})$ is at most $O(\log n) \cdot \sum_{i \geq 1} D(i) \leq O(1/\log n)$.

Thus we conclude that the distributions $\hat{\mathcal{D}}$ and $\mathcal{D}$ are at a small TV distance. ◀

References

1. Jayadev Acharya, Clément L. Canonne, Cody Freitag, and Himanshu Tyagi. Test without Trust: Optimal Locally Private Distribution Testing. *CoRR*, abs/1808.02174, 2018.
2. Jayadev Acharya, Clément L. Canonne, and Himanshu Tyagi. Distributed Simulation and Distributed Inference. *CoRR*, abs/1804.06952, 2018. `arXiv:1804.06952`.
3. Jayadev Acharya, Hirakendu Das, Ashkan Jafarpour, Alon Orlitsky, and Shengjun Pan. Competitive closeness testing. In *Proceedings of the 24th Annual Conference on Learning Theory*, pages 47–68, 2011.
4. Jayadev Acharya, Hirakendu Das, Ashkan Jafarpour, Alon Orlitsky, Shengjun Pan, and Ananda Suresh. Competitive classification and closeness testing. In *Conference on Learning Theory*, pages 22–1, 2012.
5. Jayadev Acharya, Ashkan Jafarpour, Alon Orlitsky, and Ananda Theertha Suresh. Sublinear algorithms for outlier detection and generalized closeness testing. In *Information Theory (ISIT), 2014 IEEE International Symposium on*, pages 3200–3204. IEEE, 2014.
6. Jayadev Acharya, Ziteng Sun, and Huanyu Zhang. Differentially Private Testing of Identity and Closeness of Discrete Distributions. *CoRR*, abs/1707.05128, 2017. `arXiv:1707.05128`.
7. Rudolf Ahlswede and Imre Csiszár. Hypothesis testing with communication constraints. *IEEE transactions on information theory*, 32(4), 1986.
8. Maryam Aliakbarpour, Ilias Diakonikolas, and Ronitt Rubinfeld. Differentially Private Identity and Closeness Testing of Discrete Distributions. *CoRR*, abs/1707.05497, 2017. `arXiv:1707.05497`.
9. Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comp. Sys. Sci.*, 58:137–147, 1999. Previously appeared in STOC'96.
10. S Amari et al. Statistical inference under multiterminal data compression. *IEEE Transactions on Information Theory*, 44(6):2300–2324, 1998.

11 Andrew D Barbour. Stein's method and Poisson process convergence. *Journal of Applied Probability*, 25(A):175–184, 1988.

12 Tuğkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D Smith, and Patrick White. Testing closeness of discrete distributions. *Journal of the ACM (JACM)*, 60(1):4, 2013.

13 Tuğkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D. Smith, and Patrick White. Testing Closeness of Discrete Distributions. *J. ACM*, 60(1):4:1–4:25, February 2013. doi:10.1145/2432622.2432626.

14 Amos Beimel, Paz Carmi, Kobbi Nissim, and Enav Weinreb. Private Approximation of Search Problems. *SIAM J. Comput.*, 38(5):1728–1760, 2008.

15 Amos Beimel, Renen Hallak, and Kobbi Nissim. Private Approximation of Clustering and Vertex Cover. *Computational Complexity*, 18(3):435–494, 2009.

16 Bhaswar Bhattacharya and Gregory Valiant. Testing closeness with unequal sized samples. In *Advances in Neural Information Processing Systems*, pages 2611–2619, 2015.

17 Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the Circuit Size Barrier for Secure Computation Under DDH. In *CRYPTO (1)*, volume 9814 of *Lecture Notes in Computer Science*, pages 509–539. Springer, 2016.

18 Elette Boyle, Niv Gilboa, and Yuval Ishai. Group-Based Secure Computation: Optimizing Rounds, Communication, and Computation. In *EUROCRYPT (2)*, volume 10211 of *Lecture Notes in Computer Science*, pages 163–193, 2017.

19 Mark Braverman, Ankit Garg, Tengyu Ma, Huy L Nguyen, and David P Woodruff. Communication lower bounds for statistical estimation problems via a distributed data processing inequality. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 1011–1020. ACM, 2016.

20 Sergey Bravyi, Aram W Harrow, and Avinatan Hassidim. Quantum algorithms for testing properties of distributions. *IEEE Transactions on Information Theory*, 57(6):3971–3981, 2011.

21 Bryan Cai, Constantinos Daskalakis, and Gautam Kamath. Priv'IT: Private and Sample Efficient Identity Testing. *arXiv preprint*, 2017. arXiv:1703.10127.

22 Clément L Canonne. A survey on distribution testing: Your data is big. but is it blue? In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 22(63), pages 1–9, 2015.

23 Amit Chakrabarti and Oded Regev. An optimal lower bound on the communication complexity of gap-hamming-distance. *SIAM Journal on Computing*, 41(5):1299–1317, 2012.

24 Siu-On Chan, Ilias Diakonikolas, Gregory Valiant, and Paul Valiant. Optimal algorithms for testing closeness of discrete distributions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1193–1203. Society for Industrial and Applied Mathematics, 2014.

25 Thomas M Cover. Hypothesis testing with finite statistics. *The Annals of Mathematical Statistics*, 40(3):828–835, 1969.

26 Michael Crouch, Andrew McGregor, Gregory Valiant, and David P Woodruff. Stochastic streams: Sample complexity vs. space complexity. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 57. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

27 Yuval Dagan and Ohad Shamir. Detecting Correlations with Little Memory and Communication. *CoRR*, abs/1803.01420, 2018. arXiv:1803.01420.

28 I. Diakonikolas, E. Grigorescu, J. Li, A. Natarajan, K. Onak, and L. Schmidt. Communication-Efficient Distributed Learning of Discrete Distributions. In *Advances in Neural Information Processing Systems*, 2017. To appear.

29 Ilias Diakonikolas, Themis Gouleakis, John Peebles, and Eric Price. Collision-based testers are optimal for uniformity and closeness. *arXiv preprint*, 2016. arXiv:1611.03579.

30 Ilias Diakonikolas, Moritz Hardt, and Ludwig Schmidt. Differentially private learning of structured discrete distributions. In *Advances in Neural Information Processing Systems*, pages 2566–2574, 2015.

31 Ilias Diakonikolas and Daniel M Kane. A new approach for testing properties of discrete distributions. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 685–694. IEEE, 2016.

32 Joan Feigenbaum, Yuval Ishai, Tal Malkin, Kobbi Nissim, Martin J. Strauss, and Rebecca N. Wright. Secure Multiparty Computation of Approximations. *ACM Trans. Algorithms*, 2(3):435–472, July 2006. doi:10.1145/1159892.1159900.

33 Sumegha Garg, Ran Raz, and Avishay Tal. Extractor-based time-space lower bounds for learning. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 990–1002. ACM, 2018.

34 Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178. ACM, 2009.

35 Oded Goldreich. Introduction to Property Testing (working draft), 2017. URL: www.wisdom.weizmann.ac.il/~oded/PDF/pt-v3.pdf.

36 Oded Goldreich and Dana Ron. On Testing Expansion in Bounded-Degree Graphs. *ECCC 7(20)*, 2000.

37 Michael Gutman. Asymptotically optimal classification for multiple tests with empirically observed statistics. *IEEE Transactions on Information Theory*, 35(2):401–408, 1989.

38 U. Hadar, J. Liu, Y. Polyanskiy, and O. Shayevitz. Communication Complexity of Estimating Correlations. In *Proceedings of the Symposium on Theory of Computing (STOC)*, 2019.

39 Shai Halevi, Robert Krauthgamer, Eyal Kushilevitz, and Kobbi Nissim. Private approximation of NP-hard functions. In *STOC*, pages 550–559. ACM, 2001.

40 Te Han. Hypothesis testing with multiterminal data compression. *IEEE transactions on information theory*, 33(6):759–772, 1987.

41 Martin E Hellman and Thomas M Cover. Learning with finite memory. *The Annals of Mathematical Statistics*, pages 765–782, 1970.

42 Piotr Indyk. Stable Distributions, Pseudorandom Generators, Embeddings and Data Stream Computation. *J. ACM*, 53(3):307–323, 2006. Previously appeared in FOCS'00.

43 Piotr Indyk and David Woodruff. Polylogarithmic Private Approximations and Efficient Matching. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, pages 245–264, Berlin, Heidelberg, 2006. Springer-Verlag. doi:10.1007/11681878_13.

44 Yuval Ishai, Tal Malkin, Martin J. Strauss, and Rebecca N. Wright. Private multiparty sampling and approximation of vector combinations. *Theor. Comput. Sci.*, 410(18):1730–1745, 2009.

45 Joe Kilian, André Madeira, Martin J. Strauss, and Xuan Zheng. Fast Private Norm Estimation and Heavy Hitters. In *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 176–193. Springer, 2008.

46 Gillat Kol, Ran Raz, and Avishay Tal. Time-space hardness of learning sparse parities. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1067–1080. ACM, 2017.

47 Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM J. Comput.*, 30(2):457–474, 2000. Preliminary version appeared in STOC'98.

48 Dana Moshkovitz and Michal Moshkovitz. Mixing implies lower bounds for space bounded learning. In *Conference on Learning Theory*, pages 1516–1566, 2017.

49 Moni Naor and Kobbi Nissim. Communication Complexity and Secure Function Evaluation. *CoRR*, cs.CR/0109011, 2001. arXiv:cs.CR/0109011.

50 Ran Raz. Fast learning requires good memory: A time-space lower bound for parity learning. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 266–275. IEEE, 2016.

51 Ran Raz. A time-space lower bound for a large class of learning problems. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 732–742. IEEE, 2017.

52 Ronitt Rubinfeld. Sublinear time algorithms. In *International Congress of Mathematicians*, volume 3, pages 1095–1110, 2006.

53 Ronitt Rubinfeld. Taming big probability distributions. *XRDS: Crossroads, The ACM Magazine for Students*, 19(1):24–28, 2012.

54 KR Sahasranand and Himanshu Tyagi. Extra Samples can Reduce the Communication for Independence Testing. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 2316–2320. IEEE, 2018.

55 Ofer Shayevitz. On Rényi measures and hypothesis testing. In *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, pages 894–898. IEEE, 2011.

56 Or Sheffet. Locally Private Hypothesis Testing. In *ICML*, 2018.

57 Alexander A. Sherstov. The Communication Complexity of Gap Hamming Distance. *Theory of Computing*, 8(1):197–208, 2012. doi:10.4086/toc.2012.v008a008.

58 Jayakrishnan Unnikrishnan. On optimal two sample homogeneity tests for finite alphabets. In *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, pages 2027–2031. Ieee, 2012.

59 Paul Valiant. Testing symmetric properties of distributions. *SIAM Journal on Computing*, 40(6):1927–1968, 2011. Previously in STOC'08.

60 Elad Verbin and Wei Yu. The streaming complexity of cycle counting, sorting by reversals, and other problems. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 11–25. SIAM, 2011.

61 Andrew Chi-Chih Yao. Protocols for Secure Computations (Extended Abstract). In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, FOCS '82, pages 160–164, 1982.

Two New Results About Quantum Exact Learning

Srinivasan Arunachalam
Center for Theoretical Physics, MIT, Cambridge, MA, USA
arunacha@mit.edu

Sourav Chakraborty
Indian Statistical Institute, Kolkata, India
sourav@isical.ac.in

Troy Lee
Centre for Quantum Software and Information, School of Software,
Faculty of Engineering and Information Technology, University of Technology Sydney, Australia
troyjlee@gmail.com

Manaswi Paraashar
Indian Statistical Institute, Kolkata, India
manaswi.isi@gmail.com

Ronald de Wolf
QuSoft, CWI and University of Amsterdam, The Netherlands
https://homepages.cwi.nl/~rdewolf/
rdewolf@cwi.nl

Abstract

We present two new results about exact learning by quantum computers. First, we show how to exactly learn a k-Fourier-sparse n-bit Boolean function from $O(k^{1.5}(\log k)^2)$ uniform quantum examples for that function. This improves over the bound of $\widetilde{\Theta}(kn)$ uniformly random *classical* examples (Haviv and Regev, CCC'15). Our main tool is an improvement of Chang's lemma for sparse Boolean functions. Second, we show that if a concept class $\mathcal{C}$ can be exactly learned using Q quantum membership queries, then it can also be learned using $O\left(\frac{Q^2}{\log Q}\log|\mathcal{C}|\right)$ *classical* membership queries. This improves the previous-best simulation result (Servedio-Gortler, SICOMP'04) by a $\log Q$-factor.

2012 ACM Subject Classification Hardware → Quantum computation; Theory of computation → Sample complexity and generalization bounds; Theory of computation → Boolean function learning

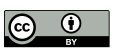
Keywords and phrases quantum computing, exact learning, analysis of Boolean functions, Fourier sparse Boolean functions

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.16

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at https://arxiv.org/abs/1810.00481.

Funding *Srinivasan Arunachalam*: Work done when at QuSoft, CWI, Amsterdam, the Netherlands. Supported by ERC Consolidator Grant 615307 QPROGRESS and MIT-IBM Watson AI Lab under the project *Machine Learning in Hilbert space*.
Sourav Chakraborty: Work done while on sabbatical at CWI, supported by ERC QPROGRESS.
Troy Lee: Part of this work was done while at the School for Physical and Mathematical Sciences, Nanyang Technological University and the Centre for Quantum Technologies, Singapore, supported by the Singapore National Research Foundation under NRF RF Award No. NRF-NRFF2013-13.
Ronald de Wolf: Supported by ERC QPROGRESS, and QuantERA project QuantAlgo 680-91-034, and NWO Gravitation project QSC.



1 Introduction

1.1 Quantum learning theory

Both quantum computing and machine learning are hot topics at the moment, and their intersection has been receiving growing attention in recent years as well. On the one hand there are particular approaches that use quantum algorithms like Grover search [18] and the Harrow-Hassidim-Lloyd linear-systems solver [19] to speed up learning algorithms for specific machine learning tasks (see [34, 29, 1, 9, 16] for recent surveys of this line of work). On the other hand there have been a number of more general results about the sample and/or time complexity of learning various concept classes using a quantum computer (see [4] for a survey). This paper presents two new results in the latter line of work. In both cases the goal is to *exactly* learn an unknown target function with high probability; for the first result our access to the target function is through quantum examples for the function, and for the second result our access is through membership queries to the function.

1.2 Exact learning of sparse functions from uniform quantum examples

Let us first explain the setting of distribution-dependent learning from examples. Let $\mathcal{C}$ be a class of functions, a.k.a. *concept class*. For concreteness assume they are ± 1-valued functions on a domain of size N; if $N = 2^n$, then the domain may be identified with $\{0,1\}^n$. Suppose $c \in \mathcal{C}$ is an unknown function (the *target* function or concept) that we want to learn. A learning algorithm is given *examples* of the form $(x, c(x))$, where x is distributed according to some probability distribution D on $[N]$. An (ε, δ)-learner for $\mathcal{C}$ w.r.t. D is an algorithm that, for every possible target concept $c \in \mathcal{C}$, produces a hypothesis $h : [N] \to \{-1, 1\}$ such that with probability at least $1 - \delta$ (over the randomness of the learner and the examples for the target concept c), h's generalization error is at most ε:

$$\Pr_{x \sim D}[c(x) \neq h(x)] \leq \varepsilon.$$

In other words, from D-distributed examples the learner has to construct a hypothesis that mostly agrees with the target concept *under the same* D.

In the early days of quantum computing, Bshouty and Jackson [11] generalized this learning setting by allowing coherent *quantum* examples. A quantum example for concept c w.r.t. distribution D, is the following $(\lceil \log N \rceil + 1)$-qubit state:

$$\sum_{x \in [N]} \sqrt{D(x)} |x, c(x)\rangle.$$

Clearly such a quantum example is at least as useful as a classical example, because measuring this state yields a pair $(x, c(x))$ where $x \sim D$. Bshouty and Jackson gave examples of concept classes that can be learned more efficiently from quantum examples than from classical random examples under specific D. In particular, they showed that the concept class of DNF-formulas can be learned in polynomial time from quantum examples under the *uniform* distribution, something we do not know how to do classically (the best classical upper bound is quasi-polynomial time [33]). The key to this improvement is the ability to obtain, from a uniform quantum example, a sample $S \sim \widehat{c}(S)^2$ distributed according to the squared *Fourier coefficients* of c.[1] This *Fourier sampling*, originally due to Bernstein and Vazirani [8], is very

[1] Parseval's identity implies $\sum_{S \in \{0,1\}^n} \widehat{f}(S)^2 = 1$, so this is indeed a probability distribution.

powerful. For example, if $\mathcal{C}$ is the class of $\mathbb{F}_2$-linear functions on $\{0,1\}^n$, then the unknown target concept c is a character function $\chi_S(x) = (-1)^{x \cdot S}$; its only non-zero Fourier coefficient is $\widehat{c}(S)$ hence one Fourier sample gives us the unknown S with certainty. In contrast, learning linear functions from classical uniform examples requires $\Theta(n)$ examples. Another example where Fourier sampling is proven powerful is in learning the class of ℓ-juntas on n bits.[2] Atıcı and Servedio [6] showed that $(\log n)$-juntas can be exactly learned under the uniform distribution in time polynomial in n. Classically it is a long-standing open question if a similar result holds when the learner is given uniform classical examples (the best known algorithm runs in quasi-polynomial time [24]). These cases (and others surveyed in [4]) show that uniform quantum examples (and in particular Fourier sampling) can be more useful than classical examples.[3]

In this paper we consider the concept class of n-bit Boolean functions that are *k-sparse in the Fourier domain*: $\widehat{c}(S) \neq 0$ for at most k different S's. This is a natural generalization of the above-mentioned case of learning linear functions, which corresponds to $k = 1$. It also generalizes the case of learning ℓ-juntas on n bits, which are functions of sparsity $k = 2^\ell$. Variants of the class of k-Fourier-sparse functions have been well-studied in the area of *sparse recovery*, where the goal is to recover a k-sparse vector $x \in \mathbb{R}^N$ given a low-dimensional linear sketch Ax for a so-called "measurement matrix" matrix $A \in \mathbb{R}^{m \times N}$. See [20, 23] for some upper bounds on the size of the measurement matrix that suffice for sparse recovery. Closer to the setting of this paper, there has also been extensive work on learning the concept class of n-bit *real-valued* functions that are k-sparse in the Fourier domain. In this direction Cheraghchi et al. [14] showed that $O(nk(\log k)^3)$ uniform examples suffice to learn this concept class, improving upon the works of Bourgain [10], Rudelson and Vershynin [27] and Candés and Tao [12].

In this paper we focus on *exactly* learning the target concept from uniform examples, with high success probability. So $D(x) = 1/2^n$ for all x, $\varepsilon = 0$, and $\delta = 1/3$. Haviv and Regev [21] showed that for classical learners $O(nk \log k)$ uniform examples suffice to learn k-Fourier-sparse functions, and $\Omega(nk)$ uniform examples are necessary. In Section 3 we study the number of uniform *quantum* examples needed to learn k-Fourier-sparse Boolean functions, and show that it is upper bounded by $O(k^{1.5}(\log k)^2)$. For $k \ll n^2$ this quantum bound is much better than the number of uniform examples used in the classical case. Proving the upper bound combines the fact that a uniform quantum example allows us to Fourier-sample the target concept, with some Fourier analysis of k-Fourier-sparse functions. In particular, we significantly strengthen "Chang's lemma" for the special case of k-Fourier-sparse Boolean functions. This lemma upper bounds the dimension of the span of the large-weight part of the Fourier support of a Boolean function, and our Theorem 13 improves this bound almost quadratically for the special case of k-Fourier-sparse functions. Our learner has two phases. In the first phase, using Chang's lemma, we show that the span of the Fourier support of the target function can be learned from $O(k(\log k)^2)$ Fourier samples. In the second phase, we reduce the number of variables to the dimension r of the Fourier support, and then invoke the classical learner of Haviv and Regev to learn the target function from $O(rk \log k)$ classical examples. Since it is known that $r = O(\sqrt{k} \log k)$ [28], the two phases together imply that $O(k^{1.5}(\log k)^2)$ uniform quantum examples suffice to exactly learn the target with high probability.

[2] We say $f : \{0,1\}^n \to \{-1,1\}$ is an *ℓ-junta* if there exists a set $S \subseteq [n]$ of size $|S| \leq \ell$ such that f depends only on the variables whose indices are in S.

[3] This is not the case in Valiant's *PAC-learning* model [32] of distribution-independent learning. There we require the same learner to be an (ε, δ)-learner for $\mathcal{C}$ w.r.t. *every* possible distribution D. One can show in this model (and also in the broader model of *agnostic* learning) that the quantum and classical sample complexities are equal up to a constant factor [5].

Since $r \geq \log k$, the second phase of our learner is always at least as expensive as the first phase. It might be possible to improve the upper bound to $O(k \cdot \text{polylog}(k))$ quantum examples, but that would require additional ideas to improve phase 2. We also prove a (non-matching) lower bound of $\Omega(k \log k)$ uniform quantum examples, using techniques from quantum information theory. We omitted some proofs due to space limitations; these may be found in [3].

1.3 Exact learning from quantum membership queries

Our second result is in a model of active learning. The learner still wants to exactly learn an unknown target concept $c : [N] \to \{-1, 1\}$ from a known concept class $\mathcal{C}$, but now the learner can choose which points of the truth-table of the target it sees, rather than those points being chosen randomly. More precisely, the learner can query $c(x)$ for any x of its choice. This is called a *membership query*.[4] Quantum algorithms have the following query operation available:

$$O_c : |x, b\rangle \mapsto |x, b \cdot c(x)\rangle,$$

where $b \in \{-1, 1\}$. For some concept classes, quantum membership queries can be much more useful than classical. Consider again the class $\mathcal{C}$ of $\mathbb{F}_2$-linear functions on $\{0, 1\}^n$. Using one query to a uniform superposition over all x and doing a Hadamard transform, we can Fourier-sample and hence learn the target concept exactly. In contrast, $\Theta(n)$ classical membership queries are necessary and sufficient for classical learners. As another example, consider the concept class $\mathcal{C} = \{\delta_i \mid i \in [N]\}$ of the N point functions, where $\delta_i(x) = 1$ iff $i = x$. Elements from this class can be learned using $O(\sqrt{N})$ quantum membership queries by Grover's algorithm, while every classical algorithm needs to make $\Omega(N)$ membership queries.

For a given concept class $\mathcal{C}$ of ± 1-valued function on $[N]$, let $D(\mathcal{C})$ denote the minimal number of classical membership queries needed for learners that can exactly identify every $c \in \mathcal{C}$ with success probability 1 (such learners are deterministic without loss of generality). Let $R(\mathcal{C})$ and $Q(\mathcal{C})$ denote the minimal number of classical and quantum membership queries, respectively, needed for learners that can exactly identify every $c \in \mathcal{C}$ with error probability $\leq 1/3$.[5] Servedio and Gortler [30] showed that these quantum and classical measures cannot be too far apart. First, using an information-theoretic argument they showed

$$Q(\mathcal{C}) \geq \Omega\left(\frac{\log |\mathcal{C}|}{\log N}\right).$$

Intuitively, this holds because a learner recovers roughly $\log |\mathcal{C}|$ bits of information, while every quantum membership query can give at most $O(\log N)$ bits of information. Note that this is tight for the class of linear functions, where the left- and right-hand sides are both constant. Second, using the so-called hybrid method they showed

$$Q(\mathcal{C}) \geq \Omega(1/\sqrt{\gamma(\mathcal{C})}),$$

[4] Think of the set $\{x \mid c(x) = 1\}$ corresponding to the target concept: a membership query asks whether x is a member of this set or not.

[5] We can identify each concept with a string $c \in \{-1, 1\}^N$, and hence $\mathcal{C} \subseteq \{-1, 1\}^N$. The goal is to learn the unknown $c \in \mathcal{C}$ with high probability using few queries to the corresponding N-bit string. This setting is also sometimes called "oracle identification" in the literature; see [4, Section 4.1] for more.

for some combinatorial parameter $\gamma(\mathcal{C})$ that we will not define here (but which is $1/N$ for the class $\mathcal{C}$ of point functions, hence this inequality is tight for that $\mathcal{C}$). They also noted the following upper bound:

$$D(\mathcal{C}) = O\left(\frac{\log|\mathcal{C}|}{\gamma(\mathcal{C})}\right).$$

Combining these three inequalities yields the following relation between $D(\mathcal{C})$ and $Q(\mathcal{C})$

$$D(\mathcal{C}) \leq O(Q(\mathcal{C})^2 \log|\mathcal{C}|) \leq O(Q(\mathcal{C})^3 \log N). \tag{1}$$

This shows that, up to a $\log N$-factor, quantum and classical membership query complexities of exact learning are polynomially close. While each of the three inequalities that together imply (1) can be individually tight (for different $\mathcal{C}$), this does not imply (1) itself is tight.

Note that Eq. (1) upper bounds the membership query complexity of *deterministic* classical learners. We are not aware of a stronger upper bound on *bounded-error* classical learners. However, in Section 4 we tighten that bound further by a $\log Q(\mathcal{C})$-factor:

$$R(\mathcal{C}) \leq O\left(\frac{Q(\mathcal{C})^2}{\log Q(\mathcal{C})}\log|\mathcal{C}|\right) \leq O\left(\frac{Q(\mathcal{C})^3}{\log Q(\mathcal{C})}\log N\right).$$

Note that this inequality is tight both for the class of linear functions and for the class of point functions.

Our proof combines the quantum adversary method [2, 7, 31] with an entropic argument to show that we can always find a query whose outcome (no matter whether it's 0 or 1) will shrink the concept class by a factor $\leq 1 - \frac{\log Q(\mathcal{C})}{Q(\mathcal{C})^2}$. While our improvement over the earlier bounds is not very large, we feel our usage of entropy to save a log-factor is new and may have applications elsewhere.

2 Preliminaries

Notation. Let $[n] = \{1, \ldots, n\}$. For an n-dimensional vector space, the standard basis vectors are $\{e_i \in \{0,1\}^n \mid i \in [n]\}$, where e_i is the vector with a 1 in the ith coordinate and 0s elsewhere. For $x \in \{0,1\}^n$, $i \in [n]$, let x^i be the input obtained by flipping the ith bit in x.

For $f : \{0,1\}^n \to \{-1,1\}$ and $B \in \mathbb{F}_2^{n\times n}$, define $f \circ B : \{0,1\}^n \to \{-1,1\}$ as $(f \circ B)(x) := f(Bx)$, where the matrix-vector product Bx is over $\mathbb{F}_2$. Throughout this paper, the rank of a matrix $B \in \mathbb{F}_2^{n\times n}$ will be taken over $\mathbb{F}_2$. Let $B_1, \ldots, B_n$ be the columns of B.

Fourier analysis on the Boolean cube. We introduce the basics of Fourier analysis here, referring to [26, 35] for more. Define the inner product between functions $f, g : \{0,1\}^n \to \mathbb{R}$ as

$$\langle f, g\rangle = \mathbb{E}_{x\in\{0,1\}^n}[f(x)\cdot g(x)],$$

where the expectation is uniform over all $x \in \{0,1\}^n$. For $S \in \{0,1\}^n$, the character function corresponding to S is given by $\chi_S(x) := (-1)^{S\cdot x}$, where the dot product $S \cdot x$ is $\sum_{i=1}^n S_i x_i$. Observe that the set of functions $\{\chi_S\}_{S\in\{0,1\}^n}$ forms an orthonormal basis for the space of real-valued functions over the Boolean cube. Hence every $f : \{0,1\}^n \to \mathbb{R}$ can be written uniquely as

$$f(x) = \sum_{S\in\{0,1\}^n} \widehat{f}(S)(-1)^{S\cdot x} \qquad \text{for all } x \in \{0,1\}^n,$$

where $\widehat{f}(S) = \langle f, \chi_S \rangle = \mathbb{E}_x[f(x)\chi_S(x)]$ is called a *Fourier coefficient* of f. For $i \in [n]$, we write $\widehat{f}(e_i)$ as $\widehat{f}(i)$ for notational convenience. Parseval's identity states that $\sum_{S\in\{0,1\}^n} \widehat{f}(S)^2 = \mathbb{E}_x[f(x)^2]$. If f has domain $\{-1,1\}$, then Parseval gives $\sum_{S\in\{0,1\}^n} \widehat{f}(S)^2 = 1$, so $\{\widehat{f}(S)^2\}_{S\in\{0,1\}^n}$ forms a probability distribution. The *Fourier weight* of function f on $\mathcal{S} \subseteq \{0,1\}^n$ is defined as $\sum_{S\in\mathcal{S}} \widehat{f}(S)^2$.

For $f : \{0,1\}^n \to \mathbb{R}$, the *Fourier support* of f is $\mathrm{supp}(\widehat{f}) = \{S : \widehat{f}(S) \neq 0\}$. The *Fourier sparsity* of f is $|\mathrm{supp}(\widehat{f})|$. The *Fourier span* of f, denoted $\mathrm{Fspan}(f)$, is the span of $\mathrm{supp}(\widehat{f})$. The *Fourier dimension* of f, denoted $\mathrm{Fdim}(f)$, is the dimension of the Fourier span. We say f is *k-Fourier-sparse* if $|\mathrm{supp}(\widehat{f})| \leq k$.

We now state a few structural results about Fourier coefficients and dimension.

▶ **Theorem 1** ([28]). *The Fourier dimension of a k-Fourier-sparse $f : \{0,1\}^n \to \{-1,1\}$ is $O(\sqrt{k}\log k)$.*

▶ **Lemma 2** ([17, Theorem 12]). *Let $k \geq 2$. The Fourier coefficients of a k-Fourier-sparse Boolean function $f : \{0,1\}^n \to \{-1,1\}$ are integer multiples of $2^{1-\lfloor \log k \rfloor}$.*

▶ **Definition 3.** *Let $f : \{0,1\}^n \to \{-1,1\}$ and suppose $B \in \mathbb{F}_2^{n\times n}$ is invertible. Define f_B as $f_B(x) = f((B^{-1})^{\mathsf{T}}x)$.*

▶ **Lemma 4.** *Let $f : \{0,1\}^n \to \mathbb{R}$ and suppose $B \in \mathbb{F}_2^{n\times n}$ is invertible. Then the Fourier coefficients of f_B are $\widehat{f_B}(Q) = \widehat{f}(BQ)$ for all $Q \in \{0,1\}^n$.*

Proof. Write out the Fourier expansion of f_B:

$$f_B(x) = f((B^{-1})^{\mathsf{T}}x) = \sum_{S\in\{0,1\}^n} \widehat{f}(S)(-1)^{S\cdot((B^{-1})^{\mathsf{T}}x)}$$
$$= \sum_{S\in\{0,1\}^n} \widehat{f}(S)(-1)^{(B^{-1}S)\cdot x} = \sum_{Q\in\{0,1\}^n} \widehat{f}(BQ)(-1)^{Q\cdot x},$$

where the third equality used $\langle S, (B^{-1})^{\mathsf{T}}x\rangle = \langle B^{-1}S, x\rangle$ and the last used the substitution $S = BQ$. ◀

An easy consequence is the next lemma:

▶ **Lemma 5.** *Let $f : \{0,1\}^n \to \{-1,1\}$, and $B \in \mathbb{F}_2^{n\times n}$ be an invertible matrix such that the first r columns of B are a basis of $\mathrm{Fspan}(f)$, and $\widehat{f}(B_1), \ldots, \widehat{f}(B_r)$ are non-zero. Then the Fourier span of f_B is spanned by $\{e_1, \ldots, e_r\}$, i.e., f_B has only r influential variables. Additionally, for every $i \in [r]$, $\widehat{f_B}(i) \neq 0$.*

Here is the well-known fact, already mentioned in the introduction, that one can Fourier-sample from uniform quantum examples:

▶ **Lemma 6.** *Let $f : \{0,1\}^n \to \{-1,1\}$. There exists a procedure that uses one uniform quantum example and satisfies the following: with probability $1/2$ it outputs an S drawn from the distribution $\{\widehat{f}(S)^2\}_{S\in\{0,1\}^n}$, otherwise it rejects.*

Information theory. We refer to [15] for a comprehensive introduction to classical information theory, and here just remind the reader of the basic definitions. A random variable $\mathbf{A}$ with probabilities $\Pr[\mathbf{A} = a] = p_a$ has *entropy* $H(\mathbf{A}) := -\sum_a p_a \log(p_a)$. For a pair of (possibly correlated) random variables $\mathbf{A}, \mathbf{B}$, the *conditional entropy* of $\mathbf{A}$ given $\mathbf{B}$, is $H(\mathbf{A} \mid \mathbf{B}) := H(\mathbf{A}, \mathbf{B}) - H(\mathbf{B})$. This equals $\mathbb{E}_{b\sim\mathbf{B}}[H(\mathbf{A} \mid \mathbf{B} = b)]$. The *mutual information*

between $\mathbf{A}$ and $\mathbf{B}$ is $I(\mathbf{A}:\mathbf{B}) := H(\mathbf{A}) + H(\mathbf{B}) - H(\mathbf{A},\mathbf{B}) = H(\mathbf{A}) - H(\mathbf{A} \mid \mathbf{B})$. The *binary entropy* $H(p)$ is the entropy of a bit with distribution $(p, 1-p)$. If ρ is a density matrix (i.e., a trace-1 positive semi-definite matrix), then its singular values form a probability distribution P, and the *von Neumann entropy* of ρ is $S(\rho) := H(P)$. We refer to [25, Part III] for a more extensive introduction to quantum information theory.

3 Exact learning of k-Fourier-sparse functions

In this section we consider exactly learning the concept class $\mathcal{C}$ of k-Fourier-sparse Boolean functions:

$$\mathcal{C} = \{f : \{0,1\}^n \to \{-1,1\} : |\mathrm{supp}(\widehat{f})| \leq k\}.$$

The goal is to exactly learn $c \in \mathcal{C}$ given *uniform examples* from c of the form $(x, c(x))$ where x is drawn from the uniform distribution on $\{0,1\}^n$. Haviv and Regev [21] considered learning this concept class and showed the following results.

▶ **Theorem 7** (Corollary 3.6 of [21]). *For every $n > 0$ and $k \leq 2^n$, the number of uniform examples that suffice to learn $\mathcal{C}$ with probability $1 - 2^{-\Omega(n \log k)}$ is $O(nk \log k)$.*

▶ **Theorem 8** (Theorem 3.7 of [21]). *For every $n > 0$ and $k \leq 2^n$, the number of uniform examples necessary to learn $\mathcal{C}$ with constant success probability is $\Omega(k(n - \log k))$.*

Our main results in this section are about the number of uniform *quantum* examples that are necessary and sufficient to exactly learn the class $\mathcal{C}$ of k-Fourier-sparse functions. A uniform quantum example for a concept $c \in \mathcal{C}$ is the quantum state

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x, c(x)\rangle.$$

We prove the following two theorems here.

▶ **Theorem 9.** *For every $n > 0$ and $k \leq 2^n$, the number of uniform quantum examples that suffice to learn $\mathcal{C}$ with probability $\geq 2/3$ is $O(k^{1.5}(\log k)^2)$.*

In the theorem below we prove the following (non-matching) lower bound on the number of uniform quantum examples necessary to learn $\mathcal{C}$.

▶ **Theorem 10.** *For every $n > 0$, constant $c \in (0,1)$ and $k \leq 2^{cn}$, the number of uniform quantum examples necessary to learn $\mathcal{C}$ with constant success probability is $\Omega(k \log k)$.*

3.1 Upper bound on learning k-Fourier-sparse Boolean functions

We split our quantum learning algorithm into two phases. Suppose $c \in \mathcal{C}$ is the unknown concept, with Fourier dimension r. In the first phase the learner uses samples from the distribution $\{\widehat{c}(S)^2\}_{S \in \{0,1\}^n}$ to learn the Fourier span of c. In the second phase the learner uses uniform *classical* examples to learn c exactly, knowing its Fourier span. Phase 1 uses $O(k(\log k)^2)$ uniform quantum examples (for Fourier-sampling) and phase 2 uses $O(rk \log k)$ uniform *classical* examples. Note that since $r \geq \log k$, phase 2 of our learner is always at least as expensive as phase 1.

▶ **Theorem 11.** *Let $k, r > 0$. There exists a quantum learner that exactly learns (with high probability) an unknown k-Fourier-sparse $c : \{0,1\}^n \to \{-1,1\}$ with Fourier dimension upper bounded by some known r, from $O(rk \log k)$ uniform quantum examples.*

The learner may not know the exact Fourier dimension r in advance, but Theorem 1 gives an upper bound $r = O(\sqrt{k} \log k)$, so our Theorem 9 follows immediately from Theorem 11.

3.1.1 Phase 1: Learning the Fourier span

A crucial ingredient that we use in phase 1 of our quantum learning algorithm is an improvement of Chang's lemma [13, 22] for k-Fourier-sparse Boolean functions. The original lemma upper bounds the dimension of the span of the "large" Fourier coefficients as follows.

▶ **Lemma 12** (Chang's lemma). *Let $\alpha \in (0,1)$ and $\rho > 0$. For every $f : \{0,1\}^n \to \{-1,1\}$ that satisfies $\widehat{f}(0^n) = 1 - 2\alpha$, we have*

$$\dim(\mathrm{span}\{S : |\widehat{f}(S)| \geq \rho\alpha\}) \leq \frac{2\log(1/\alpha)}{\rho^2}. \tag{2}$$

Let us consider Chang's lemma for k-Fourier-sparse Boolean functions. In particular, consider the case $\rho\alpha = 1/k$. In that case, since all elements of the Fourier support satisfy $|\widehat{f}(S)| \geq 1/k$ by Lemma 2, the left-hand side of Eq. (2) equals the Fourier dimension r of f. Chang's lemma gives

$$r \leq 2\alpha^2 k^2 \log k.$$

We now improve this upper bound on r nearly quadratically:

▶ **Theorem 13.** *Let $\alpha \in (0,1)$ and $k \geq 2$. For every k-Fourier-sparse $f : \{0,1\}^n \to \{-1,1\}$ that satisfies $\widehat{f}(0^n) = 1 - 2\alpha$ and $\mathrm{Fdim}(f) = r$, we have*

$$r \leq 2\alpha k \log k.$$

For a proof of this theorem, see the full version of the paper. We now illustrate how this theorem improves over Lemma 12. First, observe that $\alpha \geq 1/k$ (by Lemma 2), so $\alpha k \leq \alpha^2 k^2$. Second, consider a Boolean function f which satisfies $\alpha = 1/k^{3/4}$. Then, Chang's lemma (with $\rho = 1/k^{1/4}$) upper bounds the Fourier dimension of f as $r \leq O(\sqrt{k}\log k)$, which already follows from Theorem 1. Our Theorem 13 gives the much better upper bound $r \leq O(k^{1/4}\log k)$ in this case.

Now that we have a better understanding of the Fourier dimension of k-Fourier-sparse Boolean functions, we would like to understand how many Fourier samples suffice to obtain the Fourier span of f (in fact this will be our quantum learning algorithm for phase 1). Since the $\leq k$ squared non-zero Fourier coefficients of a k-Fourier-sparse function are each at least $1/k^2$, it is easy to see that after $O(k^2 \log k)$ Fourier samples we are likely to have seen every element in the Fourier support, and hence know the full Fourier support as well. We will improve on this easy bound below. The main idea is to show that if the span of the Fourier samples seen at a certain point has some dimension $r' < r$, then there is significant Fourier weight on elements outside of this span, so after a few more Fourier samples we will have grown the span. We now state this formally and prove the lemma.

▶ **Lemma 14.** *Let $n > 0$ and $1 \leq k \leq 2^n$. For every k-Fourier-sparse $f : \{0,1\}^n \to \{-1,1\}$ with Fourier span $\mathcal{V}$ and Fourier dimension r, the following holds: for every $r' > 0$ and $\mathcal{S} \subset \mathcal{V}$ satisfying $\dim(\mathrm{span}(\mathcal{S})) = r'$, we have*

$$\sum_{S \in \mathrm{span}(\mathcal{S})} \widehat{f}(S)^2 \leq 1 - \frac{r - r'}{k \log k}.$$

Proof. Let $B \in \mathbb{F}_2^{r \times r}$ be an invertible matrix such that the first $r' < r$ columns of B form a basis for $\mathrm{span}(\mathcal{S})$. By Lemma 5, f_B depends only on r bits, so we write $f_B : \{0,1\}^r \to \{-1,1\}$. Let $\mathcal{W} = \mathrm{span}\{e_1, \ldots, e_{r'}\} \subseteq \{0,1\}^r$. Then

$$\sum_{S \in \mathrm{span}(\mathcal{S})} \widehat{f}(S)^2 = \sum_{S \in \mathcal{W}} \widehat{f_B}(S)^2. \tag{3}$$

Let us decompose f_B as follows: $f_B(x_1,\ldots,x_r) = g(x_1,\ldots,x_{r'}) + g'(x_1,\ldots,x_r)$, where

$$g(y) = \sum_{T\in\{0,1\}^{r'}} \widehat{f_B}(T,0^{r-r'})\chi_T(y,0^{r-r'}) \quad \text{for every } y\in\{0,1\}^{r'}, \tag{4}$$

and

$$g'(x) = \sum_{S\notin\mathcal{W}} \widehat{f_B}(S)\chi_S(x) \quad \text{for every } x\in\{0,1\}^r.$$

Now by Parseval's identity we have

$$\mathbb{E}_{y\in\{0,1\}^{r'}}[g(y)^2] = \sum_{T\in\{0,1\}^{r'}} \widehat{g}(T)^2 = \sum_{S\in\mathcal{W}} \widehat{f_B}(S)^2, \tag{5}$$

where the second equality used Eq. (4). Combining Eq. (5) with an averaging argument, there exists an assignment of $a=(a_1,\ldots,a_{r'})\in\{0,1\}^{r'}$ to $(y_1,\ldots,y_{r'})$ such that

$$g(a_1,\ldots,a_{r'})^2 \geq \sum_{S\in\mathcal{W}} \widehat{f_B}(S)^2, \tag{6}$$

Consider the function h defined as

$$h(z_1,\ldots,z_{r-r'}) = f_B(a_1,\ldots,a_{r'},z_1,\ldots,z_{r-r'}) \quad \text{for every } z_1,\ldots,z_{r-r'}\in\{0,1\}. \tag{7}$$

Note that h has Fourier sparsity at most the Fourier sparsity of f_B, hence at most k. Also, the Fourier dimension of h is at most $r-r'$. Finally note that

$$\begin{aligned}
\widehat{h}(0^{r-r'}) &= \mathbb{E}_{z\in\{0,1\}^{r-r'}}[h(z)] \\
&= \mathbb{E}_{z\in\{0,1\}^{r-r'}}[f_B(a,z)] && \text{(by Eq. (7))} \\
&= \mathbb{E}_{z\in\{0,1\}^{r-r'}}\Big[\sum_{S_1\in\{0,1\}^{r'}}\sum_{S_2\in\{0,1\}^{r-r'}} \widehat{f_B}(S_1,S_2)\chi_{S_1}(a)\chi_{S_2}(z)\Big] \\
& && \text{(Fourier expansion of } f_B) \\
&= \sum_{S_1\in\{0,1\}^{r'}} \widehat{f_B}(S_1,0^{r-r'})\chi_{S_1}(a,0^{r-r'}) && \text{(using } \mathbb{E}_{z\in\{0,1\}^{r-r'}}\chi_S(z)=\delta_{S,0^{r-r'}}) \\
&= g(a_1,\ldots,a_{r'}) && \text{(by definition of } g \text{ in Eq. (4))} \\
&\geq \Big(\sum_{S\in\mathcal{W}} \widehat{f_B}(S)^2\Big)^{1/2}. && \text{(by Eq. (6))}
\end{aligned}$$

Using Theorem 13 for the function h, it follows that $\widehat{h}(0^{r-r'}) \leq 1-(r-r')/(k\log k)$, which in particular implies

$$\sum_{S\in\mathrm{span}(\mathcal{S})} \widehat{f}(S)^2 = \sum_{S\in\mathcal{W}} \widehat{f_B}(S)^2 \leq \widehat{h}(0^{r-r'})^2 \leq 1-\frac{r-r'}{k\log k},$$

where the first equality used Eq. (3). ◀

▶ **Theorem 15.** *For every k-Fourier-sparse Boolean function $f:\{-1,1\}^n\to\{-1,1\}$ with Fourier dimension r, its Fourier span can be learned using an expected number of $O(k\log k\log r)$ quantum examples.*

Proof. We only use the quantum examples for Fourier sampling; an expected number of two quantum examples suffices to get one Fourier sample. At any point of time let $\mathcal{S}$ be the set of samples we have received. Let the dimension of the span of $\mathcal{S}$ be r'. Now if we receive a new sample S such that $S \notin \text{span}(\mathcal{S})$, then the dimension of the samples we have seen increases by 1. By Lemma 14

$$\sum_{S \notin \text{span}(\mathcal{S})} \widehat{f}(S)^2 \geq \frac{r - r'}{k \log k}.$$

So the expected number of samples to increase the dimension by 1 is $\leq \frac{k \log k}{r - r'}$. Hence, the expected number of Fourier samples needed to learn the whole Fourier span of f is at most

$$\sum_{i=1}^{r} \frac{k \log k}{i} \leq O(k \log k \log r),$$

where the final inequality used $\sum_{i=1}^{r} \frac{1}{i} = O(\log r)$. ◀

3.1.2 Phase 2: Learning the function completely

In the above phase 1, the quantum learner obtains the Fourier span of c, which we will denote by $\mathcal{T}$. Using this, the learner can restrict to the following concept class

$$\mathcal{C}' = \{c : \{0,1\}^n \to \{-1,1\} \mid c \text{ is } k\text{-Fourier-sparse with Fourier span } \mathcal{T}\}$$

Let $\dim(\mathcal{T}) = r$. Let $B \in \mathbb{F}_2^{n \times n}$ be an invertible matrix whose first r columns of B form a basis for $\mathcal{T}$. Consider $c_B = c \circ (B^{-1})^{\mathsf{T}}$ for $c \in \mathcal{C}'$. By Lemma 5 it follows that c_B depends on only its first r bits, and we can write $c_B : \{0,1\}^r \to \{-1,1\}$. Hence the learner can apply the transformation $c \mapsto c \circ (B^{-1})^{\mathsf{T}}$ for every $c \in \mathcal{C}'$ and restrict to the concept class

$$\mathcal{C}'_r = \{c' : \{0,1\}^r \to \{-1,1\} \mid c' = c \circ (B^{-1})^{\mathsf{T}} \text{ for some } c \in \mathcal{C}' \text{ and invertible } B\}.$$

We now conclude phase 2 of the algorithm by invoking the classical upper bound of Haviv-Regev (Theorem 7) which says that $O(rk \log k)$ uniform classical examples of the form $(z, c'(z)) \in \{0,1\}^{r+1}$ suffice to learn $\mathcal{C}'_r$. Although we assume our learning algorithm has access to uniform examples of the form $(x, c(x))$ for $x \in \{0,1\}^n$, the quantum learner knows B and hence can obtain a uniform example $(z, c'(z))$ for c' by letting z be the first r bits of $B^{\mathsf{T}}x$ and $c'(z) = c(x)$.

3.2 Lower bound on learning k-Fourier-sparse Boolean functions

We show that $\Omega(k \log k)$ uniform quantum examples are necessary to learn the concept class of k-Fourier-sparse Boolean functions. See the full version of the paper for the proof.

▶ **Theorem 16.** *For every n, constant $c \in (0,1)$ and $k \leq 2^{cn}$, the number of uniform quantum examples necessary to learn the class of k-Fourier-sparse Boolean functions, with success probability $\geq 2/3$, is $\Omega(k \log k)$.*

4 Quantum vs classical membership queries

In this section we assume we can access the target function using membership queries rather than examples. Our goal is to simulate quantum exact learners for a concept class $\mathcal{C}$ by classical exact learners, without using many more membership queries. A key tool here

will be the ("nonnegative" or "positive-weights") adversary method. This was introduced by Ambainis [2]; here we will use the formulation of Barnum et al. [7], which is called the "spectral adversary" in the survey [31].

Let $\mathcal{C} \subseteq \{0,1\}^N$ be a set of strings. If $N = 2^n$ then we may view such a string $c \in \mathcal{C}$ as (the truth-table of) an n-bit Boolean function, but in this section we do not need the additional structure of functions on the Boolean cube and may consider any positive integer N. Suppose we want to identify an unknown $c \in \mathcal{C}$ with success probability at least $2/3$ (i.e., we want to compute the identity function on $\mathcal{C}$). The required number of quantum queries to c can be lower bounded as follows. Let Γ be a $|\mathcal{C}| \times |\mathcal{C}|$ matrix with real, nonnegative entries and 0s on the diagonal (called an "adversary matrix"). Let D_i denote the $|\mathcal{C}| \times |\mathcal{C}|$ 0/1-matrix whose (c,c')-entry is $[c_i \neq c'_i]$.[6] Then it is known that at least (a constant factor times) $\| \Gamma \| / \max_{i \in [N]} \| \Gamma \circ D_i \|$ quantum queries are needed, where $\| \cdot \|$ denotes operator norm (largest singular value) and '$\circ$' denotes entrywise product of matrices. Let

$$\mathrm{ADV}(\mathcal{C}) = \max_{\Gamma \geq 0} \frac{\| \Gamma \|}{\max_{i \in [N]} \| \Gamma \circ D_i \|}$$

denote the best-possible lower bound on $Q(\mathcal{C})$ that can be achieved this way.

The key to our classical simulation is the next lemma. It shows that if $Q(\mathcal{C})$ (and hence $\mathrm{ADV}(\mathcal{C})$) is small, then there is a query that splits the concept class in a "mildly balanced" way.

▶ **Lemma 17.** *For $N \geq 1$, let $\mathcal{C} \subseteq \{0,1\}^N$ be a concept class and suppose $\mathrm{ADV}(\mathcal{C}) = \max_{\Gamma \geq 0} \| \Gamma \| / \max_{i \in [N]} \| \Gamma \circ D_i \|$ is the nonnegative adversary bound for the exact learning problem corresponding to $\mathcal{C}$. Let μ be a distribution on $\mathcal{C}$ such that $\max_{c \in \mathcal{C}} \mu(c) \leq 5/6$, and let $\mathbf{C}$ be a random variable distributed according to μ. Then there exists an $i \in [N]$ such that*

$$\min(\mu(\mathbf{C}_i = 0), \mu(\mathbf{C}_i = 1)) \geq \frac{1}{36 \mathrm{ADV}(\mathcal{C})^2}.$$

Proof. Define unit vector $v \in \mathbb{R}_+^{|\mathcal{C}|}$ by $v_c = \sqrt{\mu(c)}$, and adversary matrix

$$\Gamma = vv^* - \mathrm{diag}(\mu),$$

where $\mathrm{diag}(\mu)$ is the diagonal matrix that has the entries of μ on its diagonal. This Γ is a nonnegative matrix with 0 diagonal (and hence a valid adversary matrix for the exact learning problem), and $\| \Gamma \| \geq \| vv^* \| - \| \mathrm{diag}(\mu) \| \geq 1 - 5/6 = 1/6$. Abbreviate $A = \mathrm{ADV}(\mathcal{C})$. By definition of A, we have for this particular Γ

$$A \geq \frac{\| \Gamma \|}{\max_i \| \Gamma \circ D_i \|} \geq \frac{1}{6 \max_i \| \Gamma \circ D_i \|},$$

hence there exists an $i \in [N]$ such that $\| \Gamma \circ D_i \| \geq \frac{1}{6A}$. We can write $v = \begin{pmatrix} v_0 \\ v_1 \end{pmatrix}$ where the entries of v_0 are the ones corresponding to cs where $c_i = 0$, and the entries of v_1 are the ones where $c_i = 1$. Then

$$\Gamma = \begin{pmatrix} v_0 v_0^* & v_0 v_1^* \\ v_1 v_0^* & v_1 v_1^* \end{pmatrix} - \mathrm{diag}(\mu) \quad \text{and} \quad \Gamma \circ D_i = \begin{pmatrix} 0 & v_0 v_1^* \\ v_1 v_0^* & 0 \end{pmatrix}.$$

[6] The bracket-notation $[P]$ denotes the truth-value of proposition P.

It is easy to see that $\| \Gamma \circ D_i \| = \| v_0 \| \cdot \| v_1 \|$. Hence

$$\frac{1}{36A^2} \leq \| \Gamma \circ D_i \|^2 = \| v_0 \|^2 \| v_1 \|^2 = \mu(\mathbf{C}_i = 0)\mu(\mathbf{C}_i = 1) \leq \min(\mu(\mathbf{C}_i = 0), \mu(\mathbf{C}_i = 1)),$$

where the last inequality used $\max(\mu(\mathbf{C}_i = 0), \mu(\mathbf{C}_i = 1)) \leq 1$. ◀

Note that if we query the index i given by this lemma and remove from $\mathcal{C}$ the strings that are inconsistent with the query outcome, then we reduce the size of $\mathcal{C}$ by a factor $\leq 1 - \Omega(1/\mathrm{ADV}(\mathcal{C})^2)$. Repeating this $O(\mathrm{ADV}(\mathcal{C})^2 \log |\mathcal{C}|)$ times would reduce the size of $\mathcal{C}$ to 1, completing the learning task. However, we will see below that analyzing the same approach in terms of entropy gives a somewhat better upper bound on the number of queries.

▶ **Theorem 18.** *For $N \geq 1$, let $\mathcal{C} \subseteq \{0,1\}^N$ be a concept class and suppose $\mathrm{ADV}(\mathcal{C}) = \max_{\Gamma \geq 0} \| \Gamma \| / \max_{i \in [N]} \| \Gamma \circ D_i \|$ is the nonnegative adversary bound for the exact learning problem corresponding to $\mathcal{C}$. Then there exists a classical learner for the concept class $\mathcal{C}$ using $O\left(\frac{\mathrm{ADV}(\mathcal{C})^2}{\log \mathrm{ADV}(\mathcal{C})} \log |\mathcal{C}|\right)$ membership queries that identifies the target concept with probability $\geq 2/3$.*

Proof. Fix an arbitrary distribution μ on $\mathcal{C}$. We will construct a deterministic classical learner for $\mathcal{C}$ with success probability $\geq 2/3$ under μ. Since we can do this for every μ, the "Yao principle" [36] then implies the existence of a randomized learner that has success probability $\geq 2/3$ for every $c \in \mathcal{C}$.

Consider the following algorithm, whose input is an N-bit random variable $\mathbf{C} \sim \mu$:

1. Choose an i that maximizes $H(\mathbf{C}_i)$ and query that i.[7]
2. Update $\mathcal{C}$ and μ by restricting to the concepts that are consistent with the query outcome.
3. Goto 1.

The queried indices are themselves random variables, and we denote them by $\mathbf{I}_1, \mathbf{I}_2, \ldots$. We can think of t steps of this algorithm as generating a binary tree of depth t, where the different paths correspond to the different queries made by the algorithm and their binary outcomes.

Let P_t be the probability that, after t queries, our algorithm has reduced μ to a distribution that has weight $\geq 5/6$ on one particular c:

$$P_t = \sum_{i_1,\ldots,i_t \in [N],\ b \in \{0,1\}^t} \Pr[\mathbf{I}_1 = i_1, \ldots, \mathbf{I}_t = i_t, \mathbf{C}_{i_1} \ldots \mathbf{C}_{i_t} = b] \cdot [\exists c \in \mathcal{C} \text{ s.t. } \mu(c \mid \mathbf{C}_{i_1} \ldots \mathbf{C}_{i_t} = b) \geq 5/6].$$

Because restricting μ to a subset $\mathcal{C}' \subseteq \mathcal{C}$ cannot decrease probabilities of individual $c \in \mathcal{C}'$, this probability P_t is non-decreasing in t. Because N queries give us the target concept completely, we have $P_N = 1$. Let T be the smallest integer t for which $P_t \geq 5/6$. We will run our algorithm for T queries, and then output the c with highest probability under the restricted version of μ we now have. With μ-probability at least $5/6$, that c will have probability at least $5/6$ (under μ conditioned on the query-results). The overall error probability under μ is therefore $\leq 1/6 + 1/6 = 1/3$.

[7] Querying this i will give a fairly "balanced" reduction of the size of $\mathcal{C}$ irrespective of the outcome of the query. If there are several maximizing is, then choose the smallest i to make the algorithm deterministic.

It remains to upper bound T. To this end, define the following "energy function" in terms of conditional entropy:

$$\begin{aligned} E_t &= H(\mathbf{C} \mid \mathbf{C}_{\mathbf{I}_1}, \ldots, \mathbf{C}_{\mathbf{I}_t}) \\ &= \sum_{i_1,\ldots,i_t \in [N],\ b \in \{0,1\}^t} \Pr[\mathbf{I}_1 = i_1, \ldots, \mathbf{I}_t = i_t, \mathbf{C}_{i_1} \ldots \mathbf{C}_{i_t} = b] \cdot H(\mathbf{C} \mid \mathbf{C}_{i_1} \ldots \mathbf{C}_{i_t} = b). \end{aligned}$$

Because conditioning on a random variable cannot increase entropy, E_t is non-increasing in t. We now show that as long as $P_t < 5/6$, the energy shrinks significantly with each new query.

Let $i_1, ..., i_t$ and b be such that there is no c in $\mathcal{C}$ s.t. $\mu(c \mid \mathbf{C}_{i_1} \ldots \mathbf{C}_{i_t} = b) \geq 5/6$ (note that the μ-probability of getting such an $i_1, ..., i_t$ and b, is $1 - P_t$). Let μ' be μ restricted to the class $\mathcal{C}'$ of concepts c where $c_{i_1} \ldots c_{i_t} = b$. The nonnegative adversary bound for this restricted concept class is $A' = \mathrm{ADV}(\mathcal{C}') \leq \mathrm{ADV}(\mathcal{C}) = A$. Applying Lemma 17 to μ', there is an $i_{t+1} \in [N]$ with $p := \min(\mu'(\mathbf{C}_{i_{t+1}} = 0), \mu'(\mathbf{C}_{i_{t+1}} = 1)) \geq \frac{1}{36A'^2} \geq \frac{1}{36A^2}$. Note that $H(p) \geq \Omega(\log(A)/A^2)$. Hence

$$\begin{aligned} H(\mathbf{C} \mid \mathbf{C}_{i_1} \ldots \mathbf{C}_{i_t} = b) - H(C \mid \mathbf{C}_{i_1} \ldots \mathbf{C}_{i_t} = b, \mathbf{C}_{i_{t+1}}) &= H(\mathbf{C}_{i_{t+1}} \mid \mathbf{C}_{i_1} \ldots \mathbf{C}_{i_t} = b) \\ &\geq \Omega(\log(A)/A^2). \end{aligned}$$

This implies $E_t - E_{t+1} \geq (1-P_t) \cdot \Omega(\log(A)/A^2)$. In particular, as long as $P_t < 5/6$, the $(t+1)$st query shrinks E_t by at least $\frac{1}{6}\Omega(\log(A)/A^2) = \Omega(\log(A)/A^2)$. Since $E_0 = H(\mathbf{C}) \leq \log|\mathcal{C}|$ and E_t cannot shrink below 0, there can be at most $O\left(\frac{A^2}{\log A} \log|\mathcal{C}|\right)$ queries before P_t grows to $\geq 5/6$. ◀

Since $\mathrm{ADV}(\mathcal{C})$ lower bounds $Q(\mathcal{C})$, Theorem 18 implies the bound $R(\mathcal{C}) \leq O\left(\frac{Q(\mathcal{C})^2}{\log Q(\mathcal{C})} \log|\mathcal{C}|\right)$ claimed in our introduction. Note that this bound is tight up to a constant factor for the class of N-bit point functions, where $A = \Theta(\sqrt{N})$, $|\mathcal{C}| = N$, and $R(\mathcal{C}) = \Theta(N)$ classical queries are necessary and sufficient.

5 Future work

Neither of our two results is tight. As directions for future work, let us state two conjectures, one for each model:

- k-Fourier-sparse functions can be learned from $O(k \cdot \mathrm{polylog}(k))$ uniform quantum examples.
- For all concept classes $\mathcal{C}$ of Boolean-valued functions on a domain of size N we have: $R(\mathcal{C}) = O(Q(\mathcal{C})^2 + Q(\mathcal{C}) \log N)$.

References

1 J. Adcock, E. Allen, M. Day, S. Frick, J. Hinchliff, M. Johnson, S. Morley-Short, S. Pallister, A. Price, and S. Stanisic. Advances in quantum machine learning, 9 Dec 2015. arXiv:1512.02900.

2 A. Ambainis. Quantum lower bounds by quantum arguments. *Journal of Computer and System Sciences*, 64(4):750–767, 2002. Earlier version in STOC'00. arXiv:quant-ph/0002066.

3 S. Arunachalam, S. Chakraborty, T. Lee, M. Paraashar, and R. de Wolf. Two new results about quantum exact learning. arXiv:1810.00481.

4 S. Arunachalam and R. de Wolf. Guest Column: A Survey of Quantum Learning Theory. *SIGACT News*, 48(2):41–67, 2017. arXiv:1701.06806.

5 S. Arunachalam and R. de Wolf. Optimal Quantum Sample Complexity of Learning Algorithms. *Journal of Machine Learning Research*, 19, 2018. Earlier version in CCC'17. arXiv:1607.00932.

6 A. Atıcı and R. Servedio. Quantum Algorithms for Learning and Testing Juntas. *Quantum Information Processing*, 6(5):323–348, 2009. arXiv:0707.3479.

7 H. Barnum, M. Saks, and M. Szegedy. Quantum query complexity and semi-definite programming. In *Proceedings of 18th IEEE Conference on Computational Complexity*, pages 179–193, 2003.

8 E. Bernstein and U. Vazirani. Quantum Complexity Theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997. Earlier version in STOC'93.

9 J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd. Quantum machine learning. *Nature*, 549(7671), 2017. arXiv:1611.09347.

10 J. Bourgain. An improved estimate in the restricted isometry problem. In *Geometric Aspects of Functional Analysis*, volume 2116 of *Lecture Notes in Mathematics*, pages 65–70. Springer, 2014.

11 N. H. Bshouty and J. C. Jackson. Learning DNF over the Uniform Distribution Using a Quantum Example Oracle. *SIAM Journal on Computing*, 28(3):1136—-1153, 1999. Earlier version in COLT'95.

12 E. J. Candés and T. Tao. Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies? *IEEE Transactions on Information Theory*, 52(12):5406–5425, 2006.

13 M. C. Chang. A polynomial bound in Freiman's theorem. *Duke Mathematics Journal*, 113(3):399–419, 2002.

14 M. Cheraghchi, V. Guruswami, and A. Velingker. Restricted Isometry of Fourier Matrices and List Decodability of Random Linear Codes. *SIAM Journal on Computing*, 42(5):1888–1914, 2013.

15 T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.

16 V. Dunjko and H. Briegel. Machine learning & artificial intelligence in the quantum domain, 8 Sep 2017. arXiv:1709.02779.

17 P. Gopalan, R. O'Donnell, R. A. Servedio, A. Shpilka, and K. Wimmer. Testing Fourier Dimensionality and Sparsity. *SIAM Journal on Computing*, 40(4):1075–1100, 2011. Earlier version in ICALP'09.

18 L. K. Grover. A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of 28th ACM STOC*, pages 212–219, 1996. arXiv:quant-ph/9605043.

19 A. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for solving linear systems of equations. *Physical Review Letters*, 103(15):150502, 2009. arXiv:0811.3171.

20 H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Nearly optimal sparse Fourier transform. In *Proceedings of 44th ACM STOC*, pages 563–578, 2012.

21 I. Haviv and O. Regev. The List-Decoding Size of Fourier-Sparse Boolean Functions. *ACM Transactions on Computation Theory*, 8(3):10:1–10:14, 2016. Earlier version in CCC'15. arXiv:1504.01649.

22 R. Impagliazzo, C. Moore, and A. Russell. An Entropic Proof of Chang's Inequality. *SIAM Journal of Discrete Mathematics*, 28(1):173–176, 2014. arXiv:1205.0263.

23 P. Indyk and M. Kapralov. Sample-Optimal Fourier Sampling in Any Constant Dimension. In *Proceedings of 55th IEEE FOCS*, pages 514–523, 2014.

24 E. Mossel, R. O'Donnell, and R. Servedio. Learning functions of k relevant variables. *Journal of Computer and System Sciences*, 69(3):421–434, 2004. Earlier version in STOC'03.

25 M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

26 R. O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

27 M. Rudelson and R. Vershynin. On sparse reconstruction from Fourier and Gaussian measurements. *Communications on Pure and Applied Mathematics*, 61(8):1025–1045, 2008.

28 S. Sanyal. Near-Optimal Upper Bound on Fourier Dimension of Boolean Functions in Terms of Fourier Sparsity. In *Proceedings of 42nd ICALP*, pages 1035–1045, 2015.

29 M. Schuld, I. Sinayskiy, and F. Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2015. arXiv:1409.3097.

30 R. Servedio and S. Gortler. Equivalences and Separations Between Quantum and Classical Learnability. *SIAM Journal on Computing*, 33(5):1067–1092, 2004. Combines earlier papers from ICALP'01 and CCC'01. quant-ph/0007036.

31 R. Špalek and M. Szegedy. All Quantum Adversary Methods are Equivalent. In *Proceedings of 32nd ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 1299–1311, 2005. arXiv:quant-ph/0409116.

32 L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134—-1142, 1984.

33 K. A. Verbeurgt. Learning DNF under the uniform distribution in quasi-polynomial time. In *Proceedings of 3rd Annual Workshop on Computational Learning Theory (COLT'90)*, pages 314–326, 1990.

34 P. Wittek. *Quantum Machine Learning: What Quantum Computing Means to Data Mining*. Elsevier, 2014.

35 R. de Wolf. A Brief Introduction to Fourier Analysis on the Boolean Cube. *Theory of Computing*, 2008. ToC Library, Graduate Surveys 1.

36 A. C-C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of 18th IEEE FOCS*, pages 222–227, 1977.

When Algorithms for Maximal Independent Set and Maximal Matching Run in Sublinear Time

Sepehr Assadi
Department of Computer Science, Princeton University, NJ, USA
sassadi@princeton.edu

Shay Solomon
School of Electrical Engineering, Tel Aviv University, Israel
shayso@post.tau.ac.il

Abstract

Maximal independent set (MIS), maximal matching (MM), and $(\Delta+1)$-(vertex) coloring in graphs of maximum degree Δ are among the most prominent algorithmic graph theory problems. They are all solvable by a simple linear-time greedy algorithm and up until very recently this constituted the state-of-the-art. In SODA 2019, Assadi, Chen, and Khanna gave a randomized algorithm for $(\Delta+1)$-coloring that runs in $\widetilde{O}(n\sqrt{n})$ time[1], which even for moderately dense graphs is sublinear in the input size. The work of Assadi et al. however contained a spoiler for MIS and MM: neither problems provably admits a sublinear-time algorithm in general graphs. In this work, we dig deeper into the possibility of achieving sublinear-time algorithms for MIS and MM.

The neighborhood independence number of a graph G, denoted by $\beta(G)$, is the size of the largest independent set in the neighborhood of any vertex. We identify $\beta(G)$ as the "right" parameter to measure the runtime of MIS and MM algorithms: Although graphs of bounded neighborhood independence may be very dense (clique is one example), we prove that carefully chosen variants of greedy algorithms for MIS and MM run in $O(n\beta(G))$ and $O(n\log n\cdot\beta(G))$ time respectively on any n-vertex graph G. We complement this positive result by observing that a simple extension of the lower bound of Assadi et al. implies that $\Omega(n\beta(G))$ time is also necessary for any algorithm to either problem for all values of $\beta(G)$ from 1 to $\Theta(n)$. We note that our algorithm for MIS is deterministic while for MM we use randomization which we prove is unavoidable: any deterministic algorithm for MM requires $\Omega(n^2)$ time even for $\beta(G)=2$.

Graphs with bounded neighborhood independence, already for constant $\beta=\beta(G)$, constitute a rich family of possibly dense graphs, including line graphs, proper interval graphs, unit-disk graphs, claw-free graphs, and graphs of bounded growth. Our results suggest that even though MIS and MM do not admit sublinear-time algorithms in general graphs, one can still solve both problems in sublinear time for a wide range of $\beta(G)\ll n$.

Finally, by observing that the lower bound of $\Omega(n\sqrt{n})$ time for $(\Delta+1)$-coloring due to Assadi et al. applies to graphs of (small) constant neighborhood independence, we unveil an intriguing separation between the time complexity of MIS and MM, and that of $(\Delta+1)$-coloring: while the time complexity of MIS and MM is strictly higher than that of $(\Delta+1)$ coloring in general graphs, the exact opposite relation holds for graphs with small neighborhood independence.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

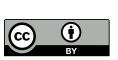
Keywords and phrases Maximal Independent Set, Maximal Matching, Sublinear-Time Algorithms, Bounded Neighborhood Independence

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.17

Category Track A: Algorithms, Complexity and Games

Funding *Sepehr Assadi*: Research supported in part by the Simons Collaboration on Algorithms and Geometry.

[1] Here, and throughout the paper, we define $\widetilde{O}(f(n)) := O(f(n)\cdot\text{polylog}(n))$ to suppress log-factors.



46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 17; pp. 17:1–17:17

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

1 Introduction

Maximal independent set (MIS) and maximal matching (MM) are two of the most prominent graph problems with a wide range of applications in particular to symmetry breaking. Algorithmic study of these problems can be traced back to at least four decades ago in the pioneering work of [33, 41, 1, 32] on PRAM algorithms. These problems have since been studied extensively in various models including distributed algorithms [40, 46, 31, 34, 38, 10, 9, 21, 19], dynamic algorithms [43, 11, 50, 45, 5, 6], streaming algorithms [18, 27, 15, 4], massively parallel computation (MPC) algorithms [37, 22, 12], local computation algorithms (LCA) [48, 2, 21, 20, 39, 23], and numerous others.

In this paper, we consider the **time complexity** of MIS and MM (in the centralized setting) and focus on one of the most basic questions regarding these two problems:

How fast can we solve maximal independent set and maximal matching problems?

At first glance, the answer to this question may sound obvious: there are text-book greedy algorithms for both problems that run in linear time and "of course" one cannot solve these problems faster as just reading the input takes linear time. This answer however is not quite warranted: for the closely related problem of $(\Delta+1)$-(vertex) coloring, very recently Assadi, Chen, and Khanna [4] gave a randomized algorithm that runs in only $\widetilde{O}(n\sqrt{n})$ time on any n-vertex graph with high probability[2]. This means that even for moderately dense graphs, one can indeed color the graph faster than reading the entire input, i.e., in sublinear time.

The Assadi-Chen-Khanna algorithm hints that one could perhaps hope for sublinear-time algorithms for MIS and MM as well. Unfortunately however, the work of [4] already contained a spoiler: neither MIS nor MM admits a sublinear-time algorithm in general graphs.

In this work, we show that despite the negative result of [4] for MIS and MM, the hope for obtaining sublinear-time algorithms for these problems need not be short lived. In particular, we identify a key parameter of the graph, namely the *neighborhood independence number*, that provides a more nuanced measure of runtime for these problems and show that both problems can be solved much faster when neighborhood independence is small. This in turn gives rise to sublinear-time algorithms for MIS and MM on a rich family of graphs with bounded neighborhood independence. In the following, we elaborate more on our results.

1.1 Our Contributions

For a graph $G(V,E)$, the neighborhood independence number of G, denoted by $\beta(G)$, is defined as the size of the largest independent set in the graph in which all vertices of the independent set are incident on some shared vertex $v \in V$. Our main result is as follows:

▷ Result 1. There exist algorithms that given a graph $G(V,E)$ find (i) a maximal independent set of G deterministically in $O(n \cdot \beta(G))$ time, and (ii) a maximal matching of G randomly in $O(n \log n \cdot \beta(G))$ time in expectation and with high probability.

When considering sublinear-time algorithms, specifying the exact data model is important as the algorithm cannot even read the entire input once. We assume that the input graph is presented in the adjacency array representation, i.e., for each vertex $v \in V$, we are given degree $\mathsf{deg}(v)$ of v followed by an array of length $\mathsf{deg}(v)$ containing all neighbors of v in

[2] We say an event happens with high probability if it happens with probability at least $1 - 1/\text{poly}(n)$.

arbitrary order. This way, we can access the degree of any vertex v or its i-th neighbor for $i \in [\mathsf{deg}(v)]$ in $O(1)$ time. We also make the common assumption that a random number from 1 to n can be generated in $O(1)$ time. This is a standard input representation for graph problems and is commonly used in the area of sublinear-time algorithms (see, e.g. [25, 26, 44]). We elaborate on several aspects of Result 1 in the following.

Optimality of Our Bounds. Assadi et al. [4] proved that any algorithm for MIS or MM requires $\Omega(n^2)$ time in general[3]. These lower bounds can be extended in an easy way to prove that $\Omega(n \cdot \beta)$ time is also necessary for both problems on graphs with neighborhood independence $\beta(G) = \beta$. Indeed, independently sample $t := n/\beta$ graphs $G_1, \ldots, G_t$ each on β vertices from the hard distribution of graphs in [4] and let G be the union of these graphs. Clearly, $\beta(G) \leq \beta$ and it follows that since solving MIS or MM on each graph G_i requires $\Omega(\beta^2)$ time by the lower bound of [4], solving t independent copies requires $\Omega(t \cdot \beta^2) = \Omega(n\beta)$ time. As such, our Result 1 is optimal for every β ranging from a constant to $\Theta(n)$ (up to a constant factor for MIS and $O(\log n)$ for MM).

Our Algorithms. Both our algorithms for MIS and MM in Result 1 are similar to the standard greedy algorithms, though they require careful adjustments and implementation. Specifically, the algorithm for MIS is the standard deterministic greedy algorithm (with minimal modification) and for MM we use a careful implementation of the (modified) randomized greedy algorithm (see, e.g. [16, 3, 42, 47]). The novelty of our work mainly lies in the analysis of these algorithms. We show, perhaps surprisingly, that already-known algorithms can in fact achieve an improved performance and run in sublinear-time for graphs with bounded neighborhood independence *even* when the value of $\beta(G)$ is unknown to the algorithms. Combined with the optimality of our bounds mentioned earlier, we believe that this makes neighborhood independence number an ideal parameter for measuring the runtime of MIS and MM algorithms.

Determinism and Randomization. Our MIS algorithm in Result 1 is deterministic which is a rare occurrence in the realm of sublinear-time algorithms. But for MM, we again fall back on randomization to achieve sublinear-time performance. This is not a coincidence however: we prove in Theorem 11 that any deterministic algorithm for MM requires $\Omega(n^2)$ time even on graphs with constant neighborhood independence number. This also suggests a separation in the time complexity of MIS and MM for deterministic algorithms.

Bounded Neighborhood Independence. Our Result 1 is particularly interesting for graphs with constant neighborhood independence as we obtain quite fast algorithms with running time $O(n)$ and $O(n \log n)$ for MIS and MM, respectively. Graphs with constant neighborhood independence capture a rich family of graphs; several illustrative examples are as follows:

- *Line graphs:* For any arbitrary graph G, the neighborhood independence number of its line graph $L(G)$ is at most 2. More generally, for any r-hyper graph $\mathcal{H}$ in which each hyper-edge connects at most r vertices, $\beta(L(\mathcal{H})) \leq r$.

[3] We note that the lower bound for MIS only appears in the full version of [4].

- *Bounded-growth graphs:* A graph $G(V, E)$ is said to be of bounded growth iff there exists a function such that for every vertex $v \in V$ and integer $r \geq 1$, the size of the largest independent set in the r-neighborhood of v is bounded by $f(r)$. Bounded-growth graphs in turn capture several intersection graphs of geometrical objects such as proper interval graphs [30], unit-disk graphs [28], quasi-unit-disk graphs [36], and general disc graphs [29].
- *Claw-free graphs:* Graphs with neighborhood independence β can be alternatively defined as β-claw-free graphs, i.e., graphs that do not contain $K_{1,\beta}$ as an induced subgraph. Claw-free graphs have been subject of extensive study in structural graph theory; see the series of papers by Chudnovsky and Seymour, starting with [14], and the survey by Faudree et al. [17].

Above graphs appear naturally in the context of symmetry breaking problems (for instance in the study of wireless networks), and there have been numerous works on MIS and MM in graphs with bounded neighborhood independence and their special cases (see, e.g. [36, 49, 28, 7, 8, 29, 19] and references therein).

1.2 Other Implications

Despite the simplicity of our algorithms in Result 1, they lead to several interesting implications, when combined with some known results and/or techniques:

(a) *Approximate vertex cover and matching:* Our MM algorithm in Result 1 combined with well-known properties of maximal matchings implies an $O(n \log n \cdot \beta(G))$ time 2-approximation algorithm to both maximum matching and minimum vertex cover. For graphs with constant neighborhood independence, our results improve upon the sublinear-time algorithms of [44] that achieve $(2+\varepsilon)$-approximation to the *size* of the optimal solution to both problems but did not find the actual edges or vertices in $\widetilde{O}_\varepsilon(n)$ time on general graphs.

(b) *Caro-Wei bound and approximation of maximum independent set:* The Caro-Wei bound [13, 51] states that any graph $G(V, E)$ contains an independent set of size at least $\sum_{v \in V} \frac{1}{\mathsf{deg}(v)+1}$, and there is a substantial interest in obtaining independent sets of this size (see, e.g. [27, 29, 15] and references therein). One standard way of obtaining such independent set is to run the greedy MIS algorithm on the vertices of the graph in the increasing order of their degrees. As our Result 1 implies that one can implement the greedy MIS algorithm for *any* ordering of vertices, we can sort the vertices in $O(n)$ time and then run our deterministic algorithm with this order to obtain an independent set with Caro-Wei bound size in $O(n\beta(G))$ time. Additionally, it is easy to see that on graphs with $\beta(G) = \beta$, any MIS is a β-approximation to the *maximum* independent set (see, e.g. [35, 49]). We hence also obtain a constant factor approximation in $O(n)$ time for maximum independent set on graphs with bounded neighborhood independence.

(c) *Separation of $(\Delta+1)$-coloring with MIS and MM:* Assadi et al. [4] gave an $\widetilde{O}(n\sqrt{n})$ time algorithm for $(\Delta+1)$ coloring and an $\Omega(n^2)$ time lower bound for MIS and MM on general graphs. It is also shown in [4] that $(\Delta+1)$ coloring requires $\Omega(n\sqrt{n})$ time and in fact the lower bound holds for graphs with constant neighborhood independence. Together with our Result 1, this implies an interesting separation between the time-complexity of MIS and MM, and that of $(\Delta+1)$-coloring: while the time complexity of MIS and MM is strictly higher than that of $(\Delta+1)$ coloring in general graphs, the exact opposite relation holds for graphs with small neighborhood independence number.

(d) *Efficient MM computation via MIS on line graphs:* The line graph $L(G)$ of a graph G contains m vertices corresponding to edges of G and up to $O(mn)$ edges. Moreover, for any graph G, $\beta(L(G)) \leq 2$. As an MIS in $L(G)$ corresponds to an MM in G, our results suggest that despite the larger size of $L(G)$, perhaps surprisingly, computing an MM of G through computing an MIS for $L(G)$ is just as efficient as directly computing an MM of G (assuming direct access to $L(G)$). This observation may come into play in real-life situations where there is no direct access to the graph but rather only to its line graph.

Preliminaries and Notation

For a graph $G(V, E)$ and vertex $v \in V$, $N(v)$ and $\mathsf{deg}(v)$ denote the neighbor-set and degree of a vertex v, respectively. For a subset $U \subseteq V$, $\mathsf{deg}_U(v)$ denotes the degree of v to vertices in U. Denote by $\beta(G)$ the neighborhood independence number of graph G.

2 Technical and Conceptual Highlights

Our first (non-technical) contribution is in identifying the neighborhood independence number as the "right" measure of time-complexity for both MIS and MM. We then show that surprisingly simple algorithms for these problems run in sublinear-time on graphs with bounded $\beta(G)$.

The textbook greedy algorithm for MIS works as follows: scan the vertices in an arbitrary order and add each scanned vertex to a set $\mathcal{M}$ iff it does not already have a neighbor in $\mathcal{M}$. Clearly the runtime of this algorithm is $\Theta(\sum_{v \in V} \mathsf{deg}(v)) = \Theta(m)$ and this bound does not improve for graphs with small β. We can slightly tweak this algorithm by making every vertex that joins $\mathcal{M}$ to *mark* all its neighbors and simply ignore scanning the already marked vertices. This tweak however is not useful in general graphs as the algorithm may waste time by repeatedly marking the same vertices over and over again without making much further progress (the complete bipartite graph is an extreme example). The same problem manifests itself in other algorithms, including those for MM, and is at the root of the lower bounds in [4] for sublinear-time computation of MIS and MM.

We prove that this issue cannot arise in graphs with bounded neighborhood independence. Noting that the runtime of the greedy MIS algorithm that uses "marks" is $\Theta(m_{\mathcal{M}})$, where we define $m_{\mathcal{M}} := \sum_{v \in \mathcal{M}} \mathsf{deg}(v)$, a key observation is that $m_{\mathcal{M}}$ is much smaller than m when β is small. Indeed, as the vertices of $\mathcal{M}$ form an independent set, all the edges incident on $\mathcal{M}$ lead to $V \setminus \mathcal{M}$, and so if $m_{\mathcal{M}}$ is large, then the *average* degree of $V \setminus \mathcal{M}$ *to* $\mathcal{M}$ cannot be "too small"; however, the latter average degree cannot be larger than β as otherwise there is some vertex in $V \setminus \mathcal{M}$ that is incident to more than β independent vertices, a contradiction. This is all we need to conclude that the runtime of the greedy MIS algorithm that uses marks is bounded by $O(n \cdot \beta)$.

Both the MIS algorithm and its analysis are remarkably simple, and *in hindsight*, this is not surprising since this parameter β is in a sense "tailored" to the MIS problem. Although MM and MIS problems are intimately connected to each other, the MM problem appears to be much more intricate for graphs with bounded neighborhood independence. Indeed, while the set U of unmatched vertices in any MM forms an independent set and hence total number m_U of edges incident on U cannot be too large by the above argument, the runtime of greedy or any other algorithm cannot be bounded in terms of m_U (as m_U can simply be zero). In fact, it is provably impossible to adjust our argument for MIS to the MM problem due to our lower bound for deterministic MM algorithms (Theorem 11) that shows that any such algorithm must incur a runtime of $\Omega(n^2)$ even for $\beta = 2$.

The main technical contribution of this paper is thus in obtaining a fast randomized MM algorithm for graphs with bounded β. Our starting point is the modified randomized greedy (MRG) algorithm of [16, 3] that finds an MM by iteratively picking an *unmatched* vertex u uniformly at random and matching it to a uniformly at random chosen *unmatched* neighbor $v \in N(u)$. On its own, this standard algorithm does not benefit from small values of β: while picking an unmatched vertex u is easy, finding an unmatched neighbor v for u is too time consuming in general. We instead make the following simple but crucial modification: instead of picking v from unmatched neighbors of u, we simply sample v from the set of *all* neighbors of u and only match it to u if it is also unmatched; otherwise we sample another vertex u and continue like this (additional care is needed to ensure that this process even terminates but we postpone the details to Section 4).

To analyze the runtime of this modified algorithm, we leverage the above argument for MIS and take it to the next step to prove a basic structural property of graphs with bounded neighborhood independence: for any set P of vertices, a constant fraction of vertices are such that their inner degree inside P is "not much smaller" than their total degree (depending both on β and size of P). Letting P to be the set of unmatched vertices in the above algorithm allows us to bound the number of iterations made by the algorithm before finding the next matching edge, and ultimately bounding the overall runtime of the algorithm by $O(n \log n \cdot \beta)$ both in expectation and with high probability.

Technical Comparison with Prior Work

Our work is most closely related to the $\widetilde{O}(n\sqrt{n})$-time $(\Delta+1)$-coloring algorithm of Assadi, Chen, and Khanna [4] (and their $\Omega(n^2)$ time lower bounds for MIS and MM on general graphs), as well as the series of work by Goel, Kapralov, and Khanna [25, 24, 26] on finding perfect matchings in regular bipartite graphs that culminated in an $O(n \log n)$ time algorithm.

The coloring algorithm of [4] works by *non-adaptively sparsifying* the graph into $O(n \log^2(n))$ edges in $\widetilde{O}(n\sqrt{n})$ time in such a way that a $(\Delta+1)$ coloring of the original graph can be found quickly from this sparsifier. The algorithms in [25, 24] were also based on the high-level idea of sparsification but the final work in this series [26] instead used a *(truncated) random walk* approach to speedup augmenting path computations in regular graphs. The sparsification methods used in [4, 25, 24] as well as the random walk approach of [26] are all quite different from our techniques in this paper that are tailored to graphs with bounded neighborhood independence. Moreover, even though every perfect matching is clearly maximal, our results and [25, 24, 26] are incomparable as d-regular bipartite graphs and graphs with bounded neighborhood independence are in a sense the exact opposite of each other: for a d-regular bipartite graph, $\beta(G) = d$ which is the largest possible for graphs with maximum degree d.

3 Maximal Independent Set

The standard greedy algorithm for MIS works as follows: Iterate over vertices of the graph in an arbitrary order and insert each one to an initially empty set $\mathcal{M}$ if none of its neighbors have already been inserted to $\mathcal{M}$. By the time all vertices have been processed, $\mathcal{M}$ clearly provides an MIS of the input graph. See Algorithm 1 for a pseudo-code.

We prove that this algorithm is fast on graphs with bounded neighborhood independence.

▶ **Theorem 1.** *The greedy MIS algorithm (as specified by Algorithm 1) computes a maximal independent set of a graph G given in adjacency array representation in $O(n \cdot \beta(G))$ time.*

Algorithm 1: The (Deterministic) Greedy Algorithm for Maximal Independent Set.

Input: An n-vertex graph $G(V, E)$ given in adjacency array representation.
Output: An MIS $\mathcal{M}$ of G.
Initialize $\mathcal{M} = \emptyset$ and $\texttt{mark}[v_i] = \text{FALSE}$ for all vertices $v_i \in V$ where $V := \{v_1, \ldots, v_n\}$.
for $i = 1$ ***to*** n **do**
  if $\texttt{mark}[v_i] = \text{FALSE}$ **then**
    add v_i to $\mathcal{M}$ and set $\texttt{mark}[u] = \text{TRUE}$ for all $u \in N(v_i)$.
  end
end
Return $\mathcal{M}$.

Proof. Let $G(V, E)$ be an arbitrary graph. Suppose we run Algorithm 1 on G and obtain $\mathcal{M}$ as the resulting MIS. To prove Theorem 1, we use the following two simple claims.

▷ Claim 2. The time spent by Algorithm 1 on a graph $G(V, E)$ is $O(n + \sum_{v \in \mathcal{M}} \mathsf{deg}(v))$.

Proof. Iterating over vertices in the **for** loop takes $O(n)$ time. Beyond that, for each vertex joining the MIS $\mathcal{M}$, we spend time that is linear in its degree to mark all its neighbors. ◁

▷ Claim 3. For any independent set $I \subseteq V$ in G, $\sum_{v \in I} \mathsf{deg}(v) \leq n \cdot \beta(G)$.

Proof. Let $E(I)$ denote the edges incident on vertices in the independent set I. Since I is an independent set, these edges connect vertices of I with vertices of $V \setminus I$. Suppose towards a contradiction that $|E(I)| = \sum_{v \in I} \mathsf{deg}(v) > n \cdot \beta(G)$. By a double counting argument, there must exist a vertex v in $V \setminus I$ with at least $|E(I)| / |V \setminus I| > \beta(G)$ neighbors in I. But since I is an independent set, this means that there exists an independent set of size $> \beta(G)$ in the neighborhood of v, which contradicts the fact that $\beta(G)$ is the neighborhood independence number of G. ◁

Theorem 1 now follows from Claims 2 and 3 as $\mathcal{M}$ is an independent set of G. ◀

4 Maximal Matching

We now consider the maximal matching (MM) problem. Similar to MIS, a standard greedy algorithm for MM is to iterate over the vertices in arbitrary order and match each vertex to one of their unmatched neighbors (if any). However, as we show in Section 5 this and any other *deterministic* algorithm for MM, cannot run in sublinear-time even when $\beta(G) = 2$.

We instead consider the following variant of the greedy algorithm, referred to as the *(modified) randomized greedy algorithm*, put forward by [16, 3] and extensively studied in the literature primarily with respect to its approximation ratio for the maximum matching problem (see, e.g. [42, 47] and the references therein). Pick an unmatched vertex u uniformly at random; pick an unmatched vertex v incident on u uniformly at random and add (u, v) to the matching M; repeat as long as there is an unmatched edge left in the graph. It is easy to see that at the end of the algorithm M will be an MM of G.

As it is, this algorithm is not suitable for our purpose as finding an unmatched vertex v incident on u is too costly. We thus instead consider the following variant which samples the set v from all neighbors of u and only match it to u if v is also unmatched (we also change the final check of the algorithm for maximality of M with a faster computation). See Algorithm 2 for a pseudo-code after proper modifications.

Algorithm 2: The (Modified) Randomized Greedy Algorithm for Maximal Matching.

Input: An n-vertex graph $G(V, E)$ given in adjacency array representation.
Output: A maximal matching M of G.
Initialize $M = \emptyset$ and $U = V$.
while $U \neq \emptyset$ **do**
  Define the threshold $\tau := \tau(U) = \frac{4n \cdot \beta(G)}{|U|}$.
  Sample a vertex u uniformly at random from U.
  if $\mathsf{deg}(u) < \tau$ **then**
    Choose a random vertex v from $N(u) \cap U$ (if non-empty), add (u, v) to M and set $U \leftarrow U \setminus \{u, v\}$. If $N(u) \cap U = \emptyset$, set $U \leftarrow U \setminus \{u\}$.
  else
    Sample a vertex v uniformly at random from $N(v)$.
    if $v \in U$ **then**
      add (u, v) to M and set $U \leftarrow U \setminus \{u, v\}$.
  end
end
Return M.

We remark that the first **if** condition in Algorithm 2 is used to remove the costly operation of checking if any unmatched edge is left in the graph. It is easy to see that this algorithm always output an MM.

We prove that Algorithm 2 is fast both in expectation and with high probability on graphs with bounded neighborhood independence. We also note that as stated, Algorithm 2 actually assumes knowledge of $\beta(G)$ (needed for the definition of the threshold parameter τ). However, we show at the end of this section that this assumption can be lifted easily and obtain a slight modification of Algorithm 2 with the same asymptotic runtime that does *not* require any knowledge of $\beta(G)$.

▶ **Theorem 4.** *The modified randomized greedy MM algorithm (as specified by Algorithm 2) computes a maximal matching of a graph G given in adjacency array representation in $O(n \log n \cdot \beta(G))$ time in expectation and with high probability.*

Let t denote the number of iterations of the **while** loop in Algorithm 2. We can bound the runtime of this algorithm based on t as follows.

▷ Claim 5. Algorithm 2 can be implemented in $O(n \log n \cdot \beta(G) + t)$ time.

Proof. First, we would like to store the set U in a data structure that supports random sampling and deletion of a vertex from U, as well as determining whether a vertex is currently in U or not, in constant time. This data structure can be easily implemented using two

arrays A_1 and A_2; we provide the rather tedious details for completeness. The arrays are initialized as $A_1[i] = A_2[i] = i$ for all $i = 1, \ldots, n$, where $A_2[i]$ holds the index of the cell in A_1 where v_i is stored, or -1 if v_i is not in U, while A_1 stores the vertices of U in its first $|U|$ cells, identifying each v_i with index i. When any unmatched vertex v_i is removed from U, we first use $A_2[i]$ to determine the cell where v_i is stored in A_1, then we move the unmatched vertex stored at the last cell in A_1, $A_1[|U|]$, to the cell currently occupied by v_i by setting $A_1[A_2[i]] = A_1[|U|]$, and finally set $A_2[A_1[|U|] = A_2[i], A_2[i] = -1$. Randomly sampling a vertex from U and determining whether a vertex belongs to U can now be done in $O(1)$ time.

Using these arrays each iteration of the **while** loop in which $\mathsf{deg}(u) \geq \tau$ can be carried out within $O(1)$ time. Iterations for which $\mathsf{deg}(u) < \tau$ are more costly, due to the need to determine $N(u) \cap U$. Nonetheless, in each such iteration we spend at most $O(\tau)$ time while at least one vertex is removed from U, hence the time required by all such iterations is bounded by $\sum_{k=1}^{n} O(\frac{n \cdot \beta(G)}{k}) = O(n \log n \cdot \beta(G))$. It follows that the total runtime of the algorithm is $O(n \log n \cdot \beta(G) + t)$. ◁

The main ingredient of the analysis is thus to bound the number t of iterations. Before proceeding we introduce some definition. We say that an iteration of the **while** loop *succeeds* iff we remove at least one vertex from U in this iteration. Clearly, there can be at most n successful iterations. We prove that each iteration of the algorithm is successful with a sufficiently large probability, using which we bound the total number of iterations.

To bound the success probability, we shall argue that for sufficiently many vertices u in U, the number of its neighbors in U, referred to as its *internal* degree, is proportionate to the number of its neighbors outside U, referred to as its *external* degree; for any such vertex u, a random neighbor v of u has a good chance of belonging to U. This is captured by the following definition.

▶ **Definition 6** (Good vertices). *For a parameter $\delta \in (0, 1)$, we say a vertex $u \in U$ is* δ-good *iff $\mathsf{deg}_U(u) \geq \delta \cdot \mathsf{deg}_{V \setminus U}(u)$ or $\mathsf{deg}(u) < 1/\delta$.*

Clearly, if we sample a δ-good vertex $u \in U$ in some iteration, then with probability $\geq \delta$ that iteration succeeds. It thus remains to show that many vertices in U are good for an appropriate choice of δ. We use the bounded neighborhood independence property (in a more sophisticated way than it was used in the proof of Theorem 1) to prove the following lemma, which lies at the core of the analysis.

▶ **Lemma 7.** *Fix any choice of U in some iteration and let $\delta := \delta(U) = 1/\tau(U)$ (for the parameter $\tau(U)$ defined in Algorithm 2). Then at least half the vertices in U are δ-good in this iteration.*

Proof. Let us say that a vertex $u \in U$ is *bad* iff it is not δ-good (for the parameter δ in the lemma statement). Let B denote the set of bad vertices and let $b := |B|$.

▷ Claim 8. There exists an independent set $I \subseteq B$ with at least $\frac{b}{2\delta}$ edges leading to $V \setminus U$.

Proof. We prove this claim using a probabilistic argument. Pick a random permutation σ of vertices in B and add each vertex $v \in B$ to an initially empty independent set $I = I_\sigma$ iff v appears before all its neighbors in B according to σ. Clearly, the resulting set I is an independent set inside B.

Let $E(I)$ denote the set of edges that connect vertices of I with vertices of $V \setminus U$. For any vertex $v \in B$, define a random variable $D_v \in \left\{0, \mathsf{deg}_{V \setminus U}(v)\right\}$ which takes value equal to

the external degree of v iff v is added to I. Clearly, $|E(I)| = \sum_{v \in V} D_v$. We have,

$$\begin{aligned}\mathbb{E}\,|E(I)| &= \sum_{v \in B} \mathbb{E}\,[D_v] = \sum_{v \in B} \mathbf{Pr}\,(v \in I) \cdot \mathsf{deg}_{V \setminus U}(v) \\ &= \sum_{v \in B} \frac{1}{\mathsf{deg}_B(v)+1} \cdot \mathsf{deg}_{V \setminus U}(v) \\ &\quad (v \text{ is only chosen in } I \text{ iff it is ranked first by } \sigma \text{ among itself and its } \mathsf{deg}_B(v) \text{ neighbors}) \\ &\geq \sum_{v \in B} \frac{1}{\delta \cdot \mathsf{deg}_{V \setminus U}(v)+1} \cdot \mathsf{deg}_{V \setminus U}(v) \qquad (\text{as } B \subseteq U \text{ and vertices in } B \text{ are all bad}) \\ &\geq \sum_{v \in B} \frac{1}{2\delta \cdot \mathsf{deg}_{V \setminus U}(v)} \cdot \mathsf{deg}_{V \setminus U}(v) = \frac{b}{2\delta}. \\ &\qquad\qquad (\text{as } \mathsf{deg}(v) \geq \tfrac{1}{\delta} \text{ and hence } \delta \cdot \mathsf{deg}_{V \setminus U}(v) \geq 1)\end{aligned}$$

It follows that there exists a permutation σ for which the corresponding independent set $I = I_\sigma \subseteq B$ has at least $\frac{b}{2\delta}$ edges leading to $V \setminus U$, finalizing the proof. ◁

We next prove Lemma 7 using an argument akin to Claim 3 in the proof of Theorem 1. Let I be the independent set guaranteed by Claim 8. As at least $\frac{b}{2\delta}$ edges are going from I to $V \setminus U$, a double counting argument implies that there exists a vertex $v \in V \setminus U$ with degree to I satisfying:

$$\mathsf{deg}_I(v) \geq \frac{b}{2\delta \cdot |V \setminus U|} \geq \frac{b \cdot \tau(U)}{2n} = \frac{2b \cdot \beta(G)}{|U|}, \tag{1}$$

where the equality is by the choice of $\tau(U)$. Suppose towards a contradiction that $b > |U|/2$. This combined with Eq (1) implies that $\mathsf{deg}_I(v) > \beta(G)$. Since I is an independent set, it follows that $N(v)$ contains an independent set of size larger than $\beta(G)$, a contradiction. Hence $b \leq |U|/2$, and so at least half the vertices in U are δ-good, as required. ◀

We now use Lemma 7 to bound the expected number of iterations t in Algorithm 2. Let us define the random variables $X_1, \ldots, X_n$, where X_k denotes the number of iterations spent by the algorithm when $|U| = k$. Clearly, the total number of iterations $t = \sum_{k=1}^{n} X_k$. We use these random variables to bound the expected value of t in the following claim. The next claim then proves a concentration bound for t to obtain the high probability result.

▷ Claim 9. The number of iterations in Algorithm 2 is in expectation $\mathbb{E}\,[t] \leq 8\beta(G) \cdot n \log n$.

Proof. As stated above, $\mathbb{E}\,[t] = \sum_{k=1}^{n} \mathbb{E}\,[X_k]$ and hence it suffices to bound each $\mathbb{E}\,[X_k]$. Fix some $k \in [n]$ and consider the case when $|U| = k$. Recall the function $\delta(U)$ in Lemma 7. As $\delta(U)$ is only a function of size of k, we slightly abuse the notation and write $\delta(k)$ instead of $\delta(U)$ where k is the size of U.

By Lemma 7, at least half the vertices in U are $\delta(k)$-good. Hence, in each iteration, with probability at least half, we sample a $\delta(k)$-good vertex u from U. Conditioned on this event, either $\mathsf{deg}(u) < 1/\delta(k)$ which means this iteration succeeds with probability 1 or $\mathsf{deg}_U(u) \geq \delta(k) \cdot \mathsf{deg}_{V \setminus U}(u)$, and hence with probability at least $\delta(k)$ the sampled vertex v belongs to U and again this iteration succeeds. As a result, as long as U has not changed, each iteration has probability at least $\delta(k)/2$ to succeed.

By the above argument, X_k statistically dominates a Poisson distribution with parameter $\delta(k)/2$ and hence $\mathbb{E}[X_k] \leq 2/\delta(k)$. To conclude,

$$\mathbb{E}[t] = \sum_{k=1}^{n} \mathbb{E}[X_k] \leq \sum_{k=1}^{n} 2/\delta(k) = 8\beta(G) \cdot n \cdot \sum_{k=1}^{n} \frac{1}{k} \leq 8\beta(G) \cdot n \log n.$$

which finalizes the proof. ◁

Claim 9 combined with Claim 5 is already enough to prove the expected runtime bound in Theorem 4. We now prove a concentration bound for t to obtain the high probability bound. We note that it seems possible to prove the following claim by using standard concentration inequalities; however doing so requires taking care of several boundary cases for the case when $|U|$ become $o(\log n)$, and hence we instead prefer to use the following direct and more transparent proof.

▷ Claim 10. The number of iterations in Algorithm 2 is $t = O(\beta(G) \cdot n \log n)$ with high probability.

Proof. Recall from the proof of Claim 9 that $t = \sum_{k=1}^{n} X_k$ and that each X_k statistically dominates a Poisson distribution with parameter $\delta(k)/2$ (as defined in Claim 9). Define $Y_1, \ldots, Y_n$ as independent random variables where Y_k is distributed according to exponential distribution with mean $\mu_k := \frac{2}{\delta(k)}$. For any $x \in \mathbb{R}^+$,

$$\mathbf{Pr}(X_k \geq x) \leq \left(\frac{\delta(k)}{2}\right)^x = \mathbf{Pr}(Y_k \geq x).$$

As such, the random variable $Y := \sum_{k=1}^{n} Y_k$ statistically dominates the random variable t for number of iterations. Moreover by Claim 9, $\mu := \mathbb{E}[Y] = 8\beta(G) \cdot n \log n$ (the equality for Y follows directly from the proof).

In the following, we prove that with high probability Y does not deviate from its expectation by much. The proof follows the standard moment generating function idea (used for instance in the proof of Chernoff-Hoeffding bound). Let $y \in \mathbb{R}^+$. For any $s > 0$,

$$\mathbf{Pr}(Y \geq y) = \mathbf{Pr}\left(e^{sY} \geq e^{sy}\right) \leq \frac{\mathbb{E}\left[e^{sY}\right]}{e^{sy}}, \tag{2}$$

where the inequality is simply by Markov bound. Additionally, since $Y = \sum_{k=1}^{n} Y_k$ and and Y_k's are independent, we have for any $s > 0$,

$$\frac{\mathbb{E}\left[e^{sY}\right]}{e^{sy}} = \frac{\mathbb{E}\left[e^{s \cdot \sum_{k=1}^{n} Y_k}\right]}{e^{sy}} = \frac{\prod_{k=1}^{n} \mathbb{E}\left[e^{s \cdot Y_k}\right]}{e^{sy}}. \tag{3}$$

Recall that for every $i \in [n]$, $\mathbb{E}[Y_k] = \mu_k$ and Y_k is distributed according to exponential distribution. Thus, for any $s < 1/\mu_k$,

$$\mathbb{E}\left[e^{sY_k}\right] = \int_{y=0}^{\infty} e^{sy} \cdot \mathbf{Pr}(Y_k = y)\, dy = \frac{1}{\mu_k} \int_{y=0}^{\infty} e^{sy} \cdot e^{-y/\mu_k} dy = \frac{1}{1 - s\mu_k}. \tag{4}$$

Recall that $\mu_k = 2/\delta(k)$ for every $k \in [n]$ and $\delta(1) < \delta(2) < \ldots < \delta(n)$ by definition. Pick $s^* = 1/2\mu_1$ and so $s^* < 1/\mu_k$ for all $k \in [n]$. By plugging in the bounds in Eq (4) for $s = s^*$

into Eq (3), we have,

$$\frac{\mathbb{E}\left[e^{s^*Y}\right]}{e^{s^*y}} = \frac{\prod_{k=1}^{n}\mathbb{E}\left[e^{s^*\cdot Y_k}\right]}{e^{s^*y}} = \prod_{k=1}^{n}\frac{e^{-s^*y}}{1-s^*\mu_k} \leq e^{-s^*y}\cdot\exp\left(2\sum_{k=1}^{n}s^*\mu_k\right)$$

(as $1-x \geq e^{-2x}$ for $x \in (0,1/2]$)

$$= \exp\left(-s^*y + 2s^*\mu\right) = \exp\left(-y/2\mu_1 + \mu/\mu_1\right).$$

We now plug in this bound into Eq (2) with the choice of $y = 4\mu$ to obtain that,

$$\mathbf{Pr}\left(Y \geq 4\mu\right) \leq \exp\left(-4\mu/2\mu_1 + \mu/\mu_1\right) = \exp\left(-\mu/\mu_1\right) = \exp\left(-\log n\right) = 1/n,$$

where we used the fact that $\mu/\mu_1 \geq \log n$. This means that with high probability, Y is only 4 times larger than its expectation, finalizing the proof. ◁

The high probability bound in Theorem 4 now follows from Claim 5 and Claim 10, concluding the whole proof of this theorem.

Unknown $\beta(G)$

We next show that our algorithm can be easily adjusted to the case when $\beta(G)$ is unknown. The idea is simply to "guess" $\beta(G)$ in powers of two, starting from $\beta = 2$ and ending at $\beta = n$, and each time to (sequentially) run Algorithm 2 under the assumption that $\beta(G) = \beta$. For each choice of β, we shall only run the algorithm for at most $t = O(n\log n\cdot\beta(G))$ iterations (where the constant hiding in the O-notation should be sufficiently large, in accordance with that in the proof of Claim 10) and if at the end of a run the set U in the algorithm has not become empty, we start a new run from scratch with the next (doubled) value of β. (For $\beta = n$, we do not terminate the algorithm prematurely and instead run it until U is empty.)

By Theorem 4, for the *first* choice of β for which $\beta \geq \beta(G)$, the algorithm must terminate with high probability within $O(n\log n\cdot\beta(G))$ time (as $\beta \leq 2\beta(G)$ also). Moreover, the runtime of every previous run is bounded *deterministically* by $O(n\log n\cdot\beta(G))$. Consequently, the total runtime is

$$O(n\log n)\cdot\sum_{\{2^i\,|\,2^i\leq 2\beta(G)\}}2^i = O(n\log n\cdot\beta(G)),$$

where this bound holds with high probability. In this way we get an algorithm that uses no prior knowledge of $\beta(G)$ and achieves the same asymptotic performance as Algorithm 2.

5 A Lower Bound for Deterministic Maximal Matching

We prove that randomization is necessary to obtain a sublinear time algorithm for MM even on graphs with bounded neighborhood independence.

▶ **Theorem 11.** *Any deterministic algorithm that finds a maximal matching in every given graph G with neighborhood independence $\beta(G) = 2$ (known to the algorithm) presented in the adjacency array representation requires $\Omega(n^2)$ time.*

For every integer $n = 8k$ for $k \in \mathbb{N}$, we define $\mathcal{G}_n$ as the family of all graphs obtained by removing a perfect matching from a clique K_n on n vertices. For a graph G in $\mathcal{G}_n$ we refer to the removed perfect matching of size $5k$ as the *non-edge matching* of G and denote it by $\overline{M}(G)$. Clearly, every graph in $\mathcal{G}_n$ has neighborhood independence $\beta(G) = 2$. Moreover, any MM in G can have at most 2 unmatched vertices.

Let $\mathcal{A}$ be a deterministic algorithm for computing an MM on every graph in $\mathcal{G}_n$. We prove Theorem 11 by analyzing a game between $\mathcal{A}$ and an adaptive adversary that answers the probes of $\mathcal{A}$ to the adjacency array of the graph. In particular, whenever $\mathcal{A}$ probes a *new* entry of the adjacency array for some vertex $v \in V$, we can think of $\mathcal{A}$ making a *query* $Q(v)$ to the adversary, and the adversary outputs a vertex v that had not been so far revealed in the neighborhood of u (as degree of all vertices in G is exactly $n-1$, $\mathcal{A}$ knows the degree of all vertices and we assume it never queries a vertex u more than $n-1$ times).

We now show that there is a strategy for the adversary to answer the queries of $\mathcal{A}$ in a way that ensures $\mathcal{A}$ needs to make $\Omega(n^2)$ queries before it can output an MM of the graph.

Adversary's Strategy. The adversary picks an arbitrary set of $2k$ vertices D referred to as *dummy* vertices. We refer to remaining vertices as *core* vertices and denote them by $C := V \setminus D$. The adversary also fixes a non-edge matching of size k between vertices in D, denoted by $\overline{M}_D$. The non-edge matching of G consists of $\overline{M}_D$ and a non-edge matching of size $4k$ between vertices in C, denoted by $\overline{M}_C$, which unlike $\overline{M}_D$ is constructed adaptively by the adversary. We assume $\mathcal{A}$ knows the partitioning of V into D and C, as well as the non-edge matching $\overline{M}_D$. Hence, the only missing information to $\mathcal{A}$ is identity of $\overline{M}_C$.

To answer a query $Q(u)$ for a dummy vertex $u \in D$, the adversary simply returns any arbitrary vertex in V (not returned so far as an answer to $Q(u)$) except for the pair of u in $\overline{M}_D$ which cannot be neighbor to u. To answer the queries $Q(w)$ for core vertices $w \in C$, the adversary maintains a partitioning of C into C_{used} and C_{free}. Initially all core vertices belong to C_{free} and hence C_{used} is empty. Throughout we only move vertices from C_{free} to C_{used}. The adversary also maintains a counter for every vertex in C_{free} on how many times they have been queried so far. Whenever a vertex $w \in C_{\mathsf{free}}$ is queried, as long as this vertex has been queried at most k times, the adversary returns an arbitrary dummy vertex u from D as the answer to $Q(w)$ (which is possible because of size of D is k). Once a vertex $w \in C_{\mathsf{free}}$ is queried for its $(k+1)$-th time, we pick another vertex w' from C_{free} also, add the pair (w, w') to the non-edge matching $\overline{M}_C$ and move both w and w' to C_{used}, and then answer $Q(w)$ for the case $w \in C_{\mathsf{used}}$ as described below.

Recall that for any vertex $w \in C_{\mathsf{used}}$, by construction, there is another fixed vertex w' in C_{used} (joined at the same time with w) where $(w, w') \in \overline{M}_C$. For any query for $w \in C_{\mathsf{used}}$, the adversary answers $Q(w)$ by returning an arbitrary vertex from $C \setminus \{w'\}$. This concludes the description of the strategy of the adversary.

We have the following basic claim regarding the correctness of the adversary's strategy.

▷ Claim 12. The answers returned by the adversary for any sequence of queries are always consistent with at least one graph G in $\mathcal{G}_n$.

Proof. We can append the current sequence of queries by the sequence that ensures all vertices are queried $n-1$ times. Thus, the adversary would eventually construct the whole non-edge matching $\overline{M}_C$ also. There exists a unique graph G in $\mathcal{G}_n$ where $\overline{M}(G) = \overline{M}_D \cup \overline{M}_C$, hence proving the claim (note that before appending the sequence, there can be more than one graph consistent with the original sequence). ◁

We now prove the following lemma which is the key step in the proof of Theorem 11.

▶ **Lemma 13.** *Suppose $\mathcal{A}$ makes at most k^2 queries to the adversary and outputs a matching M using only these queries. Then, there exists some graph G in $\mathcal{G}_n$ where G is consistent with the answers returned by the adversary to $\mathcal{A}$ and $\overline{M}(G) \cap M \neq \emptyset$.*

Proof. Since $\mathcal{A}$ makes at most k^2 queries, there can only be $2k$ vertices in C_{used} by the time the algorithm finishes its queries (as each pair of vertices in C_{used} consumed k queries). Consider the maximal matching M. There are at most $2k$ edges of M that are incident on C_{used} and k more edges incident on D. This implies that at most $3k$ vertices in C_{free} are matched to vertices outside C_{free}. As $|C_{\mathsf{free}}| \geq 5k$, it means that there are at least $2k$ vertices in C_{free} that are not matched by M to vertices outside C_{free}. However, as M is maximal, and since G is in $\mathcal{G}_n$, there should be at least $k-1$ edges in M between these vertices in C_{free}. As the adversary has not committed to the non-edge matching $\overline{M}_C$ entirely at this point and this non-edge matching pairs vertices in C_{free} to each other, the adversary can simply let $\overline{M}_C$ contains M and still obtain a graph G in $\mathcal{G}_n$. For this graph, $\overline{M}(G) \cap M \neq \emptyset$, finalizing the proof. ◀

Theorem 11 now follows immediately from Lemma 13, as it state that unless the algorithm makes $\Omega(n^2)$ queries, there always exists at least one graph in $\mathcal{G}_n$ for which the output matching of the algorithm is not feasible. As graphs in $\mathcal{G}_n$ all have $\beta(G) = 2$, we obtain the final result.

References

1 Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of algorithms*, 7(4):567–583, 1986.

2 Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient local computation algorithms. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1132–1139, 2012.

3 Jonathan Aronson, Martin E. Dyer, Alan M. Frieze, and Stephen Suen. Randomized Greedy Matching II. *Random Struct. Algorithms*, 6(1):55–74, 1995.

4 Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear Algorithms for (Δ + 1) Vertex Coloring. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019.*, pages 767–786, 2019. arXiv:1807.08886.

5 Sepehr Assadi, Krzysztof Onak, Baruch Schieber, and Shay Solomon. Fully dynamic maximal independent set with sublinear update time. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 815–826, 2018.

6 Sepehr Assadi, Krzysztof Onak, Baruch Schieber, and Shay Solomon. Fully Dynamic Maximal Independent Set with Sublinear in n Update Time. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1919–1936, 2019.

7 Leonid Barenboim and Michael Elkin. Distributed deterministic edge coloring using bounded neighborhood independence. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011, San Jose, CA, USA, June 6-8, 2011*, pages 129–138, 2011.

8 Leonid Barenboim and Michael Elkin. *Distributed Graph Coloring: Fundamentals and Recent Developments*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2013.

9 Leonid Barenboim, Michael Elkin, and Fabian Kuhn. Distributed (Delta+1)-Coloring in Linear (in Delta) Time. *SIAM J. Comput.*, 43(1):72–95, 2014.

10 Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The Locality of Distributed Symmetry Breaking. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 321–330, 2012.

11 Surender Baswana, Manoj Gupta, and Sandeep Sen. Fully Dynamic Maximal Matching in O(log n) Update Time. *SIAM J. Comput.*, 44(1):88–113, 2015.

12 Soheil Behnezhad, MohammadTaghi Hajiaghayi, and David G. Harris. Exponentially Faster Massively Parallel Maximal Matching. *CoRR*, abs/1901.03744, 2019.

13 Yair Caro. New results on the independence number. Technical report, Technical Report, Tel-Aviv University, 1979.

14 Maria Chudnovsky and Paul D. Seymour. The structure of claw-free graphs. In *Surveys in Combinatorics, 2005 [invited lectures from the Twentieth British Combinatorial Conference, Durham, UK, July 2005]*, pages 153–171, 2005.

15 Graham Cormode, Jacques Dark, and Christian Konrad. Independent Sets in Vertex-Arrival Streams. *CoRR*, abs/1807.08331, 2018.

16 Martin E. Dyer and Alan M. Frieze. Randomized Greedy Matching. *Random Struct. Algorithms*, 2(1):29–46, 1991.

17 Ralph Faudree, Evelyne Flandrin, and Zdeněk Ryjáček. Claw-free graphs?a survey. *Discrete Mathematics*, 164(1-3):87–147, 1997.

18 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005. doi:10.1016/j.tcs.2005.09.013.

19 Manuela Fischer, Mohsen Ghaffari, and Fabian Kuhn. Deterministic Distributed Edge-Coloring via Hypergraph Maximal Matching. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 180–191, 2017.

20 Pierre Fraigniaud, Marc Heinrich, and Adrian Kosowski. Local Conflict Coloring. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 625–634, 2016.

21 Mohsen Ghaffari. An Improved Distributed Algorithm for Maximal Independent Set. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 270–277, 2016.

22 Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrovic, and Ronitt Rubinfeld. Improved Massively Parallel Computation Algorithms for MIS, Matching, and Vertex Cover. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 129–138, 2018.

23 Mohsen Ghaffari and Jara Uitto. Sparsifying Distributed Algorithms with Ramifications in Massively Parallel Computation and Centralized Local Computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1636–1653, 2019.

24 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. Perfect Matchings in\˜ O (n^{1.5}) Time in Regular Bipartite Graphs. *arXiv preprint*, 2009. arXiv:0902.1617.

25 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. Perfect matchings via uniform sampling in regular bipartite graphs. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 11–17, 2009.

26 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. Perfect matchings in o(n log n) time in regular bipartite graphs. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 39–46, 2010.

27 Bjarni V. Halldórsson, Magnús M. Halldórsson, Elena Losievskaja, and Mario Szegedy. Streaming Algorithms for Independent Sets in Sparse Hypergraphs. *Algorithmica*, 76(2):490–501, 2016.

28 Magnús M. Halldórsson. Wireless Scheduling with Power Control. In *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, pages 361–372, 2009.

29 Magnús M. Halldórsson and Christian Konrad. Distributed Large Independent Sets in One Round on Bounded-Independence Graphs. In *Distributed Computing - 29th International Symposium, DISC 2015, Tokyo, Japan, October 7-9, 2015, Proceedings*, pages 559–572, 2015.

30 Magnús M. Halldórsson, Guy Kortsarz, and Hadas Shachnai. Sum Coloring Interval and k-Claw Free Graphs with Application to Scheduling Dependent Jobs. *Algorithmica*, 37(3):187–209, 2003.

31 Michal Hanckowiak, Michal Karonski, and Alessandro Panconesi. On the Distributed Complexity of Computing Maximal Matchings. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 25-27 January 1998, San Francisco, California, USA.*, pages 219–225, 1998.

32 Amos Israeli and Alon Itai. A Fast and Simple Randomized Parallel Algorithm for Maximal Matching. *Inf. Process. Lett.*, 22(2):77–80, 1986.

33 Richard M. Karp and Avi Wigderson. A Fast Parallel Algorithm for the Maximal Independent Set Problem. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1984, Washington, DC, USA*, pages 266–272, 1984.

34 Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. What cannot be computed locally! In *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004, St. John's, Newfoundland, Canada, July 25-28, 2004*, pages 300–309, 2004.

35 Fabian Kuhn, Tim Nieberg, Thomas Moscibroda, and Roger Wattenhofer. Local approximation schemes for ad hoc and sensor networks. In *Proceedings of the DIALM-POMC Joint Workshop on Foundations of Mobile Computing, Cologne, Germany, September 2, 2005*, pages 97–103, 2005.

36 Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Ad hoc networks beyond unit disk graphs. *Wireless Networks*, 14(5):715–729, 2008.

37 Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, and Sergei Vassilvitskii. Filtering: a method for solving graph problems in MapReduce. In *SPAA 2011: Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures, San Jose, CA, USA, June 4-6, 2011 (Co-located with FCRC 2011)*, pages 85–94, 2011. doi:10.1145/1989493.1989505.

38 Christoph Lenzen and Roger Wattenhofer. MIS on trees. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011, San Jose, CA, USA, June 6-8, 2011*, pages 41–48, 2011.

39 Reut Levi, Ronitt Rubinfeld, and Anak Yodpinyanee. Local Computation Algorithms for Graphs of Non-constant Degrees. *Algorithmica*, 77(4):971–994, 2017.

40 Nathan Linial. Locality in Distributed Graph Algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992.

41 Michael Luby. A Simple Parallel Algorithm for the Maximal Independent Set Problem. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 1–10, 1985.

42 Zevi Miller and Dan Pritikin. On randomized greedy matchings. *Random Struct. Algorithms*, 10(3):353–383, 1997.

43 Ofer Neiman and Shay Solomon. Simple deterministic algorithms for fully dynamic maximal matching. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 745–754, 2013.

44 Krzysztof Onak, Dana Ron, Michal Rosen, and Ronitt Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1123–1131, 2012.

45 Krzysztof Onak, Baruch Schieber, Shay Solomon, and Nicole Wein. Fully Dynamic MIS in Uniformly Sparse Graphs. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 92:1–92:14, 2018.

46 Alessandro Panconesi and Aravind Srinivasan. On the Complexity of Distributed Network Decomposition. *J. Algorithms*, 20(2):356–374, 1996.

47 Matthias Poloczek and Mario Szegedy. Randomized Greedy Algorithms for the Maximum Matching Problem with New Analysis. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 708–717, 2012.

48 Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast Local Computation Algorithms. In *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 7-9, 2011. Proceedings*, pages 223–238, 2011.

49 Johannes Schneider and Roger Wattenhofer. A log-star distributed maximal independent set algorithm for growth-bounded graphs. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Principles of Distributed Computing, PODC 2008, Toronto, Canada, August 18-21, 2008*, pages 35–44, 2008.

50 Shay Solomon. Fully Dynamic Maximal Matching in Constant Update Time. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 325–334, 2016.

51 VK Wei. A lower bound on the stability number of a simple graph. Technical report, Bell Laboratories Technical Memorandum 81-11217-9, Murray Hill, NJ, 1981.

Robust Communication-Optimal Distributed Clustering Algorithms

Pranjal Awasthi
Rutgers University, Piscataway, NJ, USA
pranjal.awasthi@rutgers.edu

Ainesh Bakshi
Carnegie Mellon University, Pittsburgh, PA, USA
abakshi@cs.cmu.edu

Maria-Florina Balcan
Carnegie Mellon University, Pittsburgh, PA, USA
ninamf@cs.cmu.edu

Colin White
Carnegie Mellon University, Pittsburgh, PA, USA
crwhite@cs.cmu.edu

David P. Woodruff
Carnegie Mellon University, Pittsburgh, PA, USA
dwoodruf@cs.cmu.edu

Abstract

In this work, we study the k-median and k-means clustering problems when the data is distributed across many servers and can contain outliers. While there has been a lot of work on these problems for worst-case instances, we focus on gaining a finer understanding through the lens of beyond worst-case analysis. Our main motivation is the following: for many applications such as clustering proteins by function or clustering communities in a social network, there is some unknown target clustering, and the hope is that running a k-median or k-means algorithm will produce clusterings which are close to matching the target clustering. Worst-case results can guarantee constant factor approximations to the optimal k-median or k-means objective value, but not closeness to the target clustering.

Our first result is a distributed algorithm which returns a near-optimal clustering assuming a natural notion of stability, namely, *approximation stability* [12], even when a constant fraction of the data are outliers. The communication complexity is $\tilde{O}(sk+z)$ where s is the number of machines, k is the number of clusters, and z is the number of outliers. Next, we show this amount of communication cannot be improved even in the setting when the input satisfies various non-worst-case assumptions. We give a matching $\Omega(sk + z)$ lower bound on the communication required both for approximating the optimal k-means or k-median cost up to any constant, and for returning a clustering that is close to the target clustering in Hamming distance. These lower bounds hold even when the data satisfies approximation stability or other common notions of stability, and the cluster sizes are balanced. Therefore, $\Omega(sk + z)$ is a communication bottleneck, even for real-world instances.

2012 ACM Subject Classification Theory of computation → Unsupervised learning and clustering

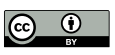
Keywords and phrases robust distributed clustering, communication complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.18

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1703.00830`.

Acknowledgements This work was supported in part by NSF grants CCF-1422910, CCF-1535967, IIS-1618714, an Office of Naval Research (ONR) grant N00014-18-1-2562, an Amazon Research Award, a Microsoft Research Faculty Fellowship, and a National Defense Science & Engineering Graduate (NDSEG) fellowship. Part of this work was done while Ainesh Bakshi and David Woodruff were visiting the Simons Institute for the Theory of Computing.



1 Introduction

Clustering is a fundamental problem in machine learning with applications in many areas including computer vision, text analysis, bioinformatics, and so on. The underlying goal is to group a given set of points to maximize similarity inside a group and dissimilarity among groups. A common approach to clustering is to set up an objective function and then approximately find the optimal solution according to the objective. Common examples of these objective functions include k-median and k-means, in which the goal is to find k centers to minimize the sum of the distances (or sum of the squared distances) from each point to its closest center. Motivated by real-world constraints, further variants of clustering have been studied. For instance, in k-clustering with outliers, the goal is to find the best clustering (according to one of the above objectives) after removing a specified number of data points, which is useful for noisy data. Finding approximation algorithms to different clustering objectives and variants has attracted significant attention in the computer science community [7, 23, 24, 25, 28, 31, 41].

As datasets become larger, sequential algorithms designed to run on a single machine are no longer feasible for real-world applications. Additionally, in many cases data is naturally spread out among multiple locations. For example, hospitals may keep records of their patients locally, but may want to cluster the entire spread of patients across all hospitals in order to do better data analysis and inference. Therefore, distributed clustering algorithms have gained popularity in recent years [18, 20, 42, 32, 40, 29, 27]. In the distributed setting, it is assumed that the data is partitioned arbitrarily across s machines, and the goal is to find a clustering which approximates the optimal solution over the entire dataset while minimizing communication among machines. Recent work in the theoretical machine learning community establishes guarantees on the clusterings produced in distributed settings for certain problems [18, 20, 42]. For example, [42] provides distributed algorithms for k-center and k-center with outliers, and [20] introduces distributed algorithms for capacitated k-clustering under any ℓ_p objective. Along similar lines, the recent work of [32] provides constant-factor approximation algorithms for k-median and k-means with z outliers in the distributed setting. The work of Guha et al. also provides the best known communication complexity bound of $O(sk + z)$ where s is the number of machines, and z is the number of outliers.

Although the above results provide a constant-factor approximation to k-median or k-means objectives, many real-world applications desire a clustering that is close to a 'ground truth' clustering in terms of the structure, i.e., the way the points are clustered rather than in terms of cost. For example, for applications such as clustering proteins by function or clustering communities in a social network, there is some unknown target clustering, and the hope is that running a k-median or k-means algorithm will produce clusterings which are close to matching the target clustering. While in general having a constant factor approximation provides no guarantees on the closeness to the optimal clustering, a series of recent works has established that this is possible if the data has certain structural properties [10, 11, 12, 16, 21, 30, 39, 46]. For example, the $(1 + \alpha, \epsilon)$-*approximation stability* condition defined by [12] states that any $(1 + \alpha)$-approximation to the clustering objective is ϵ-close to the target clustering. For such instances, it is indeed possible to output a clustering close to the ground truth in polynomial time, even for values of α such that computing a $(1+\alpha)$-approximation is NP-hard. We follow this line of research and ask whether distributed clustering is possible for non worst-case instances, in the presence of outliers.

1.1 Our contributions

A distributed clustering instance consists of a set of n points in a metric space partitioned arbitrarily across s machines. The problem is to optimize the k-median/k-means objective while minimizing the amount of communication across the machines. We consider algorithms that approximate the optimal cost as well as computing a clustering close to the target clustering in Hamming distance. Our contributions are as follows:

1. In Section 3, we give a centralized clustering algorithm whose output is ϵ-close to the target clustering, in the presence of z outliers, assuming the data satisfies $(1+\alpha, \epsilon)$-approximation stability and assuming a lower bound on the size of the optimal clusters. To the best of our knowledge, this is the first polynomial time algorithm for clustering approximation stable instances in the presence of outliers. Our results hold for arbitrary values of z, including when a constant fraction of the points are outliers, as long as there is a lower bound on the minimum cluster size.
2. We then give a distributed algorithm whose output is close to the target clustering, assuming the data satisfies $(1+\alpha, \epsilon)$-approximation stability. The communication complexity is $\widetilde{O}(sk)$, where s is the number of servers and k is the number of clusters. We also extend this to handle z outliers, with a communication complexity $\widetilde{O}(sk+z)$. This matches the worst-case communication of [32], while outputting a near-optimal clustering by taking advantage of new structural guarantees specific to approximation stability with outliers.
3. While the above algorithms improve over worst-case distributed clustering algorithms in terms of quality of the returned clustering, our algorithms use the same amount of communication as the worst case protocols. In Section 4, we show that the $\Omega(sk)$ and $\Omega(sk+z)$ communication costs for clustering without and with outliers are unavoidable even if data satisfies many types of stability assumptions that have been studied in the literature. Our lower bound of $\Omega(sk+z)$ for obtaining a c-approximation (for any $c \geq 1$) holds even when the data is arbitrarily stable, e.g., $(1+\alpha, \epsilon)$-approximation stable for all $\alpha \geq 0$ and $0 \leq \epsilon < 1$.
4. We also give an $\Omega(sk+z)$ lower bound for the problem of computing a clustering whose Hamming distance is close to the optimal clustering, even when the data is approximation-stable. Finally, we prove that our above $\Omega(sk+z)$ lower bounds hold for finding a clustering close to the optimal in Hamming distance even when it is guaranteed that the optimal clusters are completely balanced, i.e., each cluster is of size $\frac{n-z}{k}$ (in addition to the guarantee that the clustering satisfies approximation stability), implying our algorithms from Section 3 are optimal. Therefore, $\Omega(sk+z)$ is a fundamental communication bottleneck, even for real-world clustering instances.

1.2 Related Work

There is a long line of work on approximation algorithms for k-median and k-means clustering [24, 36, 41], and the current best approximation ratios are 2.675 [23] and 6.357 [4], respectively. The first constant-factor approximation algorithm for k-median with z outliers was given by Chen [28], and the current best approximation ratios for k-median and k-means with outliers are $7.081+\epsilon$ and $53.002+\epsilon$, respectively, given by Krishnaswamy et al. [38]. There is also a line of work on clustering with balance constraints on the clusters [5, 3, 30]. For k-median and k-means clustering in distributed settings, the work of Balcan et al. showed a coreset construction for k-median and k-means, which leads to a clustering algorithm with $\tilde{O}(skd)$ communication, where d is the dimension, and also studied more general graph topologies for distributed computing [18]. Huang et al. [34] showed a coreset construction for

doubling metrics. Malkomes et al. showed a distributed 13- and 4- approximation algorithm for k-center with and without outliers, respectively [42]. Chen et al. studied clustering under the broadcast model of distributed computing, and also proved a communication complexity lower bound of $\Omega(sk)$ for distributed clustering [27], building on a recent lower bound for set-disjointness in the message-passing model [22]. Recently, [32] showed a distributed algorithm with $\tilde{O}(sk + z)$ communication for computing a constant-factor approximation to k-median clustering with z outliers. They also provide bicriteria approximations that remove $(1 + \epsilon)z$ outliers to get a clustering of cost $O\left(1 + \frac{1}{\epsilon}\right)$ times the cost of the optimal k-median clustering with z outliers, for any $\epsilon > 0$. Even more recently, [40] showed that there exists a bi-criteria algorithm with communication independent of z that achieves a constant approximation to the cost. In particular, their algorithm outputs $(1 + \epsilon)z$ outliers and achieves a $(24 + \epsilon)$-approximation with $O\left(\frac{sk}{\epsilon} + \frac{s \log \Delta}{\epsilon}\right)$ communication, where Δ is the aspect ratio of the metric.

In recent years, there has also been a focused effort towards understanding clustering for non worst-case models [43, 1, 21, 39]. The work of Balcan et al. defined the notion of approximation stability and showed an algorithm which utilizes the structure to output a nearly optimal clustering [12]. Approximation stability has been studied in a wide range of contexts, including clustering [15, 17, 14], the k-means++ heuristic [2], social networks [33], and computing Nash-equilibria [9]. A recent paper by Chekuri and Gupta introduces the model of clustering with outliers under perturbation resilience, a notion of stability which is related to approximation stability [26].

2 Preliminaries

Given a set V of points of size n, a distance metric d, and an integer k, let $\mathcal{C}$ denote a clustering of V, which we define as a partition of V into k subsets $C_1, \ldots, C_k$. Each cluster C_i contains a center c_i. When d is an arbitrary distance metric, we must choose the centers from the point set. If $V \subseteq \mathbb{R}^d$ and the distance metric is the Euclidean distance, then the centers can be any k points in $\mathbb{R}^d$. In fact, this distinction only changes the cost of the optimal clustering by at most a factor of 2 by the triangle inequality for any p (see, e.g., [8]).

The k-median and the k-means costs are $\sum_i \sum_{v \in C_i} d(c_i, v)$, and $\sum_i \sum_{v \in C_i} d(c_i, v)^2$ respectively. For k clustering with z outliers, the problem is to compute the minimum cost clustering over $n - z$ points, e.g., we must decide which z points to remove, and how to cluster the remaining points, to minimize the cost. We will denote the optimal k-clustering with z outliers by $\mathcal{OPT}$, and we denote the set of outliers for $\mathcal{OPT}$ by Z. We often overload notation and let $\mathcal{OPT}$ denote the objective value of the optimal clustering as well. We denote the optimal clusters as $C_1^*, \ldots, C_k^*$, with centers $c_1, \ldots, c_k$. We say that two clusterings $\mathcal{C}$ and $\mathcal{C}'$ are δ-close if they differ by only $\delta(n - z)$ points, i.e., $\min_\sigma \sum_{i=1}^k |C_i \setminus C'_{\sigma(i)}| < \delta(n - z)$. Let $C^*_{\min} = \min_{j \in [k]} |C_j^*|$, i.e., the minimum cluster size. Given a point $c \in V$, we define $V_c \subset V$ to be the closest set of $C^*_{\min}$ points to c.

We study a notion of stability called approximation stability. Intuitively, a clustering instance satisfies this assumption if all clusterings close in *value* to $\mathcal{OPT}$ are also close in terms of the clusters themselves. This is a desirable property when running an approximation algorithm, since in many applications, the k-means or k-median costs are proxies for the final goal of recovering a clustering that is close to the desired "target" clustering. Approximation stability makes this assumption explicit. This was first defined for clustering with $z = 0$ [12], however, we generalize the definition to the setting with outliers.

▶ **Definition 1** (approximation stability). *A clustering instance satisfies* $(1+\alpha, \epsilon)$-approximation stability *for k-median or k-means with z outliers if for all k-clusterings with z outliers, denoted by $\mathcal{C}$, if cost$(\mathcal{C}) \leq (1+\alpha) \cdot \mathcal{OPT}$, then $\mathcal{C}$ is ϵ-close to $\mathcal{OPT}$.*

This definition implies that all clusterings close in cost to $\mathcal{OPT}$ must have nearly the same set of outliers, because if $\mathcal{C}$ contains more than $\epsilon(n-z)$ points from Z, then $\mathcal{C}$ and $\mathcal{OPT}$ cannot be ϵ-close. This is similar to related models of stability for clustering with outliers, e.g. [26]. Note it is standard in this line of work to assume the value of α is known [12].

We will study distributed algorithms under the standard framework of the *coordinator model*. There are s servers, and a designated coordinator. Each server can send messages back and forth with the coordinator. This model is very similar to the *message-passing model*, also known as the *point-to-point* model, in which any pair of machines can send messages back and forth. In fact, the two models are equivalent up to constant factors in the communication complexity [22]. Most of our algorithms can be applied to the mapreduce framework with a constant number of rounds. For more details, see [20, 42].

For our communication lower bounds, we work in the multi-party message passing model, where there are s players, $P_1, P_2, \ldots, P_s$, who receive inputs X^1, X^2, $\ldots X^s$ respectively. They have access to private randomness as well as a common publicly shared random string R, and the objective is to communicate with a central coordinator who computes a function $f : X^1 \times X^2 \ldots \times X^s \to \{0,1\}$ on the joint inputs of the players. The communication has multiple rounds and each player is allowed to send messages to the coordinator. Note, we can simulate communication between the players by blowing up the rounds by a factor of 2. Given X^i as an input to player i, let Π be the random variable that denotes the transcript between the players and the referee when they execute a protocol Π. For $i \in [s]$, let Π_i denote the messages sent by P_i to the referee.

A protocol Π is called a δ-error protocol for function f if there exists a function Π_{out} such that for every input, $Pr\left[\Pi_{out} = f(X^1, X^2, \ldots X^s)\right] \geq 1-\delta$. The communication cost of a protocol, denoted by $|\Pi|$, is the maximum length of Π over all possible inputs and random coin flips of all the s players and the referee. The randomized communication complexity of a function f, $R_\delta(f)$, is the communication cost of the best δ-error protocol for computing f. For our lower bounds, we also consider that the data satisfies a very strong, general notion of stability which we call c-separation.

▶ **Definition 2** (separation). *Given $c \geq 1$ and a clustering objective, a clustering instance satisfies* c-separation *if $c \cdot \max_i \max_{u,v \in C_i^*} d(u,v) < \min_i \min_{u' \in C_i^*, v' \notin C_i^*} d(u',v')$.*

Intuitively, this definition implies the maximum distance between any two points in one cluster is a factor c smaller than the minimum distance across clusters. This assumption has been used in several papers (for clustering with no outliers) to show guarantees for various algorithms [13, 44, 37]. We note that this notion of stability captures a wide class of previously studied notions including perturbation resilience [21, 10, 16, 6] and approximation stability.

▶ **Definition 3** (perturbation resilience). *For $\beta > 0$, a clustering instance (V,d) satisfies $1+\alpha$-perturbation resilience for the k-means objective, if for any function $d' : V \times V \to \mathbb{R}_{\geq 0}$, such that for all $p, q \in V$, $d(p,q) \leq d'(p,q) \leq (1+\beta)d(p,q)$, and the optimal clustering under d' is unique and equal to the optimal clustering under d, for the k-means objective.*

We note we can replace the objective with any center based objective such as k-median or k-center. Next, we show that *separation* implies *approximation stability* and *perturbation resilience*. We defer the proof to the Appendix.

▶ **Lemma 4.** *Given $\alpha, \epsilon > 0$, and a clustering objective such as k-median, let (V, d) be a clustering instance which satisfies c-separation, for $c > (1+\alpha)n$, where $n = |V|$. Then (V, d) satisfies $(1+\alpha, \epsilon)$-approximation stability and $(1+\alpha)$-perturbation resilience.*

3 Approximation Stability with Outliers

In this section, we give a centralized algorithm for clustering with z outliers under approximation stability, and then extend it to a distributed algorithm for the same problem. To the best of our knowledge, this is the first result for clustering with outliers under approximation stability, as well as the first distributed algorithm for clustering under approximation stability even without outliers. We defer the details to the Appendix. Our algorithm can handle any fraction of outliers, even when the set of outliers makes up a constant fraction of the input points. For simplicity, we focus on k-median.

▶ **Theorem 5** (Centralized Clustering)**.** *Algorithm 1 runs in poly$\left(n, \left(\frac{\alpha}{\epsilon}\left(k + \frac{1}{\alpha}\right)\right)^{\frac{1}{\alpha}}\right)$ time and outputs a clustering that is ϵ-close to $\mathcal{OPT}$ for k-median with z outliers under $(1+\alpha, \epsilon)$-approximation stability, assuming for all i, $|C_i^*| \geq 2\left(1 + \frac{5}{\alpha}\right)\epsilon(n-z)$.*

Note that the runtime is at most poly$\left(n^{\frac{1}{\alpha}}\right)$, and if $\frac{\alpha}{\epsilon} \in \Theta(k)$, the runtime is poly$\left(n, k^{\frac{1}{\alpha}}\right)$. The algorithm has two high-level steps. First, we use standard techniques from approximation stability without outliers to find a list of clusters $\mathcal{X}$, which contains clusters from the optimal solution (with $\leq \left(1 + \frac{1}{\alpha}\right)\epsilon(n-z)$ mistakes), and clusters made up mostly of outlier points. We show how all but $1/\alpha$ of the outlier clusters must have high cost if their size were to be extended to the minimum optimal cluster size, and can thus be removed from our list $\mathcal{X}$. Finally, we use brute force enumeration to remove the final $\frac{1}{\alpha}$ outlier clusters, and after another cluster purifying step, we are left with a k clustering which $(1+\alpha)$-approximates the cost and thus is guaranteed to be ϵ-close to optimal.

We begin by outlining the key properties of $(1+\alpha, \epsilon)$-approximation stability. Let w_{avg} denote the average distance from each point to its optimal center, so $w_{avg} \cdot (n-z) = \mathcal{OPT}$. The following lemma is the first of its kind for clustering with outliers and establishes two key properties for approximation stable instances. Intuitively, the first property bounds the number of points that are far away from their optimal center, and follows from Markov's inequality. The second property bounds the number of points that are either closer on average to the center of a non-optimal cluster that the optimal one or are outliers that are close to some optimal center as compared to a point belonging to that cluster.

▶ **Lemma 6.** *Given a $(1+\alpha, \epsilon)$-approximation stable clustering instance (V, d) for k-median such that for all i, $|C_i^*| > 2\epsilon(n-z)$, then **Property 1:** For all $y > 0$, there exist at most $\frac{y\epsilon}{\alpha}(n-z)$ points, v, such that $d(v, c_v) \geq \frac{\alpha w_{avg}}{y\epsilon}$. **Property 2:** There are fewer than $\epsilon(n-z)$ total points with one of the following two properties: the point v is in an optimal cluster C_i^*, and there exists $j \neq i$ such that $d(v, c_j) - d(v, c_i) \leq \frac{\alpha w_{avg}}{\epsilon}$, or, the point v is in Z, and there exists i and $v' \in C_i^*$ such that $d(v, c_i) \leq d(v', c_i) + \frac{\alpha w_{avg}}{\epsilon}$ (recall that Z denotes the set of outliers from the optimal clustering).*

We define a point as *bad* if it falls into the bad case of either Property 1 (with $y = 5$) or Property 2, and we denote the set of bad points by B. Otherwise, a point is *good*. From Properties 1 and 2, $|B| \leq \left(1 + \frac{5}{\alpha}\right)\epsilon(n-z)$. For each i, let G_i denote the good points from the optimal cluster C_i^*. We consider the graph $G' = (V, E')$ called the *neighborhood graph*, constructed by adding an edge (u, v) iff there are at least $|B| + 2$ points w such that $d(u, w), d(v, w) \leq \tau = \frac{2w_{avg}}{5}$. Under approximation stability, the graph G' has the following

structure: there is an edge between all pairs of good points from C_i^* and there is no edge between any pair of good points belonging to distinct clusters, C_i^*, C_j^*. Further, these points do not have any common neighbors. Since the set of good points in each cluster, denoted by G_i, form cliques of size $> |B|$ and are far away from one another, and there are $\leq |B|$ bad points, it follows that each G_i is in a unique connected component C_i' of G'.

In the setting without outliers, the list of connected components of size greater than $\left(1+\frac{5}{\alpha}\right)\epsilon n$ is exactly $\{C_1', \ldots, C_k'\}$. However, in the setting with outliers, we can only return a set $\mathcal{X}$ which includes $\{C_1', \ldots, C_k'\}$ but also may include many other outlier clusters which are hard to distinguish from the optimal clusters. Although approximation stability tells us that any set Z' of outliers must have a much higher cost than any optimal cluster C_i^* (since we can arrive at a contradiction by replacing the cluster C_i^* with the cluster Z'), this is not true when the size of Z' is even slightly smaller than C_i^*. Since the good clusters returned are only $O\left(\frac{\epsilon}{\alpha}\right)$-close to optimal, many good clusters may be smaller than outlier clusters, and so a key challenge is to distinguish outlier clusters Z' from good clusters C_i'.

To accomplish this task, we compute the minimum cost of each cluster, pretending that its size is at least $C^*_{\min}$ (the size of the minimum optimal cluster, which we can guess in polynomial time). In our key structural lemma (Lemma 7), we show that nearly all outlier components will have large cost. Given a set of points Q, we define $\text{cost}_{\min}(Q)$ to be the minimum cost of Q if it were extended to $C^*_{\min}$ points. Note, $\text{cost}_{\min}(Q)$ can be computed in polynomial time by iterating over all points $c \in Q$, for each such point constructing V_c by adding the the $C^*_{\min} - |Q|$ points closest to c, computing the resulting cost, and taking the minimum over all such costs.

Algorithm 1 k-median with z-outliers under Approximation Stability.

`Input`: Clustering instance (V, d), cost w_{avg}, value C^*_{min}, integer $x > 0$.

1. Create the neighborhood graph on V with parameters $\tau = \frac{2w_{avg}}{5\epsilon}$ and $b = C^*_{min} - (1 + \frac{5}{\alpha})\epsilon(n-z)$ as follows: for each $u, v \in V$, add an edge (u, v) iff there exist $\geq b$ points $w \in V$ such that $d(u, w), d(w, v) \leq \tau$. Denote the connected components by $\mathcal{X} = \{Q_1, \ldots, Q_d\}$.
2. For each Q_i, compute $\text{cost}_{\min}(Q_i) = \min_{c \in Q_i} \min_{V_c} \sum_{v \in V_c} d(c, v)$, where V_c must satisfy $|V_c| \geq C^*_{min}$ and $Q_i \subseteq V_c$. Create a new set $\mathcal{X}' = \{Q_i \mid \text{cost}_{min}(Q_i) < \left(3 + \frac{2\alpha}{5}\right)\frac{1}{x} \cdot \mathcal{OPT}\}$. .
3. For all $0 \leq t \leq x$, for each size t subset $\mathcal{X}_t' \subseteq \mathcal{X}'$ and size $(k - |\mathcal{X}'| - t)$ subset $\mathcal{X}_t \subseteq (\mathcal{X} \setminus \mathcal{X}')$,
   a. Create a new clustering $\mathcal{C} = \mathcal{X}' \cup \mathcal{X}_t \setminus \mathcal{X}_t'$.
   b. For each point $v \in V$, define $I(v)$ as the index of the cluster in $\mathcal{C}$ with minimum median distance to v, e.g., $I(v) = \text{argmin}_i\left(d_{\text{med}}(v, Q_i)\right)$ where $d_{\text{med}}(v, Q_i)$ denotes the median distance from v to Q_i.
   c. Let $V' \subseteq V$ denote the $n - z$ points with the smallest values of $d(v, c_{I(v)})$. For all i, set $Q_i' = \{v \in V' \mid I(v) = i\}$.
   d. If $\sum_i \text{cost}(Q_i') \leq (1+\alpha)\mathcal{OPT}$, return $\{Q_1, \ldots, Q_k\}$.

▶ **Lemma 7.** *Given an instance of k-median clustering with z outliers such that each optimal cluster $|C_i^*| > 2\left(1+\frac{5}{\alpha}\right)\epsilon(n-z)$, for any $x \in \mathbb{N}$, the instance satisfies $(1+\alpha, \epsilon)$-approximation stability for $\alpha > \frac{35}{5x-4}$, and there are at most x disjoint sets of outliers Z' such that $|Z'| > \min_i |C_i^*| - \left(1+\frac{5}{\alpha}\right)\epsilon(n-z)$ and $\text{cost}_{\min}(Z') \leq \left(3+\frac{2\alpha}{5}\right)\frac{1}{x}\mathcal{OPT}$.*

The key ideas behind the proof are as follows. If there are two sets of outliers Z_1 and Z_2 both with fewer than $C^*_{\min}$ points, then we can obtain a contradiction by taking into account both sets of outliers. Set $1 \leq z_1, z_2 \leq \left(1+\frac{5}{\alpha}\right)\epsilon(n-z)$ such that $|Z_1| = C^*_{\min} - z_1$ and

$|Z_2| = C^*_{\min} - z_2$, and assume without loss of generality that $z_1 < z_2$. We design a different clustering $\mathcal{C}'$ by first replacing the minimum-sized cluster in the optimal clustering with Z_1. The cost of the points in Z_2 is low by assumption. However, we have now potentially assigned more than z points to be outliers by an additive z_1 amount. Hence, in order to create a valid clustering that is far from $\mathcal{OPT}$ we need to add back at least z_1 more outlier points. We do this by choosing z_1 outlier points from Z_2 that are closest to an optimal center in $\mathcal{OPT}$. To bound the additional cost incurred, we use the fact that Z_2 must be close to at least z_2 points from $V \setminus Z$, by the assumption that $\text{cost}_{\min}(Z_2)$ is low, and use these points to bound the distance from centers in $\mathcal{OPT}$ to the z_1 points that were added back. In the full proof, we extend this idea to x sets $Z_1, \ldots, Z_x$ to achieve a tradeoff between x and α.

From Lemma 7, we show a threshold of $\text{cost}_{\min}$ for the components of $\mathcal{X}$, such that all but x optimal clusters are below the cost threshold, and all but x outlier clusters are above the cost threshold. Then we can brute force over all ways of excluding x low-cost sets and including x high-cost sets, and we will be guaranteed that one combination contains a clustering which is $O\left(\frac{\epsilon}{\alpha}\right)$-close to the optimal. However, we still need to recognize the right clustering when we see it. To do this, we show that after performing one more cluster purifying step which is inspired by arguments in [12] - reassigning all points to the component with the minimum median distance - we will reduce our error to $\epsilon(n-z)$ in Hamming distance and we show how to bound the total cost of these mistakes by $\frac{4\alpha}{5}\mathcal{OPT}$. Therefore, during brute force enumeration, we return immediately when we find a clustering with cost at most $(1+\alpha)\mathcal{OPT}$ (and thus must be ϵ-close to $\mathcal{OPT}$). Then we can try all possible values of C^*_{min} while only incurring a polynomial increase in the runtime of the algorithm. For w_{avg}, we first run an approximation algorithm for k-median with z outliers to obtain a constant approximation to w_{avg} (e.g., [38]). The constant in the minimum allowed optimal cluster size then increases by a factor of 7. This is because we need to use a smaller value of τ when constructing the neighborhood graph G', and so the number of "bad" points increases. In order to show all the good connected components from G' contain a majority of good points, we merely increase the bound on the minimum cluster size.

Distributed Setting. Next, we give a distributed algorithm for approximation stability with outliers using $\tilde{O}(sk+z)$ communication. However, as opposed to worst case, we can get close to the ground truth (target) clustering. In Section 4, we show a matching lower bound.

▶ **Theorem 8** (Distributed Clustering). *Given a $(1+\alpha, \epsilon)$-approximation stable clustering instance, there exists an algorithm that runs in poly$\left(n^{\frac{1}{\alpha}}\right)$ time and with high probability outputs a clustering that is $O(\epsilon)$-close to $\mathcal{OPT}$ for k-median with $\tilde{O}(sk+z)$ communication if each optimal cluster C^*_i has cardinality at least* $\max\left\{2\left(1+\frac{22}{\alpha}\right)\epsilon(n-z), \Omega\left(\frac{(n-z)}{sk}\right)\right\}$.

We start by giving intuition for our algorithm where there are no outliers. The high-level structure of the algorithm can be thought of as a two-round version of the centralized algorithm from approximation stability with no outliers [12]. Each machine effectively creates a coreset of its input, consisting of a weighted set of points, and sends these weighted points to the coordinator. The coordinator runs the same algorithm on these sets of weighted centers, to output the final solution.

In the analysis, we define good and bad points using Property *(1)* above with $y = 20$ as opposed to $y = 5$, so that there are more bad points than in the non-distributed setting, $|B| = \left(1+\frac{1}{20}\right)\epsilon(n-z)$, but for each optimal cluster C^*_i, the good points G_i are even more tightly concentrated. In the first round, each machine computes the neighborhood graph described above with parameter $\tau = \frac{w_{avg}}{10}$. This more stringent definition of τ ensures that

Claims *(1)* and *(2)* above are not only true for the input point set, but also true for a summarized version of the point set, where each point represents a ball of data points within a radius of τ. Therefore, there is still enough structure present such that the coordinator can compute a near-optimal clustering, and finally the coordinator sends the k resulting (near optimal) centers to each machine.

Now we expand this approach to the case with outliers. The starting point of the algorithm is the same: we perform two rounds of the sequential approximation stability algorithm with no outliers, so that each machine computes a summary of its point set, and the coordinator clusters the points it receives. Recall that in the centralized setting, running the non-outlier algorithm produces a list of clusters $\mathcal{X}$, some of which are near-optimal and some of which are outlier clusters, and then we crucially computed the $\text{cost}_{\min}$ of each potential cluster to distinguish the near-optimal clusters from the outlier clusters. In the distributed setting, we can construct the set $\mathcal{X}$ using the two-round approach.

However, the $\text{cost}_{\min}$ computation is sensitive to small sets of input points, and, as a result, the coresets will not give the coordinator enough information to perform this step correctly. In particular, this involves finding the closest points to a component that increase the cardinality to $C^*_{\min}$, and these points may be arbitrarily partitioned across the machines. Furthermore, the centralized algorithm can easily try all possible centers to compute the minimum cost of a given component Q, but it is much harder in the distributed setting to even find an approximately optimal center. Even with a center c chosen, the coordinator needs a near-exact estimate of the minimum cost of Q, however, it does not know the $C^*_{\min}$ closest points to c. Therefore our distributed algorithm must balance accuracy with communication.

For each component Q, the coordinator simulates $\log n$ random draws from Q by querying its own weighted points, and then querying the machine of the corresponding point. This allows the coordinator to find a center c whose cost is only a constant factor away from the best center. To compute $\text{cost}_{\min}(c)$, the coordinator runs a binary-search procedure with all machines to find the minimum distance t such that $B_t(c)$ contains more than $C^*_{\min}$ points.

Given a random point v from Q, by a Markov inequality, there is a $1/2$ chance that the cost of center v on V_c is at most twice the cost with center c. From a Chernoff bound, by sampling $10 \log n$ points for each component, each component will find a good center with high probability. Therefore, the coordinator can evaluate the cost of each component up to a factor of 2, which is sufficient to (nearly) distinguish the outlier clusters from the near-optimal clusters. The rest of the algorithm is similar to the centralized setting. We brute-force all combinations of removing x low-cost clusters from $\mathcal{X}$ and adding back x high-cost clusters from x. We perform one more cluster purifying step, and then check the cost of the resulting clustering. If the cost is smaller than $(1+\alpha)w_{avg}(n-z)$, then we return this clustering.

Similar to the centralized setting, we can use existing algorithms (e.g. [32]) to approximate w_{avg}, and we can use binary search to find $C^*_{\min}$. The algorithm communicates $\tilde{O}(sk+z)$ bits to approximate w_{avg}. The communication in the first step is $O(sk \log n)$, since there are at most $\min\left\{\frac{s}{\epsilon}, O(sk)\right\}$ sets of size at least $\max\left\{\frac{\epsilon n}{s}, \frac{n}{sk}\right\}$, each of which are communicated to the coordinator. The total communication to compute $\text{cost}_{\min}$ for every component is $\tilde{O}(sk)$. The binary search wrapper to find $C^*_{\min}$ adds a $\log n$ multiplicative factor. Therefore, the total communication is $\tilde{O}(sk+z)$.

4 Communication Complexity Lower Bounds

In this section, we show lower bounds for the communication complexity of distributed clustering with and without outliers. We prove $\Omega(sk + z)$ lower bounds for two types of clustering problems: computing a clustering whose cost is at most a c-approximation to the optimal (or even just to determine the cost up to a factor of c) for any $c \geq 1$, and computing a clustering which is δ-close to $\mathcal{OPT}$, for any $\delta < \frac{1}{4}$. This shows prior work of [32] is tight.

Our lower bounds hold even under c-separation (Definition 2). Furthermore, our lower bounds for δ-close clustering hold even under a weaker version of clustering, which we call *locally-consistent clustering*. In this problem, instead of assigning a globally consistent index in $\{1, \ldots, k\}$ for each point, each player only needs to assign indices to its points that is consistent in a local manner, e.g., the assignment of index set $\{1, \ldots, k\}$ to clusters $\{C_1, \ldots, C_k\}$ chosen by player 1 might be a permutation of the assignment chosen by player 2. We work in the communication model described in Section 2.

▶ **Definition 9** (Multi-party set disjointness ($\mathsf{DISJ}_{s,\ell}$)). *Given s players, denoted by P_1, P_2, $\ldots P_s$, player P_j receives as input a bit vector X^j of length ℓ. Let X denote the a binary matrix such that each X^j is a column of X. Let X_i denote the i-th row of X and $X^j[i]$ denote the (i, j)-th entry of X. Then, $\mathsf{DISJ}_{s,\ell} = \bigvee_{i\in[\ell]} \bigwedge_{j\in[s]} X^j[i]$, i.e., $\mathsf{DISJ}_{s,\ell} = 0$ if at least one row of X corresponds to the all ones vector and 1 otherwise.*

We note that set disjointness is a fundamental problem in communication complexity and we use the following lower bound for $\mathsf{DISJ}_{s,\ell}$ in the message-passing model by [22]:

▶ **Theorem 10** (Communication complexity of $\mathsf{DISJ}_{s,\ell}$ [22]). *For any $\delta > 0$, $s = \Omega(\log(n))$ and $\ell \geq 1$, the randomized communication complexity of multi-party set disjointness, $R_\delta(\mathsf{DISJ}_{s,\ell})$, is $\Omega(s\ell)$.*

We use the above theorem to show a lower bound of $\Omega(sk + z)$ for distributed clustering algorithms that attain an approximation to the cost of the optimal clustering under center-based clustering objectives such as k- median and k-means even if the instance satisfies strong beyond-worse case stability assumptions. We note that our first reduction is a slight modification of the reduction that appears in [27] and we show how to extend the reduction to stable instances and to account for outliers. Intuitively, the parameters of the reduction are carefully chosen so that the clustering instance created either has k or $k + 1$ distinct locations, toggled by the disjointness instance being yes or no. The lower bound for outliers requires starting with a two player, balanced instance of set disjointness, introduced by [45].

▶ **Theorem 11.** *Given $c_1 \geq 1$, $c_2 \geq 0$, the communication complexity for computing a c_1-approximation for k-median, k-means, or k-center clustering is $\Omega(sk)$, even when promised that the instance satisfies c_2-separation. Further, for the case of clustering with z outliers, computing a c_1-approximation to k-median, k-means, or k-center cost, given the promise that the instance satisfies c_2-separation requires $\Omega(sk + z)$ bits of communication.*

We note that thus far we have ruled out a distributed clustering algorithm that has communication complexity less than $\Omega(sk + z)$ to output the exact clustering under strong stability assumptions. Next, we show an $\Omega(sk + z)$ lower bound when the goal is to return a clustering that is $\delta < \frac{1}{4}$-close to optimal in Hamming distance, i.e., outputting a clustering that differs from the optimal clustering in a δ-fraction of the points, given that the instance is $(1 + \alpha, \epsilon)$-stable for any setting of these parameters.

We show that our lower bounds hold even when the algorithm outputs a c-approximate solution to the clustering cost of a $\frac{1}{4}$-close clustering. Intuitively, the proof is again a reduction from $\mathsf{DISJ}_{s,\ell}$, similar to the proof of Theorem 11. The main difference is that the

coordinator now adds roughly $\frac{n}{2}$ copies of a subset of points in our construction, to make the optimal clustering stand out from the rest. The main technical challenge is to figure out how to add these points such that the optimal clustering stands out in both yes and no instances. Therefore, recovering an approximation to the optimal clustering in Hamming distance provides enough information to solve set disjointness.

▶ **Theorem 12.** *Given $c_1 \geq 1$, $c_2 \geq 0$, and $0 < \delta < \frac{1}{4}$, the communication complexity for computing a c_1-approximation to the k-median, k-means, or k-center objective with z outliers and outputting a clustering that is δ-close to the optimal, $\Omega(sk + z)$, even when promised that the instance satisfies c_2-separation.*

Though the above lower bounds are quite general, it is possible that the hard instances may have the optimal clusters to be very different in cardinality if sk is large. The smallest cluster may be size $O\left(\frac{n}{sk}\right)$, while the largest cluster may be size $\Omega(n)$. Often, real-world instances have roughly balanced clusters. There is a line of work on clustering with balance constraints on the clusters [5, 3, 30], and some of our algorithmic results assume a lower bound on the minimum cluster size.

We note that our previous reduction for proving a lower bound against δ-close clustering algorithms fundamentally relies on testing the cardinality of the clusters. Therefore, we extend our previous lower bounds to the setting where we are promised that the input clusters are well balanced, i.e., have roughly the same cardinality. We still consider algorithms that only get δ-close to the optimal clustering. We begin by defining the following basic notions from information theory:

▶ **Definition 13** (Entropy and conditional entropy)**.** *The entropy of a random variable X drawn from distribution μ, denoted as $X \sim \mu$, with support χ, is given by $H(X) = \sum_{x \in \chi} \Pr_\mu[X = x] \log \frac{1}{\Pr_\mu[X=x]}$. Given two random variable X and Y with joint distribution μ, the entropy of X conditioned on Y is given by $H(X \mid Y) = \mathbb{E}_{y \sim \mu(Y)} \left[\sum_{x \in \chi} \Pr_{\mu(X|Y=y)}[X = x] \log \frac{1}{\Pr_{\mu(X|Y=y)}[X=x]}\right]$.*

Note, the binary entropy function $H_2(X)$ is the entropy function for the distribution $\mu(X)$ supported on $\{0, 1\}$ such that $\mu(X) = 1$ with probability p and $\mu(X) = 0$ otherwise.

▶ **Definition 14** (Mutual information and conditional mutual information)**.** *Given random variables X and Y, the mutual information between X and Y is given by $I(X;Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X)$. The conditional mutual information between X and Y, conditioned on a random variable Z is given by $I(X;Y \mid Z) = H(X \mid Z) - H(X \mid Y, Z) = H(Y \mid Z) - H(Y \mid X, Z)$.*

Recall, the δ-error randomized communication complexity of $\mathcal{A}$, $R_\delta(\mathcal{A})$, in the message passing model is communication complexity of any randomized protocol Π that solves $\mathcal{A}$ with error at most δ. Let μ be a distribution over $X^1, X^2, \ldots X^s$. We call a deterministic protocol (δ, μ)-error if it gives the correct answer for $\mathcal{A}$ on at least a $1-\delta$ fraction of the input, weighted by the distribution μ. Let $D_{\mu,\delta}(\mathcal{A})$ denote the cost of the minimum communication (δ, μ)-error protocol. By Yao's minimax lemma, $R_\delta(\mathcal{A}) \geq \max_\mu D_{\mu,\delta}(\mathcal{A})$. Therefore, in order to lower bound the randomized communication complexity of $\mathcal{A}$, it suffices to construct a distribution μ over the input such that any deterministic protocol that is correct on $1 - \delta$ fraction of any input can be analyzed easily. The communication complexity of a protocol Π is also lower bounded by its information complexity.

▶ **Definition 15** (Information complexity of $\mathcal{A}$)**.** *For $i \in [s]$, let Π_i be a random variable that denotes the transcript of the messages sent by player P_i to the coordinator. We overload notation by letting Π denote the concatenation of Π_1 to Π_s. Then, the information complexity of $\mathcal{A}$ is given by $IC_{\mu,\delta}(\mathcal{A}) = \min_{(\delta,\mu)\text{-error } \Pi} I(X_1, X_2, \ldots X_s; \Pi)$.*

Since information lower bounds communication (see, e.g., [35]), $R_\delta(\mathcal{A}) \geq \mathsf{IC}_{\mu,\delta}(\mathcal{A})$ in the message passing model. So our proof strategy is to construct a distributed protocol for solving the above problem using an algorithm that obtains a δ-close clustering for balanced clusters. We then design a distribution μ over the input and lower bound the information complexity of the resulting problem by $\Omega(k)$. We then amplify the bound by introducing $s/2$ copies of Alice and Bob (as before). Next, we describe this proof strategy in detail.

We begin with a two-player communication problem, where Alice and Bob receive length ℓ bit vectors, and the objective is to compute the AND function on each index in $[\ell]$. We then construct a gadget that reduces computing AND on any particular index to solving a 2-clustering problem, where each cluster has 2 points (and thus the instance is balanced). The gadget is such that Alice and Bob insert 2 points each, at a fixed set of locations determined by their input, and the optimal 2-clustering places Alice's points in different clusters iff the AND evaluates to true. The same holds for Bob. Therefore, Alice and Bob learn each other's bit simply looking at the output of the clustering algorithm. The players then create this gadget for each index in their input, and place the gadgets sufficiently far from each other.

Observe, a δ-close clustering algorithm must output a $(1-2\delta)$-fraction of the clusters correctly. Using such an algorithm as a distributed protocol enables the players to learn the AND function on a $(1-2\delta)$-fraction of the coordinates. Note the underlying communication problem here does not correspond to well-studied problems such as set disjointness. However, some proofs of the lower bound for multi-party set disjointness do reduce to computing the AND function on every index [19]. Therefore, we relate the communication complexity of the above problem to the amount of information revealed by any protocol that is correct on a large fraction of the input.

We define a distribution μ over the input such that each bit for Alice and Bob is set to be 1 with probability 1/2 independently and 0 otherwise. Here, we observe that the δ-close clustering algorithm implies a $(2\delta, \mu)$-protocol for computing AND on each index. Therefore, we prove that the information complexity of a $(2\delta, \mu)$-protocol is $\Omega(\ell)$. Intuitively, this says any correct deterministic protocol that is correct on a $1-2\delta$ fraction of the input, for the given input distribution μ, must reveal $\Omega(1)$ information on every index that has at least one 1, which amounts to communicating the bit. Since our gadget has 2 clusters for each index, setting $\ell = \Theta(k)$ obtains an $\Omega(k)$ communication lower bound. Using our previous strategy of duplicating the Alice and Bob players $s/2$ times, we obtain the following theorem:

▶ **Theorem 16.** *Given $\delta < \frac{1}{4}$ and the promise that the optimal clusters are balanced, i.e., the cardinality of each cluster is $\frac{n}{k}$, the communication complexity for computing a clustering that is δ-close to the optimal k-means or k-median clustering is $\Omega(sk)$.*

Finally, we extend the above lower bound to clustering instances that are balanced and also satisfy $(1+\alpha, \epsilon)$-approximation stability, again obtaining an $\Omega(sk+z)$ lower bound. Perhaps surprisingly, we show that there is no trade-off between the stability parameters and the communication lower bound even if the clusters are balanced and the algorithm outputs a clustering that is $\delta < \epsilon/4$ close to the optimal clustering. In contrast, our previous result can handle all $\delta < 1/4$. Intuitively, to obtain a clustering instance that is $(1+\alpha, \epsilon)$-approximation stable, we restrict the number of indices on which AND evaluates to 1 to be $O(\epsilon n)$. Therefore we start with a promise version of the multi-party set disjointness problem, where the promise states if the sets intersect, they intersect on exactly one element. Formally,

▶ **Definition 17** (Promise multi-party set disjointness ($\mathsf{PDISJ}_{s,\ell}$))**.** *Given s players denoted by $P_1, \ldots, P_s$, each player receives a bit vector X^j of length ℓ. Let X denote a binary matrix such that each X^j is a column of X. Let X_i denote the i-th row of X and X_i^j denote*

the (i, j)-th entry of X. We are promised that at most one row of X has all ones. Then, $\mathsf{PDISJ}_{s,\ell} = \bigvee_{i\in[\ell]} \bigwedge_{j\in[s]} X_i^j$, i.e., $\mathsf{PDISJ}_{s,\ell} = 0$ if any row of X corresponds to the all ones vector and 1 otherwise.

We use a result of [19] to lower bound the communication complexity of set-disjointness in the multi-party communication model.

▶ **Theorem 18** (Communication complexity of $\mathsf{PDISJ}_{s,\ell}$ [19]). *For any $\delta > 0$, $s, \ell \in \mathbb{N}$, the randomized communication complexity of promise multi-party set disjointness, $R_\delta(\mathsf{PDISJ}_{s,\ell})$, is $\Omega(\ell/s^2)$.*

We show any algorithm obtaining a δ-close clustering, given the clusters are balanced and the clustering instance is $(1+\alpha, \epsilon)$-stable can be converted into a randomized communication protocol that solves $\mathsf{PDISJ}_{s,\ell}$. At a high level, Alice and Bob receive length ℓ bit vectors and create a gadget for each index in $[\ell]$. If the number of indices on which the bit vectors intersect is at most ϵk, the instance is $(1+\alpha, \epsilon)$-stable. We ensure this by constructing gadgets that incur an arbitrarily high cost in all other cases (see the Appendix for details).

We note that if our clustering instance has exactly one index on which AND evaluates to 1, it is easy for a randomized protocol to be incorrect with good probability. In order to circumvent this issue and maintain $(1+\alpha, \epsilon)$-stability, Alice and Bob create $\epsilon n - 1 = 2\epsilon k - 1$ dummy indices that are set to 1 for both players. Further, Alice and Bob use public randomness to agree on a uniform permutation of the padded input and apply this permutation before constructing the gadgets and running the clustering algorithm. Intuitively, permuting the indices ensures that the δ-close clustering gets a typical cluster right with reasonable probability, by being oblivious to the dummy clusters that were used as padding.

Since we uniformly permute the indices of the input before running the protocol, for any given index, the corresponding cluster has Hamming distance 0 from the optimal clustering with probability at least $1-\epsilon$. This implies at most an ϵ-fraction of the clusters are incorrect. The protocol outputs a clustering that is known to both Alice and Bob. For each index of their input, they know whether their pair of points lie in the same cluster or different clusters. Let $\mathcal{I}$ be the set of indices for which Alice and Bob's points lie in different clusters. If $\mathcal{I} > 4\epsilon k$, the protocol outputs fail. Otherwise, Alice communicates her input on the set $\mathcal{I}$ to Bob. Bob applies an inverse random permutation to indices in set $\mathcal{I}$, and verifies if the indices correspond to the dummy indices that were added or indeed the sets are not disjoint. Note the verification step requires additional communication. Since $\mathcal{I} \leq 4\epsilon k$, and ϵ is at most a small constant, the total additional communication is $O(k/c)$ for some large constant c.

Consider the case where the sets are not disjoint. Then there is an index i^* such that AND on this index evaluates to 1, and with probability at least $1-\epsilon$, the clustering algorithm correctly clusters the corresponding 2-means gadget. This implies that Alice and Bob know that their pair of points lie in different clusters, thus i^* is in the set $\mathcal{I}$ and Alice communicates her input on index i^* to Bob. Bob can then verify that i^* is not a dummy index and indeed the sets are not disjoint.

The case where the sets are disjoint is more subtle. Now the clustering algorithm may return $4\epsilon k$ indices such that Alice's points belong to separate clusters, i.e., they correspond to a $(1,1)$ input, therefore leading to false positives. However, we observe that we can verify if the sets are disjoint by Alice sending over her input bits on the set $\mathcal{I}$ to Bob. Bob can verify if they correspond to the dummy indices and the sets are indeed disjoint. This increases the over all communication by $O(k/c)$. We recall that the promise problem requires $\Omega(\ell) = \Omega(k)$ communication and thus the communication of the above protocol is $\Omega(k - \epsilon k) = \Omega(k)$. We use the technique of cloning Alice and Bob $s/2$ times, so communicating the solution to each player requires $\Omega(sk)$ bits of communication. Finally, we show how to extend this lower bound to the case of outliers to get an overall $\Omega(sk + z)$ lower bound:

▶ **Theorem 19.** *Given a* $(1+\alpha, \epsilon)$*-approximation stable instance with* z *outliers such that* $\epsilon = o(1)$ *and* $\delta < \frac{\epsilon}{4}$*, and the promise that the optimal clusters are balanced, i.e., the cardinality of each cluster is* $\frac{n-z}{k}$*, the communication complexity for computing a clustering that is* δ*-close to the optimal* k*-means or* k*-median clustering is* $\Omega(sk + z)$*.*

References

1 Margareta Ackerman and Shai Ben-David. Clusterability: A theoretical study. In *Artificial Intelligence and Statistics*, pages 1–8, 2009.

2 Manu Agarwal, Ragesh Jaiswal, and Arindam Pal. k-means++ under Approximation Stability. *Theoretical Computer Science*, 588:37–51, 2015.

3 Gagan Aggarwal, Tomás Feder, Krishnaram Kenthapadi, Samir Khuller, Rina Panigrahy, Dilys Thomas, and An Zhu. Achieving anonymity via clustering. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 153–162, 2006.

4 Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better Guarantees for k-Means and Euclidean k-Median by Primal-Dual Algorithms. *arXiv preprint*, 2016. arXiv:1612.07925.

5 Sara Ahmadian and Chaitanya Swamy. Approximation Algorithms for Clustering Problems with Lower Bounds and Outliers. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 69:1–69:15, 2016. doi:10.4230/LIPIcs.ICALP.2016.69.

6 Haris Angelidakis, Konstantin Makarychev, and Yury Makarychev. Algorithms for Stable and Perturbation–Resilient Problems. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, 2017.

7 Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.

8 Pranjal Awasthi and Maria-Florina Balcan. Center based clustering: A foundational perspective. *CRC*, 2014.

9 Pranjal Awasthi, Maria-Florina Balcan, Avrim Blum, Or Sheffet, and Santosh Vempala. On nash-equilibria of approximation-stable games. In *International Symposium on Algorithmic Game Theory*, pages 78–89. Springer, 2010.

10 Pranjal Awasthi, Avrim Blum, and Or Sheffet. Center-based clustering under perturbation stability. *Information Processing Letters*, 112(1):49–54, 2012.

11 Pranjal Awasthi and Or Sheffet. Improved spectral-norm bounds for clustering. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 37–49. Springer, 2012.

12 Maria-Florina Balcan, Avrim Blum, and Anupam Gupta. Clustering under approximation stability. *Journal of the ACM (JACM)*, 60(2):8, 2013.

13 Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. A discriminative framework for clustering via similarity functions. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, pages 671–680, 2008.

14 Maria-Florina Balcan and Mark Braverman. Finding Low Error Clusterings. In *COLT*, volume 3, pages 3–4, 2009.

15 Maria-Florina Balcan, Nika Haghtalab, and Colin White. k-center Clustering under Perturbation Resilience. In *Proceedings of the Annual International Colloquium on Automata, Languages, and Programming (ICALP)*, 2016.

16 Maria Florina Balcan and Yingyu Liang. Clustering under perturbation resilience. *SIAM Journal on Computing*, 45(1):102–155, 2016.

17 Maria-Florina Balcan, Heiko Röglin, and Shang-Hua Teng. Agnostic Clustering. In *ALT*, pages 384–398. Springer, 2009.

18 Maria-Florina F Balcan, Steven Ehrlich, and Yingyu Liang. Distributed k-means and k-median Clustering on General Topologies. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1995–2003, 2013.

19 Ziv Bar-Yossef, Thathachar S Jayram, Ravi Kumar, and D Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004.

20 Mohammadhossein Bateni, Aditya Bhaskara, Silvio Lattanzi, and Vahab Mirrokni. Distributed Balanced Clustering via Mapping Coresets. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2591–2599, 2014.

21 Yonatan Bilu and Nathan Linial. Are stable instances easy? *Combinatorics, Probability and Computing*, 21(05):643–660, 2012.

22 Mark Braverman, Faith Ellen, Rotem Oshman, Toniann Pitassi, and Vinod Vaikuntanathan. A tight bound for set disjointness in the message-passing model. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 668–677. IEEE, 2013.

23 Jarosław Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median, and positive correlation in budgeted optimization. In *Proceedings of the Annual Symposium on Discrete Algorithms (SODA)*, pages 737–756. SIAM, 2015.

24 Moses Charikar, Sudipto Guha, Éva Tardos, and David B Shmoys. A constant-factor approximation algorithm for the k-median problem. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, pages 1–10. ACM, 1999.

25 Moses Charikar, Samir Khuller, David M Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the Annual Symposium on Discrete Algorithms (SODA)*, pages 642–651. Society for Industrial and Applied Mathematics, 2001.

26 Chandra Chekuri and Shalmoli Gupta. Perturbation Resilient Clustering for k-Center and Related Problems via LP Relaxations. In *Proceedings of the International Workshop on Approximation, Randomization, and Combinatorial Optimization Algorithms and Techniques (APPROX-RANDOM)*, 2018.

27 Jiecao Chen, He Sun, David Woodruff, and Qin Zhang. Communication-Optimal Distributed Clustering. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 3720–3728, 2016.

28 Ke Chen. A constant factor approximation algorithm for k-median clustering with outliers. In *Proceedings of the Annual Symposium on Discrete Algorithms (SODA)*, volume 8, pages 826–835, 2008.

29 Michael B Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 163–172. ACM, 2015.

30 Travis Dick, Mu Li, Venkata Krishna Pillutla, Colin White, Maria Florina Balcan, and Alex Smola. Data Driven Resource Allocation for Distributed Learning. In *"Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)"*, 2017.

31 Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

32 Sudipto Guha, Yi Li, and Qin Zhang. Distributed Partial Clustering. In *Proceedings of the Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2017.

33 Rishi Gupta, Tim Roughgarden, and C Seshadhri. Decompositions of triangle-dense graphs. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 471–482. ACM, 2014.

34 Lingxiao Huang, Shaofeng Jiang, Jian Li, and Xuan Wu. Epsilon-coresets for clustering (with outliers) in doubling metrics. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 814–825. IEEE, 2018.

35 Zengfeng Huang, Bozidar Radunovic, Milan Vojnovic, and Qin Zhang. Communication complexity of approximate matching in distributed graphs. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 30. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

36 Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. A local search approximation algorithm for k-means clustering. In *Proceedings of the eighteenth annual symposium on Computational geometry*, pages 10–18. ACM, 2002.

37 Ari Kobren, Nicholas Monath, Akshay Krishnamurthy, and Andrew McCallum. A Hierarchical Algorithm for Extreme Clustering. In *Proceedings of the KDD International Conference on Knowledge Discovery and Data Mining*, pages 255–264. ACM, 2017.

38 Ravishankar Krishnaswamy, Shi Li, and Sai Sandeep. Constant Approximation for k-Median and k-Means with Outliers via Iterative Rounding. *CoRR*, abs/1711.01323, 2017. `arXiv:1711.01323`.

39 Amit Kumar and Ravindran Kannan. Clustering with spectral norm and the k-means algorithm. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 299–308. IEEE, 2010.

40 Shi Li and Xiangyu Guo. Distributed k-Clustering for Data with Heavy Noise. In *Advances in Neural Information Processing Systems*, pages 7849–7857, 2018.

41 Konstantin Makarychev, Yury Makarychev, Maxim Sviridenko, and Justin Ward. A bi-criteria approximation algorithm for k Means. In *APPROX*, 2016.

42 Gustavo Malkomes, Matt J Kusner, Wenlin Chen, Kilian Q Weinberger, and Benjamin Moseley. Fast Distributed k-Center Clustering with Outliers on Massive Data. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1063–1071, 2015.

43 Rafail Ostrovsky, Yuval Rabani, Leonard J Schulman, and Chaitanya Swamy. The effectiveness of lloyd-type methods for the k-means problem. *Journal of the ACM (JACM)*, 59(6):28, 2012.

44 Kim D Pruitt, Tatiana Tatusova, Garth R Brown, and Donna R Maglott. NCBI Reference Sequences (RefSeq): current status, new features and genome annotation policy. *Nucleic acids research*, 2011.

45 Alexander A. Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106(2):385–390, 1992.

46 Konstantin Voevodski, Maria-Florina Balcan, Heiko Röglin, Shang-Hua Teng, and Yu Xia. Min-sum clustering of protein sequences with limited distance information. In *International Workshop on Similarity-Based Pattern Recognition*, pages 192–206. Springer, 2011.

Capacitated Dynamic Programming: Faster Knapsack and Graph Algorithms

Kyriakos Axiotis
MIT, Cambridge, MA, USA
kaxiotis@mit.edu

Christos Tzamos
University of Wisconsin-Madison, USA
tzamos@wisc.edu

Abstract

One of the most fundamental problems in Computer Science is the *Knapsack* problem. Given a set of n items with different weights and values, it asks to pick the most valuable subset whose total weight is below a capacity threshold T. Despite its wide applicability in various areas in Computer Science, Operations Research, and Finance, the best known running time for the problem is $O(Tn)$. The main result of our work is an improved algorithm running in time $O(TD)$, where D is the number of distinct weights. Previously, faster runtimes for Knapsack were only possible when both weights and values are bounded by M and V respectively, running in time $O(nMV)$ [17]. In comparison, our algorithm implies a bound of $O(nM^2)$ without any dependence on V, or $O(nV^2)$ without any dependence on M. Additionally, for the unbounded Knapsack problem, we provide an algorithm running in time $O(M^2)$ or $O(V^2)$. Both our algorithms match recent conditional lower bounds shown for the Knapsack problem [10, 15].

We also initiate a systematic study of general *capacitated dynamic programming*, of which Knapsack is a core problem. This problem asks to compute the maximum weight path of length k in an edge- or node-weighted directed acyclic graph. In a graph with m edges, these problems are solvable by dynamic programming in time $O(km)$, and we explore under which conditions the dependence on k can be eliminated. We identify large classes of graphs where this is possible and apply our results to obtain linear time algorithms for the problem of k-sparse Δ-separated sequences. The main technical innovation behind our results is identifying and exploiting concavity that appears in relaxations and subproblems of the tasks we consider.

2012 ACM Subject Classification Theory of computation → Algorithm design techniques

Keywords and phrases Knapsack, Fine-Grained Complexity, Dynamic Programming

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.19

Category Track A: Algorithms, Complexity and Games

Acknowledgements We are grateful to Arturs Backurs for insightful discussions that helped us improve this work.

1 Introduction

A large number of problems in Computer Science can be formulated as finding the optimal subset of items to pick in order to maximize a given objective subject to capacity constraints.

A core problem in this class is the *Knapsack problem*: In this problem, each of the n items has a value and a weight and the objective is to maximize the total value of the selected items while having total weight at most T.

A standard approach for solving such capacitated problems is to use dynamic programming. Specifically, the dynamic programming algorithm keeps a state that tracks how much of the available capacity has already been exhausted. The runtime of these algorithms typically

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 19; pp. 19:1–19:13


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

incurs a multiplicative factor equal to the total capacity. In particular, in the case of the Knapsack problem the classical dynamic programming algorithm due to Bellman [6] has a runtime of $O(Tn)$.

In contrast, uncapacitated problems do not restrict the number of elements to be selected, but charge an extra cost for each one of them (i.e. they have a *soft* as opposed to a *hard* capacity constraint). The best known algorithms for these problems are usually much faster than the ones for their capacitated counterparts, i.e. for the uncapacitated version of knapsack one would need to pick all items whose value is larger than their cost. Therefore a natural question that arises is whether or when the additional dependence of the runtime on the capacity is really necessary.

In this work, we make progress towards answering this question by exploring when this dependence can be improved or completely eliminated.

Knapsack. We first revisit the Knapsack problem and explore under which conditions we can obtain faster algorithms than the standard dynamic programming algorithm.

Despite being a fundamental problem in Computer Science, no better algorithms are known in the general case for over 60 years and it is known to be notoriously hard to improve upon. The best known algorithm for the special case where both the weights and the values of the items are small and bounded by M and V respectively, is a result by Pisinger [17] who presents an algorithm with runtime $O(nMV)$.

Even for the subset sum problem, which is a more restricted special case of knapsack where the value of every item is equal to its weight, the best known algorithm beyond the textbook algorithm by Bellman [6] was also an algorithm by Pisinger [17] which runs in time $O(nM)$ until significant recent progress by Bringmann [7] and Koiliaris and Xu [14] was able to bring its the complexity down to $\widetilde{O}(n+T)$.

However, recent evidence shows that devising a more efficient algorithm for the general Knapsack problem is much harder. Specifically, [10, 15] reduce the $(\max,+)$-convolution problem to Knapsack, proving that any truly subquadratic algorithm for Knapsack (i.e. $O((n+T)^{2-\epsilon})$) would imply a truly subquadratic algorithm for the $(\max,+)$-convolution problem. The problem of $(\max,+)$-convolution is a fundamental primitive inherently embedded into a lot of problems and has been used as evidence for hardness for various problems in the last few years (e.g. [10, 15, 3]). However, an important open question remains here: *Can we get faster algorithms that circumvent this conditional lower bound?*

We answer this question affirmatively by providing an algorithm running in time $O(TD)$, where D is the number of *distinct* weights. Our algorithm is deterministic and computes the optimal Knapsack value for *all* capacities t from 1 to T. Since $D \leq n$, its runtime either matches (for $D = \Theta(n)$) or yields an improvement (for $D = o(n)$) over Bellman's algorithm [6], for all parameter regimes. It also directly implies runtimes of $O(TM)$[1], $O(nM^2)$[2], and $O(nV^2)$, and therefore also yields an improvement over the $O(nMV)$ algorithm of Pisinger [17].

---

[1] Concurrent and independent work by Bateni, Hajiaghayi, Seddighin, and Stein [4] also obtains an algorithm running in time $\widetilde{O}(TM)$, as well as an algorithm running in time $\widetilde{O}(TV)$. In comparison to ours, their $\widetilde{O}(TM)$ algorithm is randomized and computes the answer only for a single capacity T.

[2] Eisenbrand and Weismantel [11] develop fast algorithms for Integer Programming. Concurrently and independently, they also obtain an algorithm for Knapsack that runs in time $O(nM^2)$. They provide a structural property of Knapsack using the Steinitz Lemma that enables us to remove logarithmic factors in T from our results for Unbounded Knapsack (Theorem 13), as they reduce to the case $T = \Theta(M^2)$. Combined with Theorem 8, this also implies an $O(M^3)$ algorithm for Knapsack.

Table 1 Summary of our deterministic pseudopolynomial time results on the Knapsack problem with the corresponding known conditional lower bounds based on $(\min, +)$-convolution.

Setting	Our Results	Conditional Lower bounds
Knapsack		
No bounds on weights or values	$O(TD)$ **[Theorem 8]**	$\Omega((TD)^{1-o(1)})$ [10, 15]
Weights bounded by M	$O(TM)$ **[Corollary 11]**	$\Omega((TM)^{1-o(1)})$ [10, 15]
Values bounded by V	$O(nV^2)$ **[Corollary 12]**	–
Unbounded Knapsack		
Weights bounded by M	$O(M^2)$ **[Corollary 14]**	$\Omega(M^{2-o(1)})$ [10, 15]
Values bounded by V	$O(V^2)$ **[Corollary 15]**	–

Our algorithm can be summarized as follows: First, it partitions the items into D sets according to their weights and solves the knapsack problem in each set of the partition for every possible capacity up to T. This can be done efficiently in $O(T)$ time as all items in each set have the same weight and thus knapsack can be greedily solved in those instances. Having a sequence of solutions for every capacity level for each set of items allows us to obtain the overall solution by performing $(\max, +)$-convolutions among them. Even though it is not known whether computing general $(\max, +)$-convolutions in truly sub-quadratic time is possible, we exploit the inherent concavity of the specific family of sequences produced by our algorithm to perform this in linear time. We present our results in Section 3.1.

In addition to the general Knapsack problem studied above, we also consider the *Unbounded Knapsack* problem where there are infinite copies of every item. In Section 3.2, we present novel algorithms for Unbounded Knapsack with running times $O(M^2)$[3] and $O(V^2)$, where M is the maximum weight and V is the maximum value of any item. Our algorithm again utilizes $(\max, +)$-convolutions of short sequences to compute the answer and interestingly is only pseudo-polynomial with respect to the maximum weight M or the maximum value V and not the capacity T.

Our results are summarized in Table 1.

It follows from the results of [10, 15] that, under the $(\min, +)$-convolution hardness assumption, it is not possible to obtain faster runtimes for Knapsack under most of the parameterizations that we consider. This is because, even though the lower bound claimed in these results is $\Omega((n+T)^{2-o(1)})$, the hardness construction uses a Knapsack instance where T, M, and D are $\Theta(n)$.

Capacitated Dynamic Programming. In addition to our results on the knapsack problem, we move on to study capacitated problems in a more general setting. Specifically, we consider the problem of computing a path of maximum reward between a pair of nodes in a weighted Directed Acyclic Graph, where the capacity constraint corresponds to an upper bound on the length of the path.

This model has successfully been used for uncapacitated problems [19, 15], as well as capacitated problems with weighted adjacency matrices that satisfy a specific condition, namely the *Monge property* [2, 18, 5]. In [5], it is shown that under this condition, the

[3] Jansen and Rohwedder [13] extend the results of [11] for Integer Programming and also concurrently and independently obtain an algorithm for Unbounded Knapsack running in time $O(M^2)$.

maximum weight of a path of length k is *concave* in k. Whenever such a concavity property is true, one can always solve the capacitated problem by replacing the capacity constraint with an "equivalent" cost per edge. This cost can be identified through a binary search procedure that checks whether the solution for the uncapacitated problem with this cost corresponds to a path of length k.

Our second main result, Theorem 18, gives a complete characterization of such a concavity property for transitive node-weighted graphs. We show that this holds if and only if the following graph theoretic condition is satisfied:
For every path $a \to b \to c$ of length 2, and every node v, at least one of the edges (a, v) and (v, c) exists.

To illustrate the power of our characterization, we show that a linear algorithm can be easily obtained for the problem of k-sparse Δ-separated subsequences [12] recovering recent results of [8, 16].

To complement our positive result which allows us to obtain fast algorithms for finding maximum weight paths of length k, we provide strong evidence of hardness for transitive node-weighted graphs which do not satisfy the conditions of our characterization. We base our hardness results on computational assumptions for the $(\max, +)$-convolution problem we described above.

Beyond node-weighted graphs, when there are weights on the edges, no non-trivial algorithms are known other than for Monge graphs. Even in that case, we show that linear time solutions exist only if one is interested in finding the max-weight path of length k between only one pair of nodes. If one is interested in computing the solution in Monge graphs for a single source but all possible destinations, we provide an algorithm that computes this in near-linear time in the number of edges in the graph.

2 Preliminaries

We first describe the problems of Knapsack and Unbounded Knapsack:

▶ **Definition 1** (Knapsack). *Given N items with weights $w_1, \ldots, w_N \in [M]$ and values $v_1, \ldots, v_N \in [V]$, and a parameter T, our goal is to find a set of items $S \subseteq [N]$ of total weight at most T (i.e. $\sum_{i \in S} w_i \leq T$) that maximizes the total value $\sum_{i \in S} v_i$. We will denote the number of distinct weights by D.*

▶ **Definition 2** (Unbounded Knapsack). *Given N items with weights $w_1, \ldots, w_N \in [M]$ and values $v_1, \ldots, v_N \in [V]$, and a parameter T, our goal is to find a multiset of items $S \subseteq [N]$ of total weight at most T (i.e. $\sum_{i \in S} w_i \leq T$) that maximizes the total value $\sum_{i \in S} v_i$. We will denote the number of distinct weights by D.*

Throughout the paper we make use of the following operation between two sequences called $(\max, +)$-convolution.

▶ **Definition 3** ($(\max, +)$-convolution). *Given two sequences $a_0, \ldots, a_n$ and $b_0, \ldots, b_m$, the $(\max, +)$-convolution $a \oplus b$ between a and b is a sequence $c_0, \ldots, c_{n+m}$ such that for any i*

$$c_i = \max_{0 \leq j \leq i} \{a_j + b_{i-j}\}$$

This operation is commutative, so it is also true that

$$c_i = \max_{0 \leq j \leq i} \{a_{i-j} + b_j\}$$

Our algorithms rely on uncovering and exploiting discrete concavity that is inherent in the problems we consider.

▶ **Definition 4** (Concave, k-step concave). *A sequence $b_0, \dots, b_n$ is* concave *if for all $i \in [n-1]$ we have $b_i - b_{i-1} \geq b_{i+1} - b_i$. A sequence is called k-*step concave *if its subsequence $b_0, b_k, b_{2k}, \dots$ is concave and for all i such that $i \mod k \neq 0$, we have that $b_i = b_{i-1}$.*

For the problems defined on graphs with edge weights, we typically assume that their weighted adjacency matrix is given by a Monge matrix.

▶ **Definition 5** (Monge matrices). *A matrix $A \in \mathbb{R}^{n \times m}$ is called Monge if for any $i \in [n-1]$ and $j \in [m-1]$*

$$A_{i,j} + A_{i+1,j+1} \geq A_{i+1,j} + A_{i,j+1}$$

▶ **Definition 6** (Monge weights). *We will say that a Directed Acyclic Graph has* Monge weights *if its weighted adjacency matrix is a Monge matrix.*

In addition to our positive results, we present evidence of computational hardness assuming for $(\max, +)$-convolution problem.

▶ **Definition 7** ($(\max, +)$-convolution hardness). *The $(\max, +)$-convolution hardness hypothesis states that any algorithm that computes the $(\max, +)$-convolution of two sequences of size n requires time $\Omega(n^{2-o(1)})$.*

A result of [3] shows that the $(\max, +)$-convolution problem is equivalent to the following problem: Given an integer n and three sequences $a_0, \dots, a_n$, $b_0, \dots, b_n$, and $c_0, \dots, c_n$, compute $\max_{i+j+k=n} \{a_i + b_j + c_k\}$. In our conditional lower bounds, we will be using this equivalent form of the conjecture.

3 Knapsack

In this section we present two novel pseudo-polynomial deterministic algorithms, one for Knapsack and one for Unbounded Knapsack. The running times of these algorithms significantly improve upon the best known running times in the small-weight regime. In essence, the main improvements stem from a more principled understanding and systematic use of $(\max, +)$-convolutions. Thus, we show that devising faster algorithms for special cases of $(\max, +)$-convolution lies in the core of improving algorithms for the Knapsack problem. In Theorem 8, we present an algorithm for Knapsack that runs in time $O(TD)$, where T is the size of the knapsack and D is the number of distinct item weights. Then, in Theorem 13, we present algorithms for Unbounded Knapsack with runtimes $O(M^2)$ and $O(V^2)$, where M is the maximum weight and V the maximum value of some item.

3.1 Knapsack

Given N items with weights $w_1, \dots, w_N \in [M]$ and values $v_1, \dots, v_N \in [V]$, and a parameter T, our goal is to find a set of items $S \subseteq [N]$ of total weight at most T (i.e. $\sum_{i \in S} w_i \leq T$) that maximizes the total value $\sum_{i \in S} v_i$. We will denote the number of distinct weights by D.

The following is the main theorem of this section.

▶ **Theorem 8.** *Algorithm 1 solves* Knapsack *in time $O(TD)$.*

Algorithm 1 Knapsack.

Given items with weights in $\{w_1^\#, \dots, w_D^\#\}$
Partition items into sets $S_1, \dots, S_D$, so that $S_i = \{j \mid w_j = w_i^\#\}$
for $i \in [D]$ and $t \in [T]$ **do**
    $b_t^{(i)} \leftarrow$ solution for S_i with knapsack size t
$s \leftarrow$ empty sequence
for $i \in [D]$ **do**
    $s \leftarrow s \oplus b^{(i)}$ using Lemma 10
    Truncate s after the T-th entry
Output s_T

Overview. The main ingredient of this result is an algorithm for fast $(\max, +)$-convolution in the case that one of the two sequences is *k-step concave*. Using the SMAWK algorithm [1] it is not hard to see how to do this in linear time for $k = 1$. For the general case, we show that computing the $(\max, +)$-convolution of the two sequences can be decomposed into $\frac{n}{k}$ subproblems of computing the $(\max, +)$-convolution between two size-k subsequences of the two sequences. Furthermore, the subsequence that came from the k-step concave sequence is concave and so each subproblem can be solved in time $O(k)$ and the total time spent in the subproblems will be $O(\frac{n}{k}k) = O(n)$.

▶ **Lemma 9.** *Given an arbitrary sequence $a_0, \dots, a_m$ and a concave sequence $b_0, \dots, b_n$ we can compute the $(\max, +)$ convolution between a and b in time $O(m + n)$.*

Proof. Consider a (zero-indexed) $(n + 1) \times (m + 1)$ matrix A with $A_{ij} = a_j + b_{i-j}$ for $(i, j) \in \{0, \dots, n\} \times \{0, \dots, m\}$, where we suppose that elements of the sequences with out-of-bounds indices have value $-\infty$. Note now that $(a \oplus b)_i$ is by definition equal to the maximum value of row i of A. Therefore computing $a \oplus b$ corresponds to finding the row maxima of A. Now note that for any $(i, j) \in \{0, 1, \dots, n-1\} \times \{0, 1, \dots, m-1\}$, we have

$$\begin{aligned} A_{i,j} - A_{i,j+1} &= a_j + b_{i-j} - a_{j+1} - b_{i-j-1} \\ &\overset{concavity}{\geq} a_j + b_{i+1-j} - a_{j+1} - b_{i-j} \\ &= A_{i+1,j} - A_{i+1,j+1} \end{aligned}$$

therefore A is Monge. The main result of [1] is that given an $(n+1) \times (m+1)$ Monge matrix A, one can compute all its row maxima in time $O(m + n)$, which implies the Lemma. ◀

▶ **Lemma 10.** *Given an arbitrary sequence $a_0, \dots, a_m$ and a k-step concave sequence $b_0, \dots, b_n$ we can compute the $(\max, +)$ convolution of a and b in time $O(m + n)$.*

Proof. We use the fact that we can compute the $(\max, +)$ convolutions of an arbitrary sequence with a concave sequence in linear time (Lemma 9). Since b is a k-step concave sequence, taking every k-th term of it one gets a concave sequence of size $O(n/k)$. Then, we do the same for a, taking k subsequences of size m/k each. Therefore we can compute the convolution between the concave sequence and all of these subsequences of a in linear time. The results of these convolutions can be used to compute the final sequence. We now describe this in detail.

For ease of notation, we will again assume that our sequences take value $-\infty$ in out-of-bounds indices. Let $x^{(i)} := (a_i, a_{k+i}, a_{2k+i}, \dots)$ denote the subsequence of a with indices whose remainder is i when divided by k, and $y := (b_0, b_k, b_{2k}, \dots)$. Furthermore, define

$$f_i = \max_{q=0}^{\infty} \{b_{qk} + a_{i-qk}\}$$

Now, for any j we have

$$\begin{aligned}\max_{i=j-k+1}^{j} f_i &= \max_{i=j-k+1}^{j} \max_{q=0}^{\infty}\{b_{qk} + a_{i-qk}\}\\ &= \max_{i=j-k+1}^{j} \max_{q=0}^{\infty}\{b_{qk+j-i} + a_{i-qk}\}\\ &= \max_{z=0}^{\infty}\{b_z + a_{j-z}\}\end{aligned}$$

where the second equality follows from the fact that $b_{qk+t} = b_{qk}$ for any $t \in [k-1]$ and the third from the fact that $z = qk + j - i$ can take any value in $[0, \infty)$.

This is the j-th element of the $(\max, +)$-convolution between a and b, so the elements of this convolution are exactly the maxima of size-k segments of f.

In order to compute f, note that for some p, the convolution between $x^{(p)}$ and y gives us all values of f of the form f_{qk+p}, for any q. This is because from the definition of f,

$$\begin{aligned}f_{qk+p} &= \max_{z=0}^{\infty}\{b_{zk} + a_{qk+p-zk}\}\\ &= \max_{z=0}^{\infty}\{y_z + x^{(p)}_{q-z}\}\\ &= (x^{(p)} \oplus y)_q\end{aligned}$$

Furthermore, y is a concave sequence and by Lemma 9 we can compute such a convolution in time $O((m+n)/k)$. Doing this for all $p \in \{0, \dots, k-1\}$, we can compute all values of f in time $O(m+n)$.

Now, in order to compute the target sequence, we have to compute the maxima of all size-k segments of f. We can do that using a simple sliding window technique. Specifically, suppose that for some segment $[i, i+k-1]$ we have an increasing subsequence of $f_{[i,\dots,i+k-1]}$, containing all the potentially useful elements. The first element of this subsequence is the maximum value of f in the segment $[i, i+k-1]$. Now, to move to $[i+1, i+k]$, we remove f_i if it is in the subsequence, and then we compare f_{i+k} with the last element in the subsequence. Note that if that last element has value $\leq f_{i+k}$, it will never be the maximum element in any segment. Therefore we can remove it and repeat until the last element has value greater than f_{i+k}, at which point we just insert f_{i+k} in the end of the subsequence. Note that by construction, this subsequence will always be decreasing, and the first element will be the maximum of the respective segment. The total runtime is linear if implemented with a standard queue. ◀

Now that we have these tools we can use them to prove the main result of this section:

Proof of Theorem 8. Consider any knapsack instance where D is the number of distinct item weights $w^{\#}_1, \dots, w^{\#}_D$. Now for each $i \in [D]$ let c_i be the number of items with weight $w^{\#}_i$ and $v^{(i)}_1 \geq v^{(i)}_2 \cdots \geq v^{(i)}_{c_i}$ their respective values.

If we only consider items with weights $w^{\#}_i$, the knapsack problem is easy to solve, since we will just greedily pick the most valuable items until the knapsack fills up. More specifically, if b_s is the maximum value obtainable with a knapsack of size s, we have that $b_0 = 0, b_{w_i} = v^{(i)}_1, b_{2w_i} = v^{(i)}_1 + v^{(i)}_2, \dots$, and also $b_j = b_{j-1}$ for any j not divisible by $w^{\#}_i$. Therefore b is a $w^{\#}_i$-step concave sequence.

In order to compute the full solution, we have to compute the $(\max, +)$ convolution of D such sequences. Since by Lemma 10 each convolution takes linear time and we only care about the first T values of the resulting sequence (i.e. we will only ever keep the first T values of the result of a convolution), the total runtime is $O(TD)$, where T is the size of the knapsack. ◀

▶ **Corollary 11.** *Knapsack can be solved in $O(TM)$ time.*

▶ **Corollary 12.** *Knapsack can be solved in time $O(nM^2)$ or $O(nV^2)$.*

Proof. The first bound directly follows by Corollary 11 and the fact that $T \leq nM$. For the second bound, note that by swapping the role of the weights and the values in Algorithm 1, replacing all $(\max, +)$-convolutions by $(\min, +)$-convolutions, and setting the knapsack capacity to nV as opposed to T, this algorithm runs in time $O(nV^2)$ and outputs for every possible value, the minimum weight of items that can achieve this value. The answer can then be recovered by finding the minimum value that gives a corresponding weight of at most T. ◀

3.2 Unbounded Knapsack

Given N items with weights $w_1, \ldots, w_N$ and values $v_1, \ldots, v_N$, and a parameter T, our goal is to find a multiset of items $S \subseteq [N]$ of total weight at most T (i.e. $\sum_{i \in S} w_i \leq T$) that maximizes the total value $\sum_{i \in S} v_i$. We will denote the largest item weight by M.

Note that this problem is identical to Knapsack except for the fact that there is no limit on the number of times each item can be picked. This means that we can assume that there are no two items with the same weight, since we would only ever pick the most valuable of the two.

Algorithm 2 Unbounded Knapsack.

Let $v^{(0)}$ be a sequence where $v^{(0)}_x$ is the value of the element with weight x or $-\infty$ if no such element exists
for $z = 1, \ldots, \lceil \log M \rceil$ **do**
  $v^{(z)} \leftarrow v^{(z-1)} \oplus v^{(z-1)}$
$a_{[0,M]} \leftarrow v^{(\lceil \log M \rceil)}$
for $i = \lceil \log \frac{T}{M} \rceil, \ldots, 1$ **do**
  $a_{\left[\frac{T}{2^i} - M, \frac{T}{2^i} + M\right]} \leftarrow a_{\left[\frac{T}{2^i} - M, \frac{T}{2^i}\right]} \oplus a_{[0,M]}$
  $a_{\left[\frac{T}{2^{i-1}} - M, \frac{T}{2^{i-1}}\right]} \leftarrow a_{\left[\frac{T}{2^i} - M, \frac{T}{2^i} + M\right]} \oplus a_{\left[\frac{T}{2^i} - M, \frac{T}{2^i} + M\right]}$
Output a_T

The following is the main theorem of this section:

▶ **Theorem 13.** *Algorithm 2 solves* Unbounded knapsack *in time $O(M^2 \log T)$.*

Overview. As in the algorithm for Knapsack our algorithm utilizes $(\max, +)$-convolutions, but with a different strategy. We aren't using any concavity arguments here, but in fact we will use the straightforward quadratic-time algorithm for computing $(\max, +)$-convolutions. The main argument here is that if all the weights are relatively small, one can always partition any solution in two, so that the weights of the two parts are relatively close to each other. Therefore, for any knapsack size we only have to compute the optimal values for a few knapsack sizes around its half, and not for all possible knapsack sizes.

We can now proceed to the proof of this result.

Proof of Theorem 13. Consider any valid solution to the unbounded knapsack instance. Since every item has weight at most M, we can partition the items of that solution into two multisets with respective weights W_1 and W_2, so that $|W_1 - W_2| < M$ (one can obtain

this by repeatedly moving any item from the larger part to the smaller one). This implies the following, which is the main fact used in our algorithm: If a_s is the maximum value obtainable with a knapsack of size s, then we have that

$$a_s = \left[\left(a_{s/2-M/2}, \dots, a_{s/2+M/2}\right)^{\oplus 2}\right]_s$$

where $^{\oplus 2}$ denotes $(\max,+)$-convolution squaring, i.e. applying $(\max,+)$-convolution between a sequence and itself.

First, we compute the values $a_1, \dots, a_M$ in $O(M^2 \log M)$ time as follows: We start with the sequence $v^{(0)}$, where $v_x^{(0)}$ is the value of the element with weight x, or $-\infty$ if such an item does not exist. Now define $v^{(i+1)} = (v^{(i)} \oplus v^{(i)})_{[0,M]}$. This convolution can be applied in time $O(M^2)$ for any i, since we are always only keeping the first M entries. By induction, it is immediate that $v^{(i)}$ contains the optimal values achievable for all knapsack sizes in $[M]$ using at most 2^i items. Therefore $a_{0,\dots,M} \equiv v_{0,\dots,M}^{(\lceil \log M \rceil)}$, which as we argued can be computed in time $O(M^2 \log M)$.

Now, suppose that we have computed the values $a_{\frac{T}{2^i}-M}, \dots, a_{\frac{T}{2^i}}$ for some i. By convolving this sequence with $a_0, \dots, a_M$ we can compute in time $O(M^2)$ the values $a_{\frac{T}{2^i}+1}, \dots, a_{\frac{T}{2^i}+M}$. Now, convolving the sequence $a_{\frac{T}{2^i}-M}, \dots, a_{\frac{T}{2^i}+M}$ with itself gives us $a_{\frac{T}{2^{i-1}}-M}, \dots, a_{\frac{T}{2^{i-1}}}$ (here we used the fact that to compute a_{2j} we only need the values $a_{j-M/2}, \dots, a_{j+M/2}$). Doing this for $i = \lceil \log T \rceil, \dots, 2, 1$, we are able to compute the values $a_{T-M}, \dots, a_T$ in total time $O(M^2 \log T)$. The answer to the problem, i.e. the maximum value achievable, is $\max\{a_{T-M}, \dots, a_T\}$. ◀

Recent work of [11] shows, using the *Steinitz lemma*, that an optimal Knapsack solution for a capacity in $[T-M, T]$ can be turned into an optimal solution for capacity T by inserting or removing at most M elements, where M is a bound on weight of the items. In the case of Unbounded Knapsack, a solution that only uses the best item until it exceeds capacity $T - M^2$ can always be extended into an optimal solution with capacity T. Therefore the capacity can be assumed to be $O(M^2)$. Combining this with the $\frac{M^2}{2^{\Omega\left(\sqrt{\log M}\right)}}$-time randomized algorithm of [20] (or the deterministic algorithm of [9]) for $(\max,+)$-convolution implies an algorithm that runs in time $\frac{M^2}{2^{\Omega\left(\sqrt{\log M}\right)}} O(\log M) = \frac{M^2}{2^{\Omega\left(\sqrt{\log M}\right)}}$.

▶ **Corollary 14.** *Unbounded Knapsack can be solved in time $O(M^2)$.*

A similar argument can be used to get a more efficient algorithm when we have a bound on the values of the items. In particular, using the item j with the highest value-to-weight ratio v_j/w_j, $k = \lfloor \frac{T}{w_j} \rfloor + 1$ times, until we exceed the capacity we get both a lower bound of $(k-1)v_j$ and a upper bound of kv_j on the value of the optimal solution. In addition, again by the Steinitz Lemma, there exists an optimal solution that uses item j at least $k - V$ times. This allows us to start from value $(k-V)v_j$ and compute the minimum weight required to achieve values in $[(k-1)v_j, kv_j]$. This gives an algorithm that runs in $O(V^2 \log V)$ using the naive algorithm for $(\min,+)$-convolutions, and $O\frac{V^2}{2^{\Omega\left(\sqrt{\log V}\right)}}$, again using the improved algorithms in [20] or [9].

▶ **Corollary 15.** *Unbounded Knapsack can be solved in time $O(V^2)$.*

4 k-link path in Node-weighted graphs

We now move on to study more general capacitated dynamic programming settings, described by computing the maximum reward k-link path in a directed acyclic graph. This setting can capture a lot of natural capacitated problems, either directly or indirectly, such as the Knapsack problem, k-sparse Δ-separated sequences, max-weight increasing subsequence of length k, and so on. Therefore a better understanding of these special cases might lead to improved algorithms for other capacitated problems.

More specifically, we study the problem of finding maximum-reward paths in node-weighted transitive DAGs. In Lemma 16, we show that in general this problem is hard, by reducing $(\max,+)$-convolution to it. We then proceed to show our second main result, which provides a family of graphs for which the problem can be efficiently solved.

▶ **Lemma 16** ($(\max,+)$-hardness of Node-weighted graphs). *Given a transitive DAG, a pair of vertices s and t, and an integer k, the problem of computing a maximum reward path from s to t with at most k edges is $(\max,+)$-convolution hard, i.e. requires $\Omega((mk)^{1-o(1)})$ time assuming $(\max,+)$-convolution hardness.*

As we saw in the introduction, one can solve the problem if the optimal value as a function of the capacity is concave. This is made formal in the following lemma:

▶ **Lemma 17** (Concave functions). *Let G be a node-weighted transitive DAG with n vertices and m edges, whose weights' absolute values are bounded by M, and let $f(x)$ be the maximum reward obtainable in a path of length x. If f is a concave function, then one can reduce the capacitated problem (i.e. computing $f(k)$ for some k) to solving $O(\log(nM))$ uncapacitated problems with some fixed extra cost per item. Since each one of these problems can be solved in $O(m)$ time, the total runtime is $O(m\log(nM))$.*

In Lemma 18 we give a complete graph-theoretic characterization of the graphs that have this concavity property and therefore can be solved efficiently.

▶ **Lemma 18** (Concavity characterization). *The problem of finding a maximum reward path with at most k edges in a transitive DAG is concave for all choices of node weights if and only if for any path $u_1 \to u_2 \to u_3$ and any node v either $u_1 \to v$ or $v \to u_3$ (Property $\mathcal{P}$).*

Proof. Let $f(k)$ be the maximum reward obtainable with a path of exactly k edges.
$\Rightarrow$: Let G be a DAG for which property $\mathcal{P}$ doesn't hold. Let $u_1 \to u_2 \to u_3$ be the path of length 2 and v be the vertex that has no edge to or from any of u_1, u_2, u_3. We set the node values as $val(u_1) = val(u_2) = val(u_3) = 1$, $val(v) = 1+\epsilon$, and $-\infty$ for all other vertices. Then, $f(1) = 1+\epsilon$, $f(3) = 3$, but $f(2) = 2 < \frac{f(1)+f(3)}{2}$, therefore f is not concave.
$\Leftarrow$: Suppose that property $\mathcal{P}$ is true. Now, let $P = (s, p_1, p_2, \ldots, p_{k-1}, t)$ be a path of length k such that $val(P) = f(k)$ and $Q = (s, q_1, q_2, \ldots, q_{k+1}, t)$ be a path of length $k+2$ such that $val(Q) = f(k+2)$, where P and Q can potentially have common vertices other than s and t. Since property $\mathcal{P}$ is true, we know that for any $i \in [k-1]$, there is either an edge from one of q_i, q_{i+1}, q_{i+2} to p_i, or from p_i to one of q_i, q_{i+1}, q_{i+2}. By transitivity, this implies that either $q_i \to p_i$, or $p_i \to q_{i+2}$. We distinguish three cases. In all three cases we will be able to find paths P' and Q' with $k+1$ edges each, that contain all vertices of the form p_i and q_i.

Case 1: $q_1 \to p_1$
We pick $P' = (s, q_1, p_1, \ldots, p_{k-1}, t)$ and $Q' = (s, q_2, \ldots, q_{k+1}, t)$.
Case 2: $\forall i : p_i \to q_{i+2}$
We pick $P' = (s, p_1, \ldots, p_{k-1}, q_{k+1}, t)$ and $Q' = (s, q_1, \ldots, q_k, t)$
Case 3: $\exists i : p_i \to q_{i+2},\ q_{i+1} \to p_{i+1}$
We pick $P' = (s, p_1, \ldots, p_i, q_{i+2}, \ldots, q_{k+1}, t)$ and $Q' = (s, q_1, \ldots, q_{i+1}, p_{i+1}, \ldots, p_{k-1}, t)$.

Therefore we established that in any case there exist such paths P' and Q'. Now note that

$$\max\{val(P'), val(Q')\} \geq \frac{1}{2}(val(P') + val(Q')) = \frac{1}{2}(val(P) + val(Q))$$

and therefore f is a concave function. ◀

As mentioned before, even very simple special cases of the model capture a lot of important problems. In the following lemma, we show that we can solve the k-sparse Δ-separated subsequence problem [12] in near-linear time using the main result of this section, thus recovering recent results of [8, 16].

▶ **Lemma 19** (Max-weight k-sparse Δ-separated subsequence). *Given a sequence $a_1, \dots, a_n$, find indices $i_1, i_2, \dots, i_k$ such that for all $j \in [k-1]$, $i_{j+1} \geq i_j + \Delta$ and the sum $\sum_{j\in[k]} a_{i_j}$ is maximized. This problem can be solved in $O(n \log(n \max_i |a_i|))$ time.*

Proof. Let's define a simple node-weighted DAG for this problem. We define a sequence of vertices $u_1, \dots, u_n$ each one of which corresponds to picking an element from the sequence. Then, we add an edge $u_i \to u_j$ iff $j - i \geq \Delta$. Furthermore, for all i, $val(u_i) = a_i$. It remains to prove that it satisfies the property of Lemma 18. Consider any length-2 path $u_i \to u_j \to u_k$. We know that both $k - j$ and $j - i$ are at least Δ. Now, for any u_p we have that

$$\max\{|u_p - u_k|, |u_p - u_i|\} \geq \frac{1}{2}(|u_p - u_k| + |u_p - u_i|) \geq \frac{1}{2}(|u_k - u_i|) \geq \frac{1}{2}2\Delta = \Delta$$

so there is an edge between u_p and either u_i or u_k. Therefore by Lemma 17 the problem can be solved in time $O(m \log(n \max_i |a_i|)) = O(n^2 \log(n \max_i |a_i|))$.

The quadratic runtime stems from the fact that the DAG we constructed is dense. In fact, we can do better by defining some auxiliary vertices $v_1, \dots, v_n$. The values of these extra vertices will be set to $-\infty$ to ensure that they aren't used in any solution and thus don't break the concavity. Instead of edges between vertices u_i, we only add the following edges

- $u_i \to v_i$ for all i
- $v_i \to u_{i+\Delta}$ for all $i + \Delta \leq n$
- $v_i \to v_{i+1}$ for all $i + 1 \leq n$

Now, the number of edges is $O(n)$ and so the runtime becomes $O(n \log(n \max_i |a_i|))$. ◀

As another example of a problem that can be modeled as a capacitated maximum-reward path problem in a DAG, we consider the Max-Weight Increasing Subsequence of length k problem. In contrast to its uncapacitated counterpart, which is solvable in linear time, the capacitated version requires quadratic time, assuming $(\max, +)$-convolution hardness, as witnessed in the following lemma.

▶ **Lemma 20** (Max-Weight Increasing Subsequence of length k). *Given a sequence $a_1, \dots, a_n$ with respective weights $w_1, \dots, w_n$, find indices $i_1 < i_2 < \dots < i_k$ such that for all $j \in [k-1]$, $a_{i_j} \leq a_{i_{j+1}}$ and the sum $\sum_{j\in[k]} w_{i_j}$ is maximized. This problem is $(\max, +)$-convolution hard, i.e. requires $\Omega((nk)^{1-o(1)})$ time assuming $(\max, +)$-convolution hardness.*

Proof. Consider the construction used in Lemma 16. We define an instance of the Max-Weight Increasing Subsequence of length k problem which contains an element for each node of the DAG. Specifically, let's define our sequence to be

$$x_0, x'_0, x_1, x'_1, \dots, x_k, x'_k, y_0, y'_0, \dots, y_k, y'_k, z_0, z'_0, \dots, z_k, z'_k$$

$$\text{with} \quad \begin{aligned} x_i &= i & x'_i &= 2k+1-i \\ y_i &= 2k+2+i & y'_i &= 4k+3-i \\ z_i &= 4k+4+i & z'_i &= 6k+5-i \end{aligned}$$

where the weight of each element is equal to the weight of the corresponding node in the DAG (i.e. $x_\star \leftrightarrow a_\star$, $x'_\star \leftrightarrow a'_\star$, $y_\star \leftrightarrow b_\star$, $y'_\star \leftrightarrow b'_\star$, $z_\star \leftrightarrow c_\star$, $z'_\star \leftrightarrow c'_\star$)

By definition of the sequence, the fact that we are looking for increasing subsequences implies that there is a $1-1$ correspondence between length-k increasing subsequences and $(k-1)$-hop paths of the original DAG. Therefore any $O((nk)^{1-\epsilon})$ algorithm for the Max-Weight Increasing Subsequence of length k problem implies a truly subquadratic algorithm for the $(\max,+)$-convolution problem. ◀

5 k-link path in graphs with Monge Weights

In this section we study the problem of computing maximum-reward paths with at most k edges in a DAG with edge weights satisfying the Monge property. Using the elegant algorithm of [5], one can compute a single such path in $\widetilde{O}(n)$ time.

▶ **Lemma 21** (From [5]). *Given a DAG with Monge weights, with n vertices, a pair of vertices s and t, and a positive integer k, we can compute a maximum reward path from s to t that uses at most k edges, in time $\widetilde{O}(n)$.*

Given the adjacency matrix A of the DAG, one can see this equivalently as computing one element of the matrix power A^k in the tropical semiring (i.e. we replace $(+,\cdot)$ with $(\max,+)$). Therefore, an important question is whether a whole row or column of A^k can be computed efficiently rather. This corresponds to finding maximum reward paths with k edges from some vertex s to *all* other vertices, or finding maximum reward paths with k edges from some vertex s to some vertex t for all k. In Lemma 22 we show that one needs $\Omega(n^{3/2})$ time to compute a column of A^k in general.

▶ **Lemma 22.** *Given a DAG with Monge weights, with n vertices, computing the maximum weight path of length k from a given s to all other nodes t requires $\Omega(n^{1.5})$ time.*

On the positive side, by further exploiting the Monge property, in Lemma 23 we present an algorithm that can compute any row or column of A^k in $\widetilde{O}(nnz(A)) = \widetilde{O}(m)$ time.

▶ **Lemma 23.** *Let G be a DAG of n vertices and m edges equipped with Monge weights that are integers of absolute value at most M. Given a vertex s, and a positive integer k, we can compute a maximum reward path from s to t that uses at most k edges, for all t, in time $O(m \log n \log(nM))$. Furthermore, if we are given a pair of vertices s and t, we can compute as maximum reward path from s to t that uses at most k edges, for all $k \in [n]$, in time $O(m \log n \log(nM))$.*

References

1 Alok Aggarwal, Maria M Klawe, Shlomo Moran, Peter Shor, and Robert Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2(1-4):195–208, 1987.

2 Alok Aggarwal, Baruch Schieber, and Takeshi Tokuyama. Finding a minimum-weightk-link path in graphs with the concave monge property and applications. *Discrete & Computational Geometry*, 12(3):263–280, 1994.

3 Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. Better approximations for tree sparsity in nearly-linear time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2215–2229. SIAM, 2017.

4 MohammadHossein Bateni, MohammadTaghi Hajiaghayi, Saeed Seddighin, and Cliff Stein. Fast algorithms for knapsack via convolution and prediction. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1269–1282, 2018.

5 WW Bein, LL Larmore, and JK Park. The d-edge shortest-path problem for a Monge graph. Technical report, Sandia National Labs., Albuquerque, NM (United States), 1992.

6 Richard Bellman. Dynamic Programming (DP). *Princeton University Press*, 1957.

7 Karl Bringmann. A near-linear pseudopolynomial time algorithm for subset sum. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1073–1084. Society for Industrial and Applied Mathematics, 2017.

8 Henning Bruhn and Oliver Schaudt. Fast Algorithms for Delta-Separated Sparsity Projection. *arXiv preprint*, 2017. `arXiv:1712.06706`.

9 Timothy M Chan and Ryan Williams. Deterministic apsp, orthogonal vectors, and more: Quickly derandomizing razborov-smolensky. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 1246–1255. Society for Industrial and Applied Mathematics, 2016.

10 Marek Cygan, Marcin Mucha, Karol Wegrzycki, and Michal Wlodarczyk. On Problems Equivalent to (min, +)-Convolution. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 22:1–22:15, 2017.

11 Friedrich Eisenbrand and Robert Weismantel. Proximity results and faster algorithms for integer programming using the steinitz lemma. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 808–816. Society for Industrial and Applied Mathematics, 2018.

12 Chinmay Hegde, Marco F Duarte, and Volkan Cevher. Compressive sensing recovery of spike trains using a structured sparsity model. In *SPARS'09-Signal Processing with Adaptive Sparse Structured Representations*, 2009.

13 Klaus Jansen and Lars Rohwedder. On Integer Programming and Convolution. *arXiv preprint*, 2018. `arXiv:1803.04744`.

14 Konstantinos Koiliaris and Chao Xu. A faster pseudopolynomial time algorithm for subset sum. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1062–1072. SIAM, 2017.

15 Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. On the Fine-Grained Complexity of One-Dimensional Dynamic Programming. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 21:1–21:15, 2017.

16 Aleksander Mądry, Slobodan Mitrović, and Ludwig Schmidt. A Fast Algorithm for Separated Sparsity via Perturbed Lagrangians. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, pages 20–28, 2018.

17 David Pisinger. Linear time algorithms for knapsack problems with bounded weights. *Journal of Algorithms*, 33(1):1–14, 1999.

18 Baruch Schieber. Computing a minimum weightk-link path in graphs with the concave monge property. *Journal of Algorithms*, 29(2):204–222, 1998.

19 Robert Wilber. The concave least-weight subsequence problem revisited. *Journal of Algorithms*, 9(3):418–425, 1988.

20 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 664–673. ACM, 2014.

Covering Metric Spaces by Few Trees

Yair Bartal
Department of Computer Science, Hebrew University of Jerusalem, Israel
yair@cs.huji.ac.il

Nova Fandina
Department of Computer Science, Hebrew University of Jerusalem, Israel
fandina@cs.huji.ac.il

Ofer Neiman
Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel
neimano@cs.bgu.ac.il

Abstract

A *tree cover* of a metric space (X, d) is a collection of trees, so that every pair $x, y \in X$ has a low distortion path in one of the trees. If it has the stronger property that every point $x \in X$ has a single tree with low distortion paths to all other points, we call this a *Ramsey* tree cover. Tree covers and Ramsey tree covers have been studied by [15, 31, 19, 30, 38], and have found several important algorithmic applications, e.g. routing and distance oracles. The union of trees in a tree cover also serves as a special type of spanner, that can be decomposed into a few trees with low distortion paths contained in a single tree; Such spanners for Euclidean pointsets were presented by [8].

In this paper we devise efficient algorithms to construct tree covers and Ramsey tree covers for general, planar and doubling metrics. We pay particular attention to the desirable case of distortion close to 1, and study what can be achieved when the number of trees is small. In particular, our work shows a large separation between what can be achieved by tree covers vs. Ramsey tree covers.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Theory of computation → Sparsification and spanners

Keywords and phrases tree cover, Ramsey tree cover, probabilistic hierarchical family

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.20

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1905.07559`.

Funding *Yair Bartal*: Supported in part by a grant from the Israeli Science Foundation (1817/17).
Nova Fandina: Supported in part by a grant from the Israeli Science Foundation (1817/17).
Ofer Neiman: Supported in part by a grant from the Israeli Science Foundation (1817/17) and in part by BSF grant 2015813.

Acknowledgements We are grateful to Michael Elkin and Shay Solomon for fruitful discussions.

1 Introduction

The problem of approximating metric spaces by tree metrics has been a successful research thread in the past decades, and has found numerous algorithmic applications. This is mainly due to the fact that a tree has a very simple structure that can be exploited by the algorithm designer. While a single tree cannot provide a meaningful approximation, due to a lower bound of [44] (the metric of the n point cycle requires $\Omega(n)$ distortion for embedding into a tree), several other variants have been considered in the literature. The purpose of this paper is to study the natural question whether there exists a small collection of trees (*tree cover*) such that each pair is well preserved in at least one of them. A natural stronger demand may be that for each point all of its interpoint distances to the rest of the metric are well preserved in one of the trees (*Ramsey tree covers*).

Tree covers and Ramsey tree covers have been studied by [31, 15, 19, 30, 38], and are useful ingredients in important algorithmic applications such as routing and distance oracles.

Given a metric space (X, d_X) and an edge-weighted tree T with $X \subseteq V(T)$, for $x, y \in X$ let $d_T(x, y)$ denote the length of the path in T from x to y. We say T is *dominating* if $d_T(x, y) \geq d_X(x, y)$ for all $x, y \in X$. A dominating tree T has *distortion* α for a pair $x, y \in X$, if $d_T(x, y) \leq \alpha \cdot d_X(x, y)$. In what follows, all trees we consider are always dominating (this can be assumed w.l.o.g.).

▶ **Definition 1** (Tree cover)**.** *Given a metric space (X, d_X), for $\alpha \geq 1$ and an integer k, a* tree cover *with distortion α and size k, (α, k)-tree cover in short, is a collection of k dominating trees $T_1, \ldots, T_k$, such that for any $u \neq v$ in X there is a tree T_i with distortion at most α for the pair u, v.*

If for each $u \in X$ there is a tree T_i with distortion at most α for each pair u, v with $v \in X$, we call this a Ramsey (α, k)-tree cover*.*

If the metric is a shortest path metric of some graph G, and the trees are subgraphs, we call this a *spanning* tree cover.

The notion of tree covers is closely related to the well studied notion of spanners. In the context of metric spaces, a *spanner* with distortion α for the metric (X, d_X), is a graph H with $X \subseteq V(H)$, so that for all $x, y \in X$, $d_X(x, y) \leq d_H(x, y) \leq \alpha \cdot d_X(x, y)$. It is often desired that the spanner would be a sparse graph. Note that taking H as the union of the trees in a (Ramsey) tree cover forms a sparse spanner with a special structure; that can be decomposed into a few trees, and every pair (or every point) has the distortion guarantee in one of these trees. In the context of graphs H is usually required to be a subgraph of the original graph, and then the same holds for spanning tree covers. Spanners are basic graph constructions, have been intensively studied [42, 7, 22, 20, 8, 24, 16, 47, 41] and have numerous applications in various settings, see e.g. [9, 43, 10, 22, 46, 25].

A related well-studied concept is probabilistic embedding of a metric space into tree metrics. This notion was introduced by Bartal [11], and a sequence of works by Bartal, and Fakcharoenphol et al. [12, 27, 13] culminated in obtaining a tight $O(\log n)$ bound. The result of [11] already implies a probabilistic construction of tree covers of size k with distortion $O(n^{2/k} \log n)$, for general metrics spaces. In Theorem 2 we improve this by constructing deterministic Ramsey tree covers with almost optimal distortion (nearly matching the lower bound in Theorem 27).

In the rest of the section we review known results on tree covers and Ramsey tree covers, and present the new results of this paper.

1.1 Tree Covers

In the context of Euclidean spanners, Arya et al. [8] used the so called dumbbell trees to build low distortion spanners. Rephrased in our context, they obtained a tree cover for Euclidean pointsets. More specifically, for any finite set of points in d-dimensional Euclidean space, and any parameter $\epsilon > 0$, they devised a $(1 + \epsilon)$-distortion tree cover with $O((d/\epsilon)^d \log(d/\epsilon))$ trees. We note that their trees are using Steiner points (i.e., points in Euclidean space that are not part of the input set), and it is not clear that such points can be removed from the spanner while maintaining $(1 + \epsilon)$ distortion.

Chan et al. [19] presented tree covers for doubling metrics. The *doubling constant* of a metric (X, d_X) is the minimal λ, so that every ball of radius $2r$ can be covered by λ balls of radius r. The *doubling dimension* of (X, d_X) is defined as $\log \lambda$, and a family of

metrics is called *doubling* if every metric in it has doubling dimension $O(1)$. The result of [19] used hierarchical partitioning to construct a tree cover with distortion $O(\log^2 \lambda)$ and $O(\log \lambda \cdot \log\log \lambda)$ trees.

The notion of spanning tree covers was introduced by Gupta et al. in [31], who used these for MPLS routing. They devised spanning tree covers for planar graphs: an exact (i.e. distortion 1) tree cover with $O(\sqrt{n})$ trees (more generally $O(r(n)\log n)$ trees for graphs admitting a hierarchical $r(n)$ size separators), and a spanning tree cover with distortion 3 and only $O(\log n)$ trees. They also showed the former result for planar graphs is tight, i.e., at least $\Omega(\sqrt{n})$ trees are needed for an exact tree cover.

1.1.1 Our results

As a starting point for this study, we observe, that for general metrics, the number of trees of any tree covers with distortion α must be as large as $n^{1/\alpha}$. (This bound stems from the standard example of high girth graphs and extends a previous lower bound for spanning trees of [31]). Nearly optimal upper bounds are known even for Ramsey tree covers (see next subsection). The above lower bound also implies a lower bound of $\lambda^{1/\alpha}$ in any space with doubling constant λ.

One of our main results is a tree cover for doubling metrics. We develop a novel hierarchical clustering for such metrics, built in a bottom-up manner. We then use this new clustering to show that for any $0 < \epsilon < 1$, every metric with doubling constant λ admits a tree cover with distortion $1+\epsilon$ and only $(1/\epsilon)^{O(\log \lambda)}$ trees. Since d-dimensional Euclidean space has doubling dimension $\Theta(d)$, the number of trees in the cover is therefore $(1/\epsilon)^{O(d)}$. Hence, this can be viewed as both a generalization and improvement of the result of [8]. Moreover, we improve their result in another aspect, since unlike [8] we do not require the use of Steiner points. In particular, for any $\epsilon > 0$ our result provides a $(1+\epsilon)$-spanner with $n/\epsilon^{O(\log \lambda)}$ edges, that can be decomposed to a small number of trees, and where each pair has a $1+\epsilon$ stretch path in one of the trees. We note that the number of edges in this spanner is asymptotically optimal [45], and thus so is our result.

We then turn to obtaining a distortion-size tradeoff for tree covers of doubling spaces with arbitrary distortion α. We improve and extend the result of [19]; for any parameter α, we use a more sophisticated construction of hierarchical partitions, to build a tree cover with distortion $O(\alpha)$ and $O(\lambda^{1/\alpha} \cdot \log \lambda \cdot \log \alpha)$ trees (note that setting $\alpha = \log \lambda$ yields distortion $O(\log \lambda)$ with $O(\log \lambda \cdot \log\log \lambda)$ trees). We note that the trees obtained here are in fact *ultrametrics*[1]. This result provides a special type of spanner with distortion $O(\alpha)$ and $O(n \cdot \lambda^{1/\alpha} \cdot \log \lambda \cdot \log \alpha)$ edges, which improves the recent spanner construction of [28], whose number of edges was larger by a factor of $O(\log_\lambda n)$ (though their spanner has additionally bounded lightness). The lower bound mentioned above for doubling spaces implies that our tree cover bounds cannot be substantially improved.

For planar graphs with n vertices, and more generally graphs excluding a fixed minor, we apply the path-separators framework of [48, 4], and show that for any $\epsilon > 0$, there exists a tree cover with distortion $1+\epsilon$ and $O((\log n)/\epsilon)^2$ trees. (Recall that for distortion 1, [31] used the planar separators of [37], but this requires $\Omega(\sqrt{n})$ trees.) We also observe that certain hierarchical partitions of [35] for planar (and fixed minor-free) graphs, can be used to obtain a tree cover with $O(1)$ distortion and only $O(1)$ trees (the obtained trees are ultrametrics).

[1] An ultrametric (U, d_U) is a metric satisfying a strong form of the triangle inequality, $\forall x, y, z$, $d_U(x, z) \leq \max\{d_U(x, y), d_U(y, z)\}$. An ultrametric is both a tree metric and a Euclidean metric.

See Table 1 for a succinct comparison between our and previous results on tree covers.

Table 1 Results on tree covers for general, planar (our new results also hold for fixed minor-free graphs) and doubling metrics. (The upper bound for general metrics appears in Table 2.)

Family	Reference	Number of trees	Distortion
General metrics	**New**	$\Omega(n^{1/\alpha})$	α
Doubling metrics	[19]	$O(\log\lambda \cdot \log\log\lambda)$	$O(\log^2\lambda)$
	New	$(1/\epsilon)^{\Theta(\log\lambda)}$	$1+\epsilon$
	New	$O(\lambda^{1/\alpha} \cdot \log\lambda \cdot \log\alpha)$	$O(\alpha)$
	New	$\Omega(\lambda^{1/\alpha})$	α
Planar metrics	[31]	$\Theta(\sqrt{n})$	1
	[31]	$O(\log n)$	3
	New	$O((\log n)/\epsilon)^2$	$1+\epsilon$
	[35]+**New**	$O(1)$	O(1)

1.2 Ramsey Tree Covers

Given a metric (X,d), the metric Ramsey problem asks for a large subset $S \subseteq X$ that embeds with a given distortion into a simple metric, such as a tree metric or Euclidean space. Following [15], [38] gave a probabilistic construction that finds in any n point metric (X,d) a set $S \subseteq X$ of size at least $n^{1-1/\alpha}$ that embeds into an ultrametric with distortion $O(\alpha)$. In fact, the embedding has such distortion on all pairs in $S \times X$. Applying this iteratively, [38] obtained a collection of $O(\alpha \cdot n^{1/\alpha})$ trees, so that each point $x \in X$ has a "home tree" T_x with distortion $O(\alpha)$ for every pair containing x. We call such a collection a *Ramsey tree cover*. Some further works aim at improving the leading constant in the distortion ([14, 40, 17]) and finding a deterministic construction ([14]). Recently, in the graph setting, [3] devised a *spanning* Ramsey tree cover, where the trees are subgraphs of the input graph. They obtained the same number of trees, but with slightly larger distortion $O(\alpha \cdot \log\log n)$.

We note that the number of trees in all previous works is $\alpha \cdot n^{1/\alpha} \geq \log n$ for any value of α. It seems like a natural question to understand what can be achieved in the inverse tradeoff, where the number of trees, k, is small. We remark that the lower bound via high girth graphs is rather weak, it implies that using k trees the distortion must be only $\Omega(\log_k n)$, as that is the bound on the girth of a graph with kn edges [18].

1.2.1 Our results

We focus on the regime where the number of trees is small. We first observe that a similar method as used in [38] of iteratively extracting large Ramsey subspaces can be applied in this setting as well. Given any metric space (X,d) on n points, and a parameter $k \geq 1$, there exists a Ramsey tree cover of size k (in particular, ultrametrics) and distortion $O(n^{1/k} \cdot \log^{1-1/k} n)$. We also note that the result of [3] can be translated to this setting: given a graph $G=(V,E)$ with n vertices, we find a Ramsey *spanning* tree cover with k spanning trees and distortion $O(n^{1/k} \cdot \log^{1-1/k} n \cdot \log\log n)$. The proof of this observation is given in the full version of the paper.

Next, we investigate the tightness of this bound. We find a graph on n vertices, such that any Ramsey tree cover with k trees requires distortion $\Omega(n^{1/k})$, significantly improving the $\Omega(\log_k n)$ bound obtained from high girth graphs. This also implies that our upper bound is tight, up to lower order terms.

Moreover, the graph we construct is series-parallel (in particular a planar graph) and also has $O(1)$ doubling dimension. Thus, our lower bound indicates a large separation between what can be achieved by a tree cover vs. a Ramsey tree cover; Our upper bounds give a tree cover with $O(1)$ trees and constant distortion for planar and doubling metrics (even $1+\epsilon$ distortion for the latter), as opposed to the $n^{\Omega(1)}$ distortion required with a constant number of Ramsey trees, for both planar and doubling metrics.

We also use a result of [15] to show a lower bound for planar and doubling metrics in the low distortion regime: there are n-point planar (in fact, series-parallel) doubling metrics, such that any Ramsey tree cover with distortion α must contain at least $n^{\Omega(1/(\alpha \log \alpha))}$ trees.

Overall, for general, planar and doubling metrics, our results solve the question of covering metrics by Ramsey trees, up to logarithmic terms, in every regime of parameters. See Table 2 for a concise description of previous and our results.

Table 2 Previous and our results on Ramsey trees for general, planar and doubling metrics.

Family	Reference	Number of trees	Distortion
General metrics	[38]	$O(\alpha \cdot n^{1/\alpha})$	$O(\alpha)$
	New	k	$O(n^{1/k} \cdot \log^{1-1/k} n)$
Planar & doubling metrics	**New**	k	$\Omega(n^{1/k})$
	New	$n^{\Omega(1/(\alpha \log \alpha))}$	α

1.3 Overview of Techniques

Tree cover for doubling metrics. The standard way to construct a $(1+\epsilon)$-spanner for doubling metrics is along the following lines [29, 32]: Choose a hierarchical collection of 2^i-nets (see Section 3 for definitions), and assign every vertex to its nearest net-point at the level i when it first leaves the net hierarchy; this creates a *net tree*. Then additional edges are added to other net-points within distance $\approx 2^i/\epsilon$. This spanner cannot be decomposed into a few trees as low distortion paths for the pairs use both the net tree and the additional edges.

We use a different approach for constructing a spanner, so that it can be decomposed to trees; We first partition the hierarchical net into a small number of well-separated sub-nets (so that in level i, distances between points in the sub-net are at least $2^i/\epsilon$). Then construct a tree for each hierarchical sub-net, by iterative clustering around the sub-net points in a bottom-up manner. In order to control the radius increase caused by the clustering of lower level sub-nets, we also take sufficiently large gaps between consecutive levels used in the same tree.

Tree cover for minor-free graphs. We apply the path separators of [4], asserting that graphs excluding a fixed minor have a separator consisting of $O(1)$ shortest paths (see Section 4 for the precise definitions). Adding for each point $O(1/\epsilon)$ edges to each shortest path guarantees small distortion for all separated pairs [48, 34]. However, since we desire trees, we can allow each point to add only 1 edge (per tree) to the path separator. Using a simple randomized algorithm to choose these edge connections, we show that w.h.p. all pairs will have a low distortion tree.

Hierarchical partitions. We construct a collection of HST spaces (a special type of ultrametric spaces, see Section 5) via a hierarchical probabilistic partitions similarly to [19]. Yet instead of using the basic probabilistic partitions (e.g. [11]) we use the probabilistic partitions

of [1], which have two main strong properties: The padding of the partition can be set as a parameter, which may also be a constant depending on distortion α; The partitions are local, i.e. the probability of being padded is not affected by the structure of the clusters that are far enough. This allows showing that intersecting a bundle of such independent partitions achieves a good tradeoff when the number of scale levels is small. Combining these with the idea of bottom-up union of clusters similar to that of [19], we are able to construct hierarchies with diameters of clusters decreasing by a *constant factor* while letting the padding parameter depend on α, obtaining more general and improved bounds.

1.4 Related Work

We note that Charikar et al. [21] studied a related question of bounding the number of trees sufficient for probabilistic embedding. The result they obtain implies an exponentially weaker cover size than those that follow from [11] and from our construction.

In [30], Gupta et al. considered a stronger version of tree covers (stronger than Ramsey tree covers), which they used to devise an oblivious algorithm for network design problems. We note that the lower bounds given in this paper show that the bound they get using this method is almost tight, even for doubling or planar metrics.

In the context of spanning trees, the problem of computing a spanning tree with low average stretch was first studied by Alon et al. [6]. Following Elkin et al. [23], Abraham et al. [2, 5] obtained a nearly tight $O(\log n \cdot \log\log n)$ bound.

2 Ramsey Tree Covers for General Metrics with Few Trees

In this section we show a deterministic Ramsey tree cover construction for general metrics. Unlike previous works, we build a cover with a small (possible constant) number of trees.

▶ **Theorem 2.** *For any n-point metric (X,d) and any $k \geq 1$, there is a deterministic algorithm that constructs a Ramsey tree cover for X of size k with distortion $O(n^{1/k} \cdot (\log n)^{1-1/k})$.*

Our deterministic construction follows directly from the following theorem on deterministic Ramsey embedding into a tree metric that was presented by Bartal [14] and later by Abraham et al. [3] (alternatively, a randomized construction can be based on [38]).

▶ **Theorem 3** ([14, 3])**.** *Let (X,d) be a metric space, fix any subset $S \subseteq X$, and let $\alpha \geq 1$ be a parameter. There is a deterministic algorithm that finds a subset $Z \subseteq S$, of size $|Z| \geq |S|^{1-\frac{1}{\alpha}}$, and an embedding f of X into an ultrametric T with distortion $O(\alpha)$ for any pair $(u,v) \in Z \times X$.*

Proof of Theorem 2. Let $S_1 = X$. For $i = 1, \ldots, k-1$ iteratively apply the algorithm of Theorem 3 on the subset S_i with parameter α (to be determined later), and obtain trees $T_1, \ldots, T_{k-1}$. The last tree T_k will be constructed separately. Let $Z_i \subseteq S_i$ be the set of size at least $|S_i|^{1-1/\alpha}$ guaranteed by Theorem 2, and define $S_{i+1} = S_i \setminus Z_i$. Note that every point $x \in X \setminus S_k$ has a tree with distortion $O(\alpha)$ for all pairs in $\{x\} \times X$. For each $i \geq 1$, we have $|S_{i+1}| = |S_i| - |Z_i| \leq |S_i| \left(1 - n^{-1/\alpha}\right)$. Therefore, after $k-1$ iterations we have $|S_k| \leq n \left(1 - n^{-1/\alpha}\right)^{k-1}$. We can embed the metric S_k into an ultrametric with distortion $|S_k| - 1$ by the embedding of [15, 32]. This embedding can be extended to all of X, with distortion $O(|S_k|)$ for pairs in $S_k \times X$ by [38, Lemma 4.1]. Therefore, α should be chosen so that $n \cdot \left(1 - n^{-1/\alpha}\right)^{k-1} \leq \alpha$. Using the inequality $e^x \geq 1 + x$ for all $x \in \mathbb{R}$ we have: $n \cdot \left(1 - n^{-1/\alpha}\right)^{k-1} = n \cdot \left(1 - e^{-\frac{\ln n}{\alpha}}\right)^{k-1} \leq n \cdot \left(\frac{\ln n}{\alpha}\right)^{k-1}$. Taking $\alpha = n^{1/k} \cdot (\ln n)^{\left(1-\frac{1}{k}\right)}$ gives a Ramsey tree cover with distortion $O(\alpha)$. ◀

3 $(1+\epsilon)$-Distortion Tree Covers for Doubling Metrics

In this section we devise a tree cover for doubling metrics with distortion arbitrarily close to 1. Let (X,d) be a metric with doubling constant λ, and fix $0<\epsilon<1/8$.

▶ **Definition 4.** *An r-net $N\subseteq X$ is a set satisfying: 1) For every $x,y\in N$, $d(x,y)>r$, and 2) For every $u\in X$ there exists $x\in N$ with $d(x,u)\le r$. We say that a collection $\{N_i\}$ of 2^i-nets is* hierarchical *if $N_{i+1}\subseteq N_i$.*

It is well-known that a simple greedy algorithm can construct (hierarchical) nets. Also, it is known that the size of an r-net of a ball of radius R is bounded by $\lambda^{O(\log(R/r))}$.

Let $\{N_i\}$ be a hierarchical collection of 2^i-nets of X. (It suffices to take the indices i from the range $[\log(\epsilon\delta),\log\Delta]$ where $\delta=\min_{x\neq y\in X}\{d(x,y)\}$ and $\Delta=\max_{x,y\in X}\{d(x,y)\}$.)

▷ Claim 5. There is a partition of N_i to $t=\lambda^{O(\log(1/\epsilon))}$ sets $N_{i1},\ldots,N_{it}$, so that for every $x,y\in N_{ij}$, $d(x,y)\ge 6/\epsilon\cdot 2^i$. It is also hierarchical: if $x\in N_{ij}$ then $x\in N_{i'j}$ for every $i'<i$.

Proof. First place in N_{ij} all the points of $N_{(i+1)j}$ for each j, and denote $N_i'=N_i\setminus(\bigcup_j N_{(i+1)j})$. Next, for $j=1,2,...,t$ complete N_{ij} by choosing greedily from the points remaining in $N_i'\setminus(N_{i1}\cup\cdots\cup N_{i(j-1)})$. Since for any $x\in N_i$ the ball of radius $6/\epsilon\cdot 2^i$ contains less than t net points of N_i, we will surely pick x to some N_{ij} in some iteration $j\le t$. ◁

Construction of trees. Assume w.l.o.g that $\log(1/\epsilon)$ is an integer, we will construct $t\cdot\log(1/\epsilon)$ trees (in fact, forests). The tree $T_{j,p}$ is indexed by the pair (j,p) with $1\le j\le t$ and $0\le p<\log(1/\epsilon)$. Fix j and p, we now describe how to build $T_{j,p}$. Let $I_p=\{i\,:\,i\equiv p\pmod{\log(1/\epsilon)}\}$. Initially all points in X are unclustered. We go over all $i\in I_p$ (from small to large in order), and for every $x\in N_{ij}$ we add an edge from x to every unclustered point $y\in X$ satisfying $d(x,y)<3/\epsilon\cdot 2^i$, of weight $d(x,y)$. These points connected to x are now clustered. (The center x is not considered clustered.)

The following observation is proved in the full version of the paper:

▶ **Observation 6.**

a. *No point $x\in N_{ij}$ is clustered when iteration i is complete.*

b. *Every $u\in X$ can be clustered by at most 1 point.*

c. *If C_x is the connected component created by the clustering of $x\in N_{ij}$ at level $i\in I_p$, then* $\mathrm{diam}(C_x)\le 8/\epsilon\cdot 2^i$.

▷ Claim 7. When the process completes we have a forest.

Proof. By Observation 6(b) every point u adds at most a single edge to $T_{j,p}$, at the time it becomes clustered. As u adds this edge to an unclustered point, it cannot close a cycle. (More formally, if we give each vertex a time stamp which is the time it becomes clustered, then the single edge every vertex adds is to a vertex with a higher time stamp.) ◁

▷ Claim 8. Let C_x be the connected component created when we clustered points to x at some level $i\in I_p$. Then for every point $y\in B(x,2/\epsilon\cdot 2^i)$ we have $d_{C_x}(x,y)\le d(x,y)+2^{i+4}$.

Proof. If y was unclustered when creating C_x then $d_{C_x}(x,y)=d(x,y)$ by definition. Otherwise, let z be the (unique) unclustered point in C_z, the connected component containing y before executing the clustering of level i. Let $i'<i$ be the level in which z created C_z, and by Observation 6(c) we have $\mathrm{diam}(C_z)\le 8/\epsilon\cdot 2^{i'}\le 8\cdot 2^i$ (recall $i'\le i-\log(1/\epsilon)$ since $i'\in I_p$). As $8\le 1/\epsilon$ it follows that $d(x,z)\le d(x,y)+d(y,z)\le 2/\epsilon\cdot 2^i+8\cdot 2^i\le 3/\epsilon\cdot 2^i$, so x will cluster z (recall that by Observation 6(b) no other center can cluster z). Furthermore, $d_{C_x}(x,y)=d_{C_x}(x,z)+d_{C_z}(z,y)=d(x,z)+d_{C_z}(z,y)\le d(x,y)+2d_{C_z}(z,y)\le d(x,y)+2^{i+4}$. ◁

▶ **Lemma 9.** *For every $u, v \in X$, there is a tree $T = T_{j,p}$ so that $d_T(u,v) \le (1+O(\epsilon))\cdot d(u,v)$.*

Proof. Choose i such that $2^{i-1}/\epsilon \le d(u,v) < 2^i/\epsilon$, and let $p = i \bmod \log(1/\epsilon)$. Let $x \in N_i$ be the nearest net point to u, so that $d(x,u) \le 2^i$, and let $1 \le j \le t$ be such that $x \in N_{ij}$.

Observe that $d(x,v) \le d(x,u) + d(u,v) \le 2^i + 2^i/\epsilon < 2/\epsilon \cdot 2^i$, so by Claim 8

$$d_{C_x}(x,v) \le d(x,v) + 2^{i+4} \le d(x,u) + d(u,v) + 2^{i+4} = (1+O(\epsilon)) \cdot d(u,v).$$

We also have by the same claim that $d_{C_x}(x,u) \le d(x,u) + 2^{i+4} = O(\epsilon) \cdot d(u,v)$. The fact that C_x is a subtree of the forest $T = T_{j,p}$ completes the proof. ◀

Since we use edge weights that are the actual distances in (X,d), clearly $d_T \ge d$. Rescaling ϵ by a constant yields the following.

▶ **Theorem 10.** *For every metric (X,d) with doubling constant λ, and any $0 < \epsilon < 1$, there is an efficient algorithm to construct a tree cover of size $\lambda^{O(\log(1/\epsilon))}$, with distortion $1+\epsilon$.*

4 $(1+\epsilon)$-Distortion Tree Covers for Planar and Minor-Free Graphs

In this section we use *path-separators* for planar [48] and more generally minor-free graphs [4], to devise tree covers with $1+\epsilon$ distortion and $O((\log n)/\epsilon)^2$ trees. We start with some preliminary definitions.

A graph G has H as a minor if one can obtain H from G by a sequence of edge deletions, vertex deletions and edge contractions. The graph G is H-minor-free if it does not contain H as a minor.

▶ **Definition 11.** *A graph $G = (V,E)$ on n vertices is* s-path separable *if there exists an integer t and a separator $S \subseteq V$ such that:*

1. *$S = V(\mathcal{P}_0) \cup V(\mathcal{P}_1) \cup \cdots \cup V(\mathcal{P}_t)$, where for each $0 \le i \le t$, $\mathcal{P}_i$ is a collection of shortest paths in the graph $G \setminus (\bigcup_{0 \le j < i} \mathcal{P}_j)$ (and $V(\mathcal{P}_i)$ is the vertex set used by the paths in $\mathcal{P}_i$).*
2. *$\sum_{i=0}^{t} |\mathcal{P}_i| \le s$, that is, the total number of paths is at most s.*
3. *Each connected component of $G \setminus S$ is s-path separable and has at most $n/2$ vertices.*

▶ **Theorem 12** ([4])**.** *Every H-minor-free graph is s-path separable for some $s = s(H)$, and an s-path separator can be computed in polynomial time.*

The following Lemma is implicit in the works of [34, 48] (for completeness a proof is included in the full version of the paper):

▶ **Lemma 13.** *Let $G = (V,E)$ be an edge-weighted graph, fix any $0 < \epsilon < 1$, and let P be a shortest path in G. Then one can find for each $x \in V$ a set of landmarks L_x on P of size $|L_x| = O(1/\epsilon)$, such that for any $x, y \in V$ whose shortest path between them intersects P, there exists $u \in L_x$ and $v \in L_y$ satisfying $d_G(x,u) + d_P(u,v) + d_G(v,y) \le (1+\epsilon) \cdot d_G(x,y)$.*

Construction. Using these tools, we are ready to describe our tree cover for minor-free graphs. Apply the path separator of Theorem 12 on the input graph $G = (V,E)$, $|V| = n$, to obtain a collection $\mathcal{P}$ of s paths, and denote $S = V(\mathcal{P})$. For each path $P \in \mathcal{P}$, apply Lemma 13 to get a set of landmarks for each vertex, and let $\ell = \max_{x \in V}\{|L_x|\} = O(1/\epsilon)$ be the maximal size of a landmark set. Let T be a tree formed by taking P, and for each $x \in V \setminus V(P)$ add a single edge to $u \in L_x$ chosen uniformly and independently at random. Let $d_G(x,u)$ be the weight of a chosen edge. We pick $(C \log n)/\epsilon^2$ such trees independently for each path P, for sufficiently large constant C.

Next, we continue recursively on each connected component of $G \setminus S$. Since the number of vertices halves at every iteration, there will be $O(\log n)$ iterations. Furthermore, the trees of different connected components can be viewed as a forest of G (which can be arbitrarily completed to a tree), thus the total number of trees is $O((\log n)/\epsilon)^2$.

Analysis. Fix some $x, y \in V$, and let P be the first path in $\mathcal{P}$ that intersects the shortest path between x, y in G. (It may be the case that $\mathcal{P}$ is a path separator in a deep level of the recursion, when we decompose some subgraph G'. Note that $d_{G'}(x,y) = d_G(x,y)$, since no path intersected the shortest path from x to y so far. So w.l.o.g. we call the current graph G.) Let $u \in L_x$ and $v \in L_y$ be such that $d_G(x,u) + d_P(u,v) + d_G(v,y) \leq (1+\epsilon) \cdot d_G(x,y)$, which are guaranteed to exist by Lemma 13. If we choose a tree T that contains P and both edges $(x,u),(y,v)$, then T will have distortion at most $1+\epsilon$ for the pair x, y. The probability that both x, y add these edges to T is at least $1/\ell^2 = \Omega(\epsilon^2)$. Thus, the probability that none of the trees created for the path P has distortion at most $1+\epsilon$ for the pair x, y is at most $(1-\Omega(\epsilon^2))^{(C \log n)/\epsilon^2} \leq e^{-3 \ln n} = 1/n^3$, whenever C is sufficiently large. By the union bound over the $\binom{n}{2}$ pairs, with high probability all pairs have a tree with distortion $1+\epsilon$ in that tree. We have proven the following.

▶ **Theorem 14.** *Let G be a graph on n vertices that is H-minor-free. For any $0 < \epsilon < 1$, there is a randomized efficient algorithm that w.h.p. constructs a tree cover for G containing $O((\log n)/\epsilon)^2$ trees with distortion $1+\epsilon$. (The constant in the O-notation depends on $|H|$.)*

5 Tree Covers for Doubling Metrics with Distortion-Size Tradeoff

In this section we prove that any metric space with doubling constant λ has a tree cover with distortion $O(\alpha)$ of size $O(\lambda^{1/\alpha} \log \lambda \log \alpha)$.

Recall that ultrametric is a metric space obeying a strong form of the triangle inequality. It is well known that any finite ultrametric (U, ρ) can be represented by a finite labeled tree T, with the points of U being the leaves of T. Each node $u \in T$ has a label $\Delta(u) \geq 0$ and the label of each leaf is 0. For any two nodes u and v, such that v is a child of u, $\Delta(u) \geq \Delta(v)$. For $u, v \in U$, the distance $\rho(u,v)$ is defined to be the label of their least common ancestor. We refer to ultrametrics by their tree representation. If the labels in an ultrametric tree T are decreasing by a factor at most $\mu > 1$, then T is called a μ-Hierarchically Separated Tree metric (μ-HST) [11]. We note that an ultrametric space can also be represented as a shortest path metric on a Steiner tree.

For a finite metric (X,d), let $d_{\max} = \max_{x \neq y \in X}\{d(x,y)\}$, and $d_{\min} = \min_{x \neq y \in X}\{d(x,y)\}$. Let $\Phi(X) := d_{\max}/d_{\min}$ denote the aspect ratio of X.

5.1 Probabilistic Hierarchical Partition Family

With start with the neceassary definitions. For any $\Delta > 0$, a Δ-bounded partition P of a finite metric space (X,d) is a collection of pairwise disjoint clusters $P_i \subseteq X$, such that $\cup P_i = X$, and for each cluster $P_i \in P$, $\mathrm{diam}(P_i) \leq \Delta$. We assume that each cluster has some point designated as its center. For a point $x \in X$, let $P(x) \in P$ denote the cluster that contains x. A Δ-bounded probabilistic partition of X is a distribution $\mathcal{P}$ over a set of Δ-bounded partitions of X.

The notion of a padding parameter of a random partition is studied in various papers [36, 33, 11, 27]. We use a stronger definition given by Abraham et al. in [1], where the padding parameter depends on the desired probability of success. The following is a rephrased version of their original definition ([1], Definition 17):

▶ **Definition 15** (Padded Probabilistic Partition). *Let $\eta(\delta)\colon (0,1] \to (0,1]$ be some function, and $(a,b] \subseteq (0,1]$ be some range. A Δ-bounded probabilistic partition $\mathcal{P}$ is $\eta(\delta)$-padded on the range $(a,b]$, if for all $x \in X$ and for all $\delta \in (a,b]$, $\Pr_{P\sim\mathcal{P}}[B(x,\eta(\delta)\cdot\Delta) \subseteq P(x)] \geq \delta$.*

In addition, the authors defined a notion of a *locally* padded probabilistic partition (on the range $(a,b]$): $\mathcal{P}$ is $\eta(\delta)$-locally padded if for all $a < \delta \leq b$ the event $B(x,\eta(\delta)\cdot\Delta) \subseteq P(x)$ occurs with probability at least δ regardless of the structure of the partition outside the ball $B(x,2\Delta)$. Formally stated, for all $x \in X$, for all subsets $C \subseteq X \setminus B(x,2\Delta)$ and all partitions P' of C, $\Pr_{P\sim\mathcal{P}}[B(x,\eta(\delta)\cdot\Delta) \subseteq P(x) \mid P[C] = P'] \geq \delta$, where $P[C]$ denotes the restriction of the partition P to C. Our construction uses their random partitions as a building block:

▶ **Lemma 16** ([1], Lemma 8). *Given a finite metric space X with doubling constant λ, and given any $0 < \Delta < \mathrm{diam}(X)$, there is a Δ-bounded, $\left(\frac{\log(1/\delta)}{2^6 \log\lambda}\right)$-locally padded probabilistic partition $\mathcal{P}$ of X, for $\delta \in \left[\lambda^{-2^{12}}, 1\right]$.*

A set of nested partitions of X forms a hierarchy:

▶ **Definition 17** (Hierarchical Partition). *For all $\mu > 1$, $\Delta \leq d_{\max}(X)$ and integer $1 \leq B \leq \log_\mu \Phi(X)$, let $\Delta_i = \Delta/\mu^i$, for all $0 \leq i \leq B$. A μ-Hierarchical Partition of X for range $[\Delta, \Delta_B]$, is a collection $H = \{P_0, \ldots, P_B\}$ of partitions of X such that: For all $0 \leq i \leq B$, P_i is a Δ_i-bounded partition of X; Each P_{i+1} is a refinement of P_i, i.e. each cluster in P_i is a union of some clusters in P_{i+1}. Let μ-$HP_B(\Delta)$ denote such a collection.*

A full range Hierarchical Partition, denoted by μ-HP, is the μ-$HP_B(\Delta)$, for $\Delta = d_{\max}(X)$ and $B = \log_\mu \Phi(X)$ (we assume this is an integer).

There is a natural way to associate a dominating μ-HST tree to a μ-HP. For each cluster of the partition P_i there is a node in the tree. The nodes associated with clusters of the partition P_{i+1} are the children of nodes associated with clusters of P_i. The label of all level i nodes in the tree is Δ/μ^i. The points of X are at the leaves.

▶ **Definition 18** (η-Padded μ-Hierarchical Partition Family). *Let $\eta < 1$ and $\mu > 1$. For a finite metric space (X,d), an η-padded μ-Hierarchical Partition Family of X, (η,μ)-HPF, is a set $\mathcal{H}$ of μ-Hierarchical Partitions $\{H^j\}_{j\geq 1}$ of X such that: For all $x \in X$ and for all scales $0 \leq i \leq \log_\mu \Phi(X)$, there is an $H^j \in \mathcal{H}$ such that $B(x,\eta\Delta_i) \subseteq P_i^{(j)}(x)$, where $P_i^{(j)}$ is a Δ_i-bounded partition of H^j. The size of $\mathcal{H}$ is the number of hierarchical partitions it has.*

The following lemma shows the connection between hierarchical family and a tree cover:

▶ **Lemma 19.** *If there is an (η,μ)-HPF of size k of X, then there is an $(\mu/\eta, k)$-tree cover of X.*

Proof. Let $H^1, \ldots, H^k$ be an (η,μ)-HPF of X. Consider an associated collection of dominating μ-HST trees $T_1, \ldots, T_k$. Given any $x \neq y \in X$, let i be the minimal index such that $d(x,y) \geq \eta\Delta_i$. If $i = 0$, then by the construction, for any tree T_j, $d_{T_j}(x,y) \leq \Delta_0$, implying $d_{T_j}(x,y)/d(x,y) \leq 1/\eta$. If $i \geq 1$, then $\eta\Delta_i \leq d(x,y) \leq \eta\Delta_{i-1}$. The padding property implies that there is H^j such that $B(x,\eta\Delta_{i-1}) \subseteq P_{i-1}^{(j)}(x)$. As $y \in B(x,\eta\Delta_{i-1})$, it holds that $d_{T^j}(x,y) \leq \Delta_{i-1}$. Therefore, $d_{T^j}(x,y)/d(x,y) \leq \Delta_{i-1}/\eta\Delta_i = \mu/\eta$. ◀

In what follows, we construct $(\Omega(1/\alpha), 2)$-HPF of X, of size $O(\lambda^{1/\alpha}\log\lambda\log\alpha)$. We note that the notion of hierarchical family also appeared in [35], where the authors constructed an $(\Omega(s^{-2})), O(s^2))$-HPF of size 3^s for any metric of a $K_{s,s}$-minor free graph. As a corollary, we conclude

▶ **Corollary 20.** *For any metric induced on a $K_{s,s}$-minor free graph, there is a tree cover with distortion $O(s^4)$, of size 3^s.*

In our proofs we will use the following version of the Lovasz Local Lemma:

▶ **Lemma 21** ([26]). *Let $\mathcal{E}_1, \ldots, \mathcal{E}_n$ be a family of events. Let $G(V, E)$ be a directed graph on n vertices with out-degree at most d, where each vertex corresponds to an event. Assume that for all $1 \leq i \leq n$, for all $Q \subseteq \{j \mid (\mathcal{E}_i, \mathcal{E}_j) \notin E\}$, $\Pr[\mathcal{E}_i \mid \bigcap_{j \in Q} \neg\mathcal{E}_j] \leq p$. If $ep(d+1) \leq 1$, then $\Pr\left[\bigcap_{i \in [1,n]} \neg\mathcal{E}_i\right] > 0$.*

5.2 Constructing Hierarchical Padded Family of Bounded Size

Our main hierarchical partitions result is:

▶ **Theorem 22.** *For any finite metric space X with doubling constant λ and for any $\alpha \geq 2$, there is an $\Omega(1/\alpha)$-padded 2-Hierarchical Partition Family of X, of size $O\left(\lambda^{1/\alpha} \log\alpha \log\lambda\right)$.*

Note that taking $\alpha = O(\log\lambda)$, we obtain a hierarchical family with padding $\Omega(1/\log\lambda)$, of size $O(\log\lambda \log\log\lambda)$, which is an improvement over the result of [19]: $O(\log\lambda)$-hierarchical partitions with padding $\Omega(1/\log\lambda)$, of the same size. They construct a family of hierarchies, where each hierarchy is constructed in a bottom-up manner: the clusters of larger diameters are the union of the clusters of lower diameters. Preserving the padding parameter requires the diameters of the clusters to increase by a factor of $O(\log\lambda)$, thus covering only $\log\log\lambda$ of all the distance scales in the metric space. This results in $O(\log^2\lambda)$ distortion. Using the Lovasz Local Lemma they were able to bound the size of this family.

In our construction, we combine the bottom-up union of clusters technique with an *intersection of clusters* procedure. Essentially, there are two steps. First, we use the locally padded partitions of Lemma 16 to create a padded hierarchy with diameters decreasing by a constant factor, by intersecting the clusters of levels of the hierarchy of larger diameter. Using the locality property and the fact that the padding parameter depends on the success probability, we show that using $\log\log\lambda$ such levels of partitions with diameters increasing by factor 2, results in a 2-hierarchy with $\Omega(1/\alpha)$ padding, thus covering the $\log\log\lambda$ scales uncovered by the construction of [19]. We apply the Lovasz Local Lemma to bound the size of the family of such hierarchies by $O\left(\lambda^{1/\alpha} \log\alpha \log\lambda\right)$. Second, we combine the hierarchies obtained by cutting clusters, in a bottom-up manner, by defining higher scales clusters as the union of lower level clusters, thus obtaining a hierarchy with diameters decreasing by a factor of 2 in all its levels, while padding is $\Omega(1/\alpha)$.

To prove Theorem 22, we consider hierarchical partitions that cover a range of scales: for any Δ and an integer B we build a family $\{H^j\}_{j \geq 1}$, where each H^j is a μ-HP$_B(\Delta)$. The padding property is then required to hold for all points $x \in X$ and for all scales $\Delta_i \in [\Delta, \Delta_B]$. We call such family as (η, μ)-HPF for range $[\Delta, \Delta_B]$.

The following lemma is used as a subroutine in the construction of the hierarchical family:

▶ **Lemma 23.** *Let X be a finite metric space with doubling constant λ . For a given $\alpha \geq 2$, $\Delta \leq \mathrm{diam}(X)$ and an integer $1 \leq B \leq \log_\mu \Phi(x)$, there exists an $(\Omega(1/\alpha), 2)$-HPF for range $[\Delta, \Delta_B]$, of size $O\left(\lambda^{1/\alpha} \log\lambda(\log\alpha + B)\right)$.*

Proof. For a given distortion $\alpha \geq 2$, let $\delta = \lambda^{-1/(2\alpha)}$. Therefore, for such δ we have $\eta(\delta) := \frac{\log(1/\delta)}{2^6 \log\lambda} = 2^{-7}/\alpha$. Also note that for any $\alpha \geq 1$, it holds that $\delta \in [\lambda^{-2^{12}}, 1]$. Thus, we will show that there exists a hierarchical family with padding $\Omega(\eta(\delta))$, of size $k := O(\left(\lambda^{1/\alpha} \log\lambda(\log\alpha + B)\right))$.

Let $N \subseteq X$ be an $(\eta(\delta)\Delta_B/4)$-net of X. We show the claim is true for N and the extension of it to X is immediate, with a constant factor loss in distortion. In the sequel, all the balls are balls of metric space N.

Let $\Delta_i = \Delta/2^i$, for all $0 \leq i \leq B$. Consider the following random process: For each scale Δ_i in the range $[\Delta, \Delta_B]$, independently generate Δ_i-bounded partitions $P_0, \ldots P_B$ of N by invoking the locally padded probabilistic decomposition of Lemma 16. To obtain a 2-Hierarchical Partition H for the scales $[\Delta_0, \Delta_B]$ we cut all the clusters of all the partitions, to get Δ_i-bounded nested partitions $\hat{P}_0, \ldots, \hat{P}_B$. Let $\hat{P}_0 = P_0$, for all $i \geq 1$, define $\hat{P}_i = \cup_{\hat{C} \in \hat{P}_{i-1}} \cup_{C \in P_i} C \cap \hat{C}$.

Now, independently repeat the above random process k times to obtain a randomly generated family $H^{(1)}, \ldots, H^{(k)}$ of 2-Hierarchical Partitions of the net N, for range $[\Delta, \Delta_B]$. Each hierarchical partition $H^{(t)}$ consists of Δ_i-bounded partitions, denoted by $\hat{P}_i^{(t)}$.

For each $x \in N$ and for each scale $\Delta_i \in [\Delta, \Delta_B]$, let $\mathcal{E}_{x,i}$ be an event that the ball $B(x, \eta(\delta)\Delta_i)$ is not padded at the i-th level partition $\hat{P}_i^{(t)}$ in any of the hierarchical partitions $H^{(1)}, \ldots, H^{(k)}$. We use the Lovasz Local Lemma (Lemma 21) to prove that for the chosen value of k, $\Pr\left[\bigcap_{\substack{x \in X, \\ 0 \leq i \leq B}} \neg\mathcal{E}_{x,i}\right] > 0$. Let $G = (V, E)$ be a directed graph with $V = \{\mathcal{E}_{x,i}\}$, for all $x \in N$ and $0 \leq j \leq B$. The vertex $\mathcal{E}_{x,i}$ is connected with an out-edge with all the verticies $\mathcal{E}_{y,j}$, such that $y \in B(x, 2\Delta)$ and $0 \leq j \leq B$. In the full version we prove the following lemma:

▶ **Lemma 24.** *For all $Q \subseteq N \setminus B(x, 2\Delta)$, for all $J \subseteq [0, B]$, $Pr[\mathcal{E}_{x,i} \mid \bigcap_{\substack{y \in Q, \\ j \in J}} \neg\mathcal{E}_{y,j}] \leq (1-\delta^2)^k$.*

Then, for $\delta = \lambda^{-1/(2\alpha)}$, and $k = O\left(\lambda^{1/\alpha} \log\lambda(\log\alpha + B)\right)$, $(1-\delta^2)^k \leq e^{-\delta^2 k} \leq \lambda^{-\Theta(\log\alpha + B)}$.

In addition, the out degree d of G is bounded by

$$d = B \cdot |N \cap B(x, 2\Delta)| = B \cdot O\left(\left(\frac{\Delta}{\eta(\delta)\Delta_B}\right)^{\log\lambda}\right) = B \cdot \lambda^{O(\log(1/\eta(\delta)) + B)} = \lambda^{O(\log(1/\eta(\delta)) + B)}.$$

Thus, the LLL can be applied to conclude the proof. ◀

Proof of Theorem 22. Let $\Phi = \Phi(X)$, $\Delta_0 = d_{\max}(X)$, and for all $1 \leq i \leq \log\Phi$, $\Delta_i = \Delta_0/2^i$. Let $\mathcal{I} = \{\Delta_i \mid 0 \leq i \leq \log\Phi\}$. We build a small family of 2-HP's, such that the padding property is satisfied for all points in X and for all scales $\Delta_i \in \mathcal{I}$, with padding parameter $\Omega(1/\alpha)$. Let $B = \lceil \log(2\alpha/c') \rceil$, where $c' < 1$ will be defined later, and let $k = O\left(\lambda^{1/\alpha} \log\alpha \log\lambda\right)$. We will build a family $\mathcal{H} \cup \mathcal{R}$ such that:

1. $\mathcal{H}$ is a collection of size k of 2-HP's. For[2] $\mathcal{I}_{\mathcal{H}} := \left\{\Delta_j \mid \Delta_j \in \bigcup_{0 \leq l \leq L} \left[\Delta_{2lB}, \Delta_{(2l+1)B}\right]\right\} \subseteq \mathcal{I}$, where $L = \log(\Phi^{1/(2B)}) - 1/2$, the following padding property holds: For all $x \in X$ and for all scale $\Delta_j \in \mathcal{I}_{\mathcal{H}}$, there is a 2-HP $H \in \mathcal{H}$ such that $B(x, \Omega(1/\alpha)\Delta_j) \subseteq P_j(x)$, for the j-th level partition $P_j \in H$.
2. $\mathcal{R}$ is a collection of size k of 2-HP's. The padding property as for $\mathcal{H}$ holds for all $x \in X$ and for all scales $\Delta_j \in \mathcal{I}_{\mathcal{R}} := \mathcal{I} \setminus \mathcal{I}_{\mathcal{H}}$.

Namely, the hierarchical partitions of $\mathcal{R}$ are padded for the scales that are not padded in the partitions of $\mathcal{H}$. Thus, together these two collections constitute an $\Omega(1/\alpha)$-padded 2-HP Family for X, of size $2k$. We describe the construction of $\mathcal{H}$, while $\mathcal{R}$ is constructed similarly.

[2] For the simplicity of representation, we assume that B is integer and that $\log\Phi$ is a multiple of B.

Let $\mathcal{H} = H^{(1)}, \ldots, H^{(k)}$ denote the set of 2-HP's. We construct it iteratively in a bottom up fashion. Assume by induction that we have already constructed a family $\hat{\mathcal{H}} = \hat{H}^{(1)}, \ldots, \hat{H}^{(k)}$ such that: Each $\hat{H}^{(t)}$ is a 2-HP for range $[\Delta_{2B}, \Delta_{(2L+1)B}]$; The padding property holds with parameter $\Omega(1/\alpha)$ for all scales $\Delta_j \in \mathcal{I}_{\mathcal{H}} \setminus \{\Delta_i \in [\Delta_0, \Delta_B]\}$.

Let $c = 1 + \frac{1}{2^{B-1}-1}$, by Lemma 23 there is $(\Omega(1/\alpha), 2)$-HPF $F^{(1)}, \ldots, F^{(k)}$ for range $[\tilde{\Delta}_0, \tilde{\Delta}_B]$, where $\tilde{\Delta}_j = \Delta_j/c$, for $0 \le j \le B$. For each $1 \le t \le k$, $H^{(t)}$ is obtained by adding the partitions of $F^{(t)}$ to $\hat{H}^{(t)}$ in the following way. Let $\hat{P}^{(t)}_{2B}$ denote the Δ_{2B}-bounded partition of $\hat{H}^{(t)}$. First, for all scale $\Delta_j \in [\Delta_{B+1}, \Delta_{2B}]$ add to $H^{(t)}$, Δ_j-bounded partition $P^{(t)}_j := \hat{P}^{(t)}_{2B}$ (these artificial partitions are added to have a well defined 2-HP family). Next, let $\{\tilde{P}^{(t)}_j\}_{0 \le j \le B}$ denote the set of $\tilde{\Delta}_j$-bounded partitions of $F^{(t)}$. For all j starting from $j = B$ down to $j = 0$, the partition $P^{(t)}_j$ is constructed a s follows: for each $\tilde{C} \in \tilde{P}^{(t)}_j$, add a cluster C to $P^{(t)}_j$, defined by $C = \cup\{C' \in P^{(t)}_{2B} \mid \text{center of } C' \in \tilde{C}\}$. Finally, the partitions of $\hat{H}^{(t)}$ are unchanged.

For all $B \le j \le 2B$, $P_j(t)$ is Δ_j-bounded, since $\Delta_{2B} \le \Delta_j$. For $0 \le j \le B$ the diameter of each cluster in partition $P^{(t)}_j$ is bounded by $\tilde{\Delta}_j + 2\Delta_{2B} = \Delta_j/c + \Delta_B/2^{B-1} \le \Delta_j\left(1/c + 1/2^{B-1}\right) = \Delta_j$, for a chosen value of c. In addition, by the construction, the partitions $P^{(t)}_j$ form a hierarchy. It is left to show that the padding property holds in $\mathcal{H}$ for the scales $[\Delta_0, \Delta_B]$. By Lemma 23, for any $x \in X$, for any $\tilde{\Delta}_j \in [\tilde{\Delta}_0, \tilde{\Delta}_B]$, there is $F^{(t)}$ such that $B(x, (c'/\alpha)\tilde{\Delta}_j) \subseteq \tilde{P}^{(t)}_j(x)$, for $\tilde{P}^{(t)}_j \in F^{(t)}$, for some constant c'. Consider some cluster $P^{(t)}_j(x)$, for some $x \in X$. In the process of constructing $P^{(t)}_j(x)$ some points from $\tilde{P}^{(t)}_j(x)$ may be removed, due to removal of some cluster $C' \in P^{(t)}_{2B}$ whose center falls outside the cluster $\tilde{P}^{(t)}_j(x)$. For B as defined above, for $r = \frac{c'}{\alpha}\tilde{\Delta}_j - \Delta_{2B} \ge \left(\frac{c'}{2\alpha}\right)\Delta_j$, we have that $B(x, r) \subseteq P^{(t)}_j(x)$. This completes the proof. ◀

▶ **Theorem 25.** *For any finite metric space (X, d) with doubling constant λ, for any $\alpha \ge 2$, there is a tree cover of X with distortion $O(\alpha)$ and of size $O\left(\lambda^{1/\alpha} \log \alpha \log \lambda\right)$.*

Proof. Apply Lemma19 on the Hierarchical Family of Theorem 22. ◀

Note that the tree cover of Theorem 25 can be deterministically constructed in polynomial time via the constructive local lemma due to [39].

6 Lower Bounds

In full version of the paper we show that there is an n-point metric (X, d) (the metric of a high girth graph) with doubling constant λ, such that any tree cover for X with distortion α requires at least $\Omega(\lambda^{1/\alpha})$ trees. Additionally, we show below a nearly tight lower bound for Ramsey tree covers of a doubling planar metric space.

6.1 Lower Bound on Ramsey Tree Cover

In this section we show an asymptotically tight lower bound on Ramsey tree covers, in the regime where the number of trees is small. The lower bound on high girth graphs mentioned above can only give distortion $\Omega(\log_k n)$ for tree covers with k trees. Here we use a different example (which is a planar metric with $O(1)$ doubling dimension) that strengthen the lower bound on the distortion to $\Omega(n^{1/k})$.

We will need the following notion of a composition of metric spaces that was introduced in [15] (we present here a simplification of the original definition).

▶ **Definition 26.** *Let (S, d_S), (T, d_T) be finite metric spaces. For $\beta \geq 1/2$, the β-composition of S with T, denoted by $Z = S_\beta[T]$, is a metric space of size $|Z| = |S| \cdot |T|$ constructed by replacing each point $u \in S$ with a copy of T, denoted by $T^{(u)}$. Let $\gamma = \frac{\max_{t \neq t' \in T}\{d_T(t,t')\}}{\min_{s \neq s' \in S}\{d_S(s,s')\}}$. For $z_i \neq z_j \in Z$ such that $z_i \in T^{(u)}$ and $z_j \in T^{(v)}$ the distance is defined as follows: if $u = v$, then $d_Z(z_i, z_j) = \frac{1}{\beta\gamma} \cdot d_T(z_i, z_j)$, otherwise (if $u \neq v$), $d_Z(z_i, z_j) = d_S(u, v)$.*

It is easily checked that the choice of the factor $1/(\beta\gamma)$ guarantees that d_Z is indeed a metric. For a finite metric space S and an integer $t \geq 1$, let $[S]^t_\beta$ denote the metric space obtained by β-composition of S with itself t times. The following theorem asserts that when the number of trees is small our upper bound on the distortion of Ramsey tree covers is tight up to a logarithmic factor. Although the example is not described as a planar and doubling metric space, in the full version of the paper we show that it can be approximated with constant distortion by a shortest path metric on a series-parallel graph with constant doubling dimension.

▶ **Theorem 27.** *For any $k \geq 1$ and large enough n, there is an n-point doubling metric space X, such that any Ramsey tree cover of X of size k, has distortion $\Omega\left(n^{\frac{1}{k}}\right)$.*

Proof. Let C_N denote the shortest path metric on the unweighted N-point cycle graph. For any integers $k, N \geq 1$ and for any $\beta \geq 1/2$, consider the metric space $Z_k(N) = [C_N]^k_\beta$. We prove by induction on k, that any Ramsey tree cover of $Z_k(N)$ with k trees has distortion at least $\frac{1}{3}|Z_k(N)|^{1/k} - 1$. The details are differed to the full version of the paper. ◀

References

1 Ittai Abraham, Yair Bartal, and Ofer Neiman. Advances in metric embedding theory. *Advances in Mathematics*, 228(6):3026–3126, 2011. doi:10.1016/j.aim.2011.08.003.

2 Ittai Abraham, Yair Bartal, and Ofer Neiman. Embedding Metrics into Ultrametrics and Graphs into Spanning Trees with Constant Average Distortion. *SIAM J. Comput.*, 44(1):160–192, 2015. doi:10.1137/120884390.

3 Ittai Abraham, Shiri Chechik, Michael Elkin, Arnold Filtser, and Ofer Neiman. Ramsey Spanning Trees and their Applications. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1650–1664, 2018. doi:10.1137/1.9781611975031.108.

4 Ittai Abraham and Cyril Gavoille. Object location using path separators. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing, PODC 2006, Denver, CO, USA, July 23-26, 2006*, pages 188–197, 2006. doi:10.1145/1146381.1146411.

5 Ittai Abraham and Ofer Neiman. Using petal-decompositions to build a low stretch spanning tree. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 395–406, 2012. doi:10.1145/2213977.2214015.

6 Noga Alon, Richard M. Karp, David Peleg, and Douglas West. A Graph-Theoretic Game and its Application to the k-Server Problem. *SIAM J. Comput.*, 24(1):78–100, 1995.

7 Ingo Althöfer, Gautam Das, David Dobkin, and Deborah Joseph. Generating sparse spanners for weighted graphs. In John R. Gilbert and Rolf Karlsson, editors, *SWAT 90*, pages 26–37, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

8 S. Arya, G. Das, D. M. Mount, J. S. Salowe, and M. H. M. Smid. Euclidean spanners: short, thin, and lanky. In *Proc. of 27th STOC*, pages 489–498, 1995.

9 Baruch Awerbuch. Complexity of network synchronization. *J. ACM*, 32(4):804–823, October 1985. doi:10.1145/4221.4227.

10 Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg. Near-Linear Cost Sequential and Distribured Constructions of Sparse Neighborhood Covers. In *34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993*, pages 638–647, 1993. doi:10.1109/SFCS.1993.366823.

11 Yair Bartal. Probabilistic Approximations of Metric Spaces and Its Algorithmic Applications. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 184–193, 1996.

12 Yair Bartal. On Approximating Arbitrary Metrices by Tree Metrics. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 161–168, 1998. doi:10.1145/276698.276725.

13 Yair Bartal. Graph Decomposition Lemmas and their Role in Metric Embedding Methods. In *12th Annual European Symposium on Algorithms*, pages 89–97, 2004. URL: http://springerlink.metapress.com/openurl.asp?genre=article&issn=0302-9743&volume=3221&spage=89.

14 Yair Bartal. Lecture notes. In *Metric embedding theory and its algorithmic applications course*, 2011. URL: http://moodle.cs.huji.ac.il/cs10/file.php/67720/GM_Lecture6.pdf.

15 Yair Bartal, Nathan Linial, Manor Mendel, and Assaf Naor. On metric ramsey-type phenomena. *Annals Math*, 162(2):643–709, 2005.

16 S. Baswana and S. Sen. A Simple Linear Time algorithm for Computing a $(2k-1)$-Spanner of $O(n^{1+1/k})$ Size in Weighted Graphs. In *Proceedings of the 30th International Colloquium on Automata, Languages and Programming*, volume 2719 of *LNCS*, pages 384–396. Springer, 2003.

17 Guy E. Blelloch, Yan Gu, and Yihan Sun. A New Efficient Construction on Probabilistic Tree Embeddings. *CoRR*, abs/1605.04651, 2016. arXiv:1605.04651.

18 Béla Bollobas. *Extremal Graph Theory*. Dover Publications, Inc., New York, NY, USA, 2004.

19 T-H. Hubert Chan, Anupam Gupta, Bruce M. Maggs, and Shuheng Zhou. On Hierarchical routing in Doubling Metrics. In *Proc. 16th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2005.

20 B. Chandra, G. Das, G. Narasimhan, and J. Soares. New sparseness results on graph spanners. *Int. J. Comput. Geometry Appl.*, 5:125–144, 1995.

21 Moses Charikar, Chandra Chekuri, Ashish Goel, Sudipto Guha, and Serge Plotkin. Approximating a Finite Metric by a Small Number of Tree Metrics. In *FOCS '98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, page 379, Washington, DC, USA, 1998. IEEE Computer Society.

22 Edith Cohen. Fast algorithms for constructing t-spanners and paths with stretch t. In *34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993*, pages 648–658, 1993. doi:10.1109/SFCS.1993.366822.

23 Michael Elkin, Yuval Emek, Daniel A. Spielman, and Shang-Hua Teng. Lower-stretch spanning trees. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 494–503, New York, NY, USA, 2005. ACM Press. doi:10.1145/1060590.1060665.

24 Michael Elkin and David Peleg. (1+epsilon, beta)-Spanner Constructions for General Graphs. *SIAM J. Comput.*, 33(3):608–631, 2004. doi:10.1137/S0097539701393384.

25 Michael Elkin and Seth Pettie. A Linear-Size Logarithmic Stretch Path-Reporting Distance Oracle for General Graphs. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 805–821, 2015. doi:10.1137/1.9781611973730.55.

26 P. Erdős and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. *Infinite and finite sets*, 10(2):609–â€“627, 1975.

27 Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, November 2004. doi:10.1016/j.jcss.2004.04.011.

28 Arnold Filtser and Ofer Neiman. Light Spanners for High Dimensional Norms via Stochastic Decompositions. In *26th Annual European Symposium on Algorithms*, 2018.

29 Jie Gao, Leonidas J. Guibas, and An Nguyen. Deformable Spanners and Applications. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, pages 190–199, New York, NY, USA, 2004. ACM. doi:10.1145/997817.997848.

30 Anupam Gupta, Mohammad T. Hajiaghayi, and Harald Räcke. Oblivious Network Design. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, pages 970–979, Philadelphia, PA, USA, 2006. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=1109557.1109665.

31 Anupam Gupta, Amit Kumar, and Rajeev Rastogi. Traveling with a Pez Dispenser (or, Routing Issues in MPLS). *SIAM J. Comput.*, 34(2):453–474, 2004. doi:10.1137/S0097539702409927.

32 Sariel Har-Peled and Manor Mendel. Fast Construction of Nets in Low-Dimensional Metrics and Their Applications. *SIAM J. Comput*, 35(5):1148–1184, 2006.

33 Philip Klein, Serge A. Plotkin, and Satish Rao. Excluded minors, network decomposition, and multicommodity flow. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, STOC '93, pages 682–690, New York, NY, USA, 1993. ACM. doi:10.1145/167088.167261.

34 Philip N. Klein. Preprocessing an undirected planar network to enable fast approximate distance queries. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA.*, pages 820–827, 2002. URL: http://dl.acm.org/citation.cfm?id=545381.545488.

35 Robert Krauthgamer, James R. Lee, Manor Mendel, and Assaf Naor. Measured descent: A new embedding method for finite metrics. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 434–443. IEEE, October 2004.

36 Nathan Linial and Michael Saks. Decomposing graphs into regions of small diameter. In *SODA '91: Proceedings of the second annual ACM-SIAM symposium on Discrete algorithms*, pages 320–330, Philadelphia, PA, USA, 1991. Society for Industrial and Applied Mathematics.

37 Richard J. Lipton and Robert Endre Tarjan. Applications of a Planar Separator Theorem. *SIAM J. Comput.*, 9(3):615–627, 1980. doi:10.1137/0209046.

38 Manor Mendel and Assaf Naor. Ramsey partitions and proximity data structures. *Journal of the European Mathematical Society*, 9(2):253–275, 2007.

39 Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovasz Local Lemma. *CoRR*, abs/0903.0544, 2009. arXiv:0903.0544.

40 Assaf Naor and Terence Tao. Scale-oblivious metric fragmentation and the nonlinear Dvoretzky theorem. *Israel Journal of Mathematics*, 192(1):489–504, 2012. doi:10.1007/s11856-012-0039-7.

41 G. Narasimhan and M. Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007.

42 D. Peleg and A. Schäffer. Graph Spanners. *J. Graph Theory*, 13:99–116, 1989.

43 D. Peleg and E. Upfal. A tradeoff between size and efficiency for routing tables. *J. of the ACM*, 36:510–530, 1989.

44 Yuri Rabinovich and Ran Raz. Lower Bounds on the Distortion of Embedding Finite Metric Spaces in Graphs. *Discrete & Computational Geometry*, 19(1):79–94, 1998. URL: http://link.springer.de/link/service/journals/00454/bibs/19n1p79.html.

45 Satish Rao and Warren D. Smith. Approximating Geometrical Graphs via "Spanners" and "Banyans". In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 540–550, 1998. doi:10.1145/276698.276868.

46 M. Thorup and U. Zwick. Approximate Distance Oracles. In 33^{rd} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 183–192, Hersonissos, Crete, Greece, July 2001.

47 M. Thorup and U. Zwick. Spanners and emulators with sublinear distance errors. In *Proc. of Symp. on Discr. Algorithms*, pages 802–809, 2006.

48 Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, 2004. doi:10.1145/1039488.1039493.

Even Faster Elastic-Degenerate String Matching via Fast Matrix Multiplication

Giulia Bernardini
Department of Informatics, Systems and Communication, University of Milano - Bicocca, Italy
giulia.bernardini@unimib.it

Paweł Gawrychowski
Institute of Computer Science, University of Wrocław, Poland
gawry@cs.uni.wroc.pl

Nadia Pisanti
Department of Computer Science, University of Pisa, Italy
ERABLE Team, INRIA, France
pisanti@di.unipi.it

Solon P. Pissis
CWI, Amsterdam, The Netherlands
solon.pissis@cwi.nl

Giovanna Rosone
Department of Computer Science, University of Pisa, Italy
giovanna.rosone@unipi.it

Abstract

An elastic-degenerate (ED) string is a sequence of n sets of strings of total length N, which was recently proposed to model a set of similar sequences. The ED string matching (EDSM) problem is to find all occurrences of a pattern of length m in an ED text. The EDSM problem has recently received some attention in the combinatorial pattern matching community, and an $\mathcal{O}(nm^{1.5}\sqrt{\log m}+N)$-time algorithm is known [Aoyama et al., CPM 2018]. The standard assumption in the prior work on this question is that N is substantially larger than both n and m, and thus we would like to have a linear dependency on the former. Under this assumption, the natural open problem is whether we can decrease the 1.5 exponent in the time complexity, similarly as in the related (but, to the best of our knowledge, not equivalent) *word break* problem [Backurs and Indyk, FOCS 2016].

Our starting point is a conditional lower bound for the EDSM problem. We use the popular combinatorial Boolean matrix multiplication (BMM) conjecture stating that there is no truly subcubic *combinatorial* algorithm for BMM [Abboud and Williams, FOCS 2014]. By designing an appropriate reduction we show that a combinatorial algorithm solving the EDSM problem in $\mathcal{O}(nm^{1.5-\epsilon}+N)$ time, for any $\epsilon > 0$, refutes this conjecture. Of course, the notion of combinatorial algorithms is not clearly defined, so our reduction should be understood as an indication that decreasing the exponent requires fast matrix multiplication.

Two standard tools used in algorithms on strings are string periodicity and fast Fourier transform. Our main technical contribution is that we successfully combine these tools with fast matrix multiplication to design a non-combinatorial $\mathcal{O}(nm^{1.381}+N)$-time algorithm for EDSM. To the best of our knowledge, we are the first to do so.

2012 ACM Subject Classification Theory of computation → Pattern matching

Keywords and phrases string algorithms, pattern matching, elastic-degenerate string, matrix multiplication, fast Fourier transform

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.21

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1905.02298`.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 21; pp. 21:1–21:15

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

Funding GR and NP are partially supported by MIUR-SIR project CMACBioSeq "Combinatorial methods for analysis and compression of biological sequences" grant n. RBSI146R5L.

1 Introduction

Boolean matrix multiplication (BMM) is one of the most fundamental computational problems. Apart from its theoretical interest, it has a wide range of applications [37, 52, 29, 27, 46]. BMM is also the core combinatorial part of integer matrix multiplication. In both problems, we are given two $\mathcal{N} \times \mathcal{N}$ matrices and we are to compute $\mathcal{N}^2$ values. Integer matrix multiplication can be performed in *truly subcubic* time, *i.e.*, in $\mathcal{O}(\mathcal{N}^{3-\epsilon})$ operations over the field, for some $\epsilon > 0$. The fastest known algorithms for this problem run in $\mathcal{O}(\mathcal{N}^{2.373})$ time [30, 54]. These algorithms are known as *algebraic*: they rely on the underlying ring structure.

There also exists a different family of algorithms for the BMM problem known as *combinatorial*. Their focus is on unveiling the combinatorial structure in the Boolean matrices to reduce redundant computations. A series of results [7, 9, 15] culminating in an $\hat{\mathcal{O}}(\mathcal{N}^3/\log^4 \mathcal{N})$-time algorithm [58] (the $\hat{\mathcal{O}}(\cdot)$ notation suppresses poly(log log) factors) has led to the popular *combinatorial BMM conjecture* stating that there is no combinatorial algorithm for BMM working in time $\mathcal{O}(\mathcal{N}^{3-\epsilon})$, for any $\epsilon > 0$ [2]. There has been ample work on applying this conjecture to obtain *BMM hardness* results: see, *e.g.*, [44, 2, 49, 33, 43, 42, 17].

String matching is another fundamental problem. The problem is to find all fragments of a string *text* of length n that match a string *pattern* of length m. This problem has several linear-time solutions [22]. In many real-world applications, it is often the case that letters at some positions are either unknown or uncertain. A way of representing these positions is with a subset of the alphabet Σ. Such a representation is called *degenerate string*. The first efficient algorithm for a degenerate text and a standard pattern was published by Fischer and Paterson in 1974 [28]. It has undergone several improvements since then [36, 39, 20, 19]. The first efficient algorithm for a degenerate pattern and a standard text was published by Abrahamson in 1987 [3], followed by several practically efficient algorithms [57, 47, 34].

Degenerate letters are used in the IUPAC notation [38] to represent a position in a DNA sequence that can have multiple possible alternatives. These are used to encode the consensus of a population of sequences [21, 4] in a multiple sequence alignment (MSA). In the presence of insertions or deletions in the MSA, we may need to consider alternative representations. Consider the following MSA of three closely-related sequences (on the left):

```
GCAACGGGTA--TT
GCAACGGGTATATT
GCACCTGG----TT
```

$$\tilde{T} = \{\ \texttt{GCA}\ \} \cdot \left\{ \begin{array}{c} \texttt{A} \\ \texttt{C} \end{array} \right\} \cdot \{\ \texttt{C}\ \} \cdot \left\{ \begin{array}{c} \texttt{G} \\ \texttt{T} \end{array} \right\} \cdot \{\ \texttt{GG}\ \} \cdot \left\{ \begin{array}{c} \texttt{TA} \\ \texttt{TATA} \\ \varepsilon \end{array} \right\} \cdot \{\ \texttt{TT}\ \}$$

These sequences can be compacted into a single sequence $\tilde{T}$ of sets of strings (on the right) containing some deterministic and some *non-deterministic* segments. A non-deterministic segment is a finite set of deterministic strings and may contain the empty string ε corresponding to a deletion. The total number of segments is the *length* of $\tilde{T}$ and the total number of letters is the *size* of $\tilde{T}$. We denote the length by $n = |\tilde{T}|$ and the size by $N = ||\tilde{T}||$.

This representation has been defined in [35] by Iliopoulos et al. as an *elastic-degenerate* (ED) string. Being a sequence of subsets of Σ^*, it can be seen as a generalization of a degenerate string. The natural problem that arises is finding all matches of a deterministic pattern P in an ED text $\tilde{T}$. This is the *elastic-degenerate string matching* (EDSM) problem. Since its introduction in 2017 [35], it has attracted some attention in the combinatorial pattern matching community, and a series of results have been published. The simple

algorithm by Iliopoulos et al. [35] for EDSM was first improved by Grossi et al. in the same year, who showed that, for a pattern of length m, the EDSM problem can be solved *on-line* in $\mathcal{O}(nm^2 + N)$ time [32]; on-line means that the text is read segment-by-segment and an occurrence is detected as soon as possible. This result was improved by Aoyama et al. [6] who presented an $\mathcal{O}(nm^{1.5}\sqrt{\log m} + N)$-time algorithm. An important feature of these bounds is their *linear dependency* on N. A different branch of on-line algorithms waiving the linear-dependency restriction exists [32, 48, 18]. Moreover, the EDSM problem has been considered under Hamming and edit distance [12].

A question with a somewhat similar flavor is the *word break* problem. We are given a dictionary $\mathcal{D}$, $m = ||\mathcal{D}||$, and a string S, $n = |S|$, and the question is whether we can split S into fragments that appear in $\mathcal{D}$ (the same element of $\mathcal{D}$ can be used multiple times). Backurs and Indyk [8] designed an $\tilde{\mathcal{O}}(nm^{1/2-1/18}+m)$-time algorithm for this problem (the $\tilde{\mathcal{O}}$ notation suppresses poly(log) factors). Bringmann et al. [14] improved this to $\tilde{\mathcal{O}}(nm^{1/3} + m)$ and showed that this is optimal for combinatorial algorithms by a reduction from k-Clique. Their algorithm uses fast Fourier transform (FFT), and so it is not clear whether it should be considered combinatorial. While this problem seems similar to EDSM, there does not seem to be a direct reduction and so their lower bound does not immediately apply.

Our Results. It is known that BMM and triangle detection in graphs either both have truly subcubic combinatorial algorithms or none of them do [56]. Recall also that the currently fastest algorithm with linear dependency on N for the EDSM problem runs in $\mathcal{O}(nm^{1.5}\sqrt{\log m} + N)$ time [6]. In this paper we prove the following two theorems.

▶ **Theorem 1.** *If the EDSM problem can be solved in $\mathcal{O}(nm^{1.5-\epsilon} + N)$ time, for any $\epsilon > 0$, with a combinatorial algorithm, then there exists a truly subcubic combinatorial algorithm for triangle detection.*

Arguably, the notion of combinatorial algorithms is not clearly defined, and Theorem 1 should be understood as an indication that in order to achieve a better complexity one should use fast matrix multiplication. Indeed, there are examples where a lower bound conditioned on BMM was helpful in constructing efficient algorithms using fast matrix multiplication [1, 16, 13, 45, 24, 55, 59]. We successfully design such a non-combinatorial algorithm by combining three ingredients: a string periodicity argument, FFT, and fast matrix multiplication. While periodicity is the usual tool in combinatorial pattern matching [40, 23, 41] and using FFT is also not unusual (for example, it often shows up in approximate string matching [3, 5, 19, 31]), to the best of our knowledge, we are the first to combine these with fast matrix multiplication. Specifically, we show the following result for the EDSM problem.

▶ **Theorem 2.** *The EDSM problem can be solved on-line in expected $\mathcal{O}(nm^{1.381} + N)$ time.*

An important building block in our solution that might find applications in other problems is a method of selecting a small set of length-ℓ substrings of the pattern, called *anchors*, so that any relevant occurrence of a string from an ED text set contains at least one but not too many such anchors inside. This is obtained by rephrasing the question in a graph-theoretical language and then generalizing the well-known fact that an instance of the *hitting set* problem with m sets over $[n]$, each of size at least k, has a solution of size $\mathcal{O}(n/k \cdot \log m)$. While the idea of carefully selecting some substrings of the same length is not new, for example Kociumaka et al. [41] used it to design a data structure for pattern matching queries on a string, our setting is different and hence so is the method of selecting these substrings.

Roadmap. Section 2 provides the necessary definitions and notation as well the algorithmic toolbox used throughout the paper. In Section 3 we prove our hardness result for the EDSM problem (Theorem 1). In Section 4 we present our algorithm for the same problem (Theorem 2); this is the most technically involved part of the paper.

2 Preliminaries

Let $T = T[1]T[2]\dots T[n]$ be a *string* of length $|T| = n$ over a finite ordered alphabet Σ of size $|\Sigma| = \sigma$. For two positions i and j on T, we denote by $T[i\,..\,j] = T[i]\dots T[j]$ the *substring* of T that starts at position i and ends at position j (it is of length 0 if $j < i$). By ε we denote the *empty string* of length 0. A *prefix* of T is a substring of the form $T[1\,..\,j]$, and a suffix of T is a substring of the form $T[i\,..\,n]$. T^r denotes the *reverse* of T, that is, $T[n]T[n-1]\dots T[1]$. We say that a string X is a *power* of a string Y if there exists an integer $k > 1$, such that X is expressed as k consecutive concatenations of Y, denoted by $X = Y^k$. A period of a string X is any integer $p \in [1, |X|]$ such that $X[i] = X[i+p]$ for every $i = 1, 2, \dots, |X| - p$, and *the period*, denoted by $\mathrm{per}(X)$, is the smallest such p. We call a string X *strongly periodic* if $\mathrm{per}(X) \le |X|/4$.

▶ **Lemma 3** ([26]). *If p and q are both periods of the same string X, and additionally $p + q \le |X| + 1$, then $\gcd(p, q)$ is also a period of X.*

A *trie* is a rooted tree in which every edge is labeled with a single letter, and every two edges outgoing from the same node have different labels. The label of a node u in such a tree T, denoted by $\mathcal{L}(u)$, is defined as the concatenation of the labels of all the edges on the path from the root of T to u. Thus, the label of the root of T is ε, and a trie is a representation of a set of strings consisting of the labels of all its leaves. By replacing each path p consisting of nodes with exactly one child by an edge labeled by the concatenation of the labels of the edges of p we obtain a *compact trie*. The nodes of the trie that are removed after this transformation are called *implicit*, while the remaining ones are referred to as *explicit*. The *suffix tree* of a string S is the compact trie representing all suffixes of $S\$$, $\$ \notin \Sigma$, where instead of explicitly storing the label $S[i\,..\,j]$ of an edge we represent it by a pair (i, j).

A *heavy path decomposition* of a tree T is obtained by selecting, for every non-leaf node $u \in T$, its child v such that the subtree rooted at v is the largest. This decomposes the nodes of T into node-disjoint paths, with each such path p (called a heavy path) starting at some node, called the *head* of p, and ending at a leaf. An important property of such a decomposition is that the number of distinct heavy paths above any leaf (that is, intersecting the path from a leaf to the root) is only logarithmic in the size of T [51].

Let $\tilde{\Sigma}$ denote the set of all finite non-empty subsets of Σ^*. Previous works (cf. [35, 32, 12, 6, 48]) define $\tilde{\Sigma}$ as the set of all finite non-empty subsets of Σ^* excluding $\{\varepsilon\}$ but we waive here the latter restriction as it has no algorithmic implications. An *elastic-degenerate string*, or ED string, over alphabet Σ, is a string over $\tilde{\Sigma}$, i.e., an ED string is an element of $\tilde{\Sigma}^*$.

Let $\tilde{T}$ denote an ED string of *length* n, i.e. $|\tilde{T}| = n$. We assume that for any $1 \le i \le n$, the set $\tilde{T}[i]$ is implemented as an array and can be accessed by an index, i.e., $\tilde{T}[i] = \{\tilde{T}[i][k] \mid k = 1, \dots, |\tilde{T}[i]|\}$. For any $\tilde{c} \in \tilde{\Sigma}$, $||\tilde{c}||$ denotes the total length of all strings in $\tilde{c}$, and for any ED string $\tilde{T}$, $||\tilde{T}||$ denotes the total length of all strings in all $\tilde{T}[i]$s or the *size* of $\tilde{T}$, i.e., $||\tilde{c}|| = \sum_{s\in\tilde{c}} |s|$ and $||\tilde{T}|| = \sum_{i=1}^{n} ||\tilde{T}[i]||$. An ED string $\tilde{T}$ can be thought of as a representation of the set of strings $\mathcal{A}(\tilde{T}) = \tilde{T}[1] \times \dots \times \tilde{T}[n]$, where $A \times B = \{xy \mid x \in A, y \in B\}$ for any sets of strings A and B. For any ED string $\tilde{X}$ and a pattern P, we say that P *matches* $\tilde{X}$ if

1. $|\tilde{X}| = 1$ and P is a substring of some string in $\tilde{X}[1]$, or,
2. $|\tilde{X}| > 1$ and $P = P_1 \dots P_{|\tilde{X}|}$, where P_1 is a suffix of some string in $\tilde{X}[1]$, $P_{|\tilde{X}|}$ is a prefix of some string in $\tilde{X}[|\tilde{X}|]$, and $P_i \in \tilde{X}[i]$, for all $1 < i < |\tilde{X}|$.

We say that an *occurrence* of a string P *ends* at position j of an ED string $\tilde{T}$ if there exists $i \leq j$ such that P matches $\tilde{T}[i] \dots \tilde{T}[j]$. We will refer to string P as the *pattern* and to ED string $\tilde{T}$ as the *text*. We define the main problem considered in this paper.

Elastic-Degenerate String Matching (EDSM)
INPUT: A string P of length m and an ED string $\tilde{T}$ of length n and size $N \geq m$.
OUTPUT: All positions in $\tilde{T}$ where at least one occurrence of P ends.

▶ **Example 4.** Pattern $P = \texttt{GTAT}$ ends at positions 2, 6, and 7 of the following text $\tilde{T}$.

$$\tilde{T} = \{\ \texttt{ATGTA}\ \} \cdot \left\{ \begin{matrix} \texttt{A} \\ \texttt{T} \end{matrix} \right\} \cdot \{\ \texttt{C}\ \} \cdot \left\{ \begin{matrix} \texttt{G} \\ \texttt{T} \end{matrix} \right\} \cdot \{\ \texttt{CG}\ \} \cdot \left\{ \begin{matrix} \texttt{TA} \\ \texttt{TATA} \\ \varepsilon \end{matrix} \right\} \cdot \left\{ \begin{matrix} \texttt{TATGC} \\ \texttt{TTTTA} \end{matrix} \right\}$$

Aoyama et al. [6] obtained an on-line $\mathcal{O}(nm^{1.5}\sqrt{\log m} + N)$-time algorithm by designing an efficient solution for the following problem.

Active Prefixes (AP)
INPUT: A string P of length m, a bit vector U of size m, a set $\mathcal{S}$ of strings of total length N.
OUTPUT: A bit vector V of size m with $V[j] = 1$ if and only if there exists $S \in \mathcal{S}$ and $i \in [1, m], U[i] = 1$, such that $P[1\,..\,i] \cdot S = P[1\,..\,i + |S|]$ and $j = i + |S|$.

In more detail, given an ED text one should consider an instance of the AP problem per segment. Hence, an $\mathcal{O}(f(m) + N_i)$ solution for AP (with N_i being the size of the i-th segment of the ED text) implies an $\mathcal{O}(n \cdot f(m) + N)$ solution for EDSM, as $N = \sum_{i=1}^{n} N_i$. We provide an example of the AP problem.

▶ **Example 5.** Let $P = \texttt{ababbababab}$ of length $m = 11$, $U = 01000100000$, and $\mathcal{S} = \{\varepsilon, \texttt{ab}, \texttt{abb}, \texttt{ba}, \texttt{baba}\}$. We have that $V = 01011101010$.

For our hardness results we rely on BMM and the following closely related problem.

Boolean Matrix Multiplication (BMM)
INPUT: Two $\mathcal{N} \times \mathcal{N}$ Boolean matrices A and B.
OUTPUT: $\mathcal{N} \times \mathcal{N}$ Boolean matrix C, where $C[i, j] = \bigvee_k (A[i, k] \wedge B[k, j])$.

Triangle Detection (TD)
INPUT: Three $\mathcal{N} \times \mathcal{N}$ Boolean matrices A, B and C.
OUTPUT: Are there i, j, k such that $A[i, j] = B[j, k] = C[k, i] = 1$?

An algorithm is called *truly subcubic* if it runs in $\mathcal{O}(\mathcal{N}^{3-\epsilon})$ time, for some $\epsilon > 0$. TD and BMM either both have truly subcubic combinatorial algorithms, or none of them do [56].

3 EDSM Conditional Lower Bound

We show a conditional lower bound for the EDSM problem. Specifically, we show that TD can be reduced to the EDSM problem. We work with the decision version: the goal is to detect whether there exists at least one occurrence of P in $\tilde{T}$.

▶ **Theorem 1.** *If the EDSM problem can be solved in $\mathcal{O}(nm^{1.5-\epsilon}+N)$ time, for any $\epsilon>0$, with a combinatorial algorithm, then there exists a truly subcubic combinatorial algorithm for triangle detection.*

Proof. Consider an instance of TD, where we are given three $\mathcal{N}\times\mathcal{N}$ Boolean matrices A, B, C, and the question is to check if there exist i, j, k such that $A[i, j] = B[j, k] = C[k, i] = 1$. Let s be a parameter to be determined later that corresponds to decomposing B into blocks of size $(\mathcal{N}/s)\times(\mathcal{N}/s)$. We reduce to an instance of EDSM over an alphabet Σ of size $\mathcal{O}(\mathcal{N})$.

Pattern P. We construct P by concatenating, in some fixed order, the following strings:

$$P(i,x,y) = v(i)xa^{\mathcal{N}/s}x\$\$ya^{\mathcal{N}/s}yv(i)$$

for every $i=1,2,\ldots,\mathcal{N}$ and $x,y=1,2,\ldots,s$, where $a\in\Sigma_1$, $\$\in\Sigma_2$, $x\in\Sigma_3$, $y\in\Sigma_4$, $v(i)\in\Sigma_5$, and $\Sigma_1,\Sigma_2,\ldots,\Sigma_5$ are disjoint subsets of Σ.

ED text $\tilde{T}$. The text $\tilde{T}$ consists of three parts. Its middle part encodes all the entries equal to 1 in matrices A, B and C, and consists of three string sets $\mathcal{X}=\mathcal{X}_1\cdot\mathcal{X}_2\cdot\mathcal{X}_3$, where:

1. $\mathcal{X}_1$ contains all strings of the form $v(i)xa^j$, for some $i=1,2,\ldots,\mathcal{N}$, $x=1,2,\ldots,s$ and $j=1,2,\ldots,\mathcal{N}/s$ such that $A[i,(x-1)\cdot(\mathcal{N}/s)+j]=1$;
2. $\mathcal{X}_2$ contains all strings of the form $a^{\mathcal{N}/s-j}\ x\$\$ya^{\mathcal{N}/s-k}$, for some $x,y=1,2,\ldots,s$ and $j,k=1,2,\ldots,\mathcal{N}/s$ such that $B[(x-1)\cdot(\mathcal{N}/s)+j,(y-1)\cdot(\mathcal{N}/s)+k]=1$, i.e., if the corresponding entry of B is 1;
3. $\mathcal{X}_3$ contains all strings of the form $a^kyv(i)$, for some $i=1,2,\ldots,\mathcal{N}$, $y=1,2,\ldots,s$ and $k=1,2,\ldots,\mathcal{N}/s$ such that $C[(y-1)\cdot(\mathcal{N}/s)+k,i]=1$.

It is easy to see that $|P(i,x,y)|=\mathcal{O}(\mathcal{N}/s)$. This implies the following:

1. The length of the pattern is $m=\mathcal{O}(\mathcal{N}\cdot s^2\cdot\mathcal{N}/s)=\mathcal{O}(\mathcal{N}^2\cdot s)$;
2. The size of $\mathcal{X}$ is $||\mathcal{X}||=\mathcal{O}(\mathcal{N}\cdot s\cdot\mathcal{N}/s\cdot\mathcal{N}/s+s^2\cdot(\mathcal{N}/s)^2\cdot\mathcal{N}/s+\mathcal{N}\cdot s\cdot\mathcal{N}/s\cdot\mathcal{N}/s)=\mathcal{O}(\mathcal{N}^3/s)$.

By the above construction, we obtain the following fact.

▶ **Fact 6.** $P(i,x,y)$ matches $\mathcal{X}$ if and only if the following holds for some $j,k=1,2,\ldots,\mathcal{N}/s$:

$$A[i,(x-1)\cdot(\mathcal{N}/s)+j]=B[(x-1)\cdot(\mathcal{N}/s)+j,(y-1)\cdot(\mathcal{N}/s)+k]=C[(y-1)\cdot(\mathcal{N}/s)+k,i]=1.$$

Solving the TD problem thus reduces to taking the disjunction of all such conditions. Let us write down all strings $P(i,x,y)$ in some arbitrary but fixed order to obtain $P=P_1P_2\ldots P_z$ with $z=\mathcal{N}s^2$, where every $P_t=P(i,x,y)$, for some i,x,y. We aim to construct a small number of sets of strings that, when considered as an ED text, match any prefix $P_1P_2\ldots P_t$ of the pattern, $1\le t\le z-1$; a similar construction can be carried on to obtain sets of strings that match any suffix $P_k\ldots P_{z-1}P_z$, $2\le k\le z$. These sets will then be added to the left and to the right of $\mathcal{X}$, respectively, to obtain the ED text $\tilde{T}$.

ED Prefix. We construct $\log z$ sets of strings as follows. The first one contains the empty string ε and $P_1P_2\ldots P_{z/2}$. The second one contains ε, $P_1P_2\ldots P_{z/4}$ and $P_{z/2+1}\ldots P_{z/2+z/4}$. The third one contains ε, $P_1P_2\ldots P_{z/8}$, $P_{z/4+1}\ldots P_{z/4+z/8}$, $P_{z/2+1}\ldots P_{z/2+z/8}$ and $P_{z/2+z/4+1}\ldots P_{z/2+z/4+z/8}$. Formally, for every $i=1,2,\ldots,\log z$, the i-th of such sets is:

$$\tilde{T}_i^p=\varepsilon\cup\{P_{j\frac{z}{2^{i-1}}+1}\ldots P_{j\frac{z}{2^{i-1}}+\frac{z}{2^i}}\mid j=0,1,\ldots,2^{i-1}-1\}.$$

ED Suffix. We similarly construct $\log z$ sets to be appended to $\mathcal{X}$:

$$\tilde{T}_i^s = \varepsilon \cup \{P_{z-j\frac{z}{2^{i-1}}-\frac{z}{2^i}+1} \dots P_{z-j\frac{z}{2^{i-1}}} \mid j = 0, 1, \dots, 2^{i-1}-1\}.$$

The total length of all the ED prefix and ED suffix strings is $\mathcal{O}(\log z \cdot \mathcal{N}^2 \cdot s) = \mathcal{O}(\mathcal{N}^2 \cdot s \cdot \log \mathcal{N})$. The whole ED text $\tilde{T}$ is: $\tilde{T} = \tilde{T}_1^p \cdot \dots \cdot \tilde{T}_{\log z}^p \cdot \mathcal{X} \cdot \tilde{T}_{\log z}^s \cdot \dots \cdot \tilde{T}_1^s$.

▶ **Lemma 7.** *The pattern P occurs in the ED text $\tilde{T}$ if and only if there exist i, j, k such that $A[i, j] = B[j, k] = C[k, i] = 1$.*

Proof. By Fact 6, if such i, j, k exist then P_t matches $\mathcal{X}$, for some $t \in \{1, \dots, z\}$. Then, by construction of the sets $\tilde{T}_i^p$ and $\tilde{T}_i^s$, the prefix $P_1 \dots P_{t-1}$ matches the ED prefix (this can be proved by induction), and similarly the suffix $P_{t+1} \dots P_z$ matches the ED suffix, so the whole P matches $\tilde{T}$, and so P occurs in $\tilde{T}$. Because of the letters \$ appearing only in the center of P_is and strings from $\mathcal{X}_2$, every P_i and a concatenation of $X_1 \in \mathcal{X}_1$, $X_2 \in \mathcal{X}_2$, $X_3 \in \mathcal{X}_3$ having the same length, and the P_is being distinct, there is an occurrence of the pattern P in $\tilde{T}$ if and only if $X_1X_2X_3 = P_t$ for some t and $X_1 \in \mathcal{X}_1$, $X_2 \in \mathcal{X}_2$, $X_3 \in \mathcal{X}_3$. But then, by Fact 6 there exists a triangle. ◀

Note that for the EDSM problem we have $m = \mathcal{N}^2 \cdot s$, $n = 1 + 2\log z$ and $N = ||\mathcal{X}|| + \mathcal{O}(\mathcal{N}^2 \cdot \dots \cdot \log \mathcal{N})$. Thus if we had a solution running in $\mathcal{O}(\log z \cdot m^{1.5-\epsilon} + ||\mathcal{X}|| + \mathcal{N}^2 \cdot s \cdot \log \mathcal{N}) = \mathcal{O}(\log \mathcal{N} \cdot (\mathcal{N}^2 \cdot s)^{1.5-\epsilon} + \mathcal{N}^3/s)$ time, for some $\epsilon > 0$, by choosing a sufficiently small $\alpha > 0$ and setting $s = \mathcal{N}^\alpha$ we would obtain an $\mathcal{O}(\mathcal{N}^{3-\delta})$-time algorithm for TD, for some $\delta > 0$. ◀

4 An $\mathcal{O}(nm^{1.381} + N)$-time Algorithm for EDSM

Our goal is to design a non-combinatorial $\mathcal{O}(nm^{1.381} + N)$-time algorithm for EDSM. It suffices to solve an instance of the AP problem in $\mathcal{O}(m^{1.381} + N)$ time. We further reduce the AP problem to a logarithmic number of restricted instances of the problem, in which the length of every string $S \in \mathcal{S}$ is in $[(10/9)^k, (10/9)^{k+1})$, for $k = 0, \dots, \log m / \log(10/9)$. If we solve every such instance in $\mathcal{O}(f(m) + N)$ time, then we can solve the original instance in $\mathcal{O}(f(m) \log m + N)$ time by taking the disjunction of results. We partition the strings in $\mathcal{S}$ into three types, compute the corresponding bit vector V for each type separately and in different ways, and, finally, take the disjunction to obtain the answer for the restricted instance.

Partitioning S. Let $\ell = 8/9 \cdot (10/9)^k$ (to avoid clutter we assume that ℓ is an integer, but this can be avoided by appropriately adjusting the constants), so that the length of every string in S belongs to $[9/8 \cdot \ell, 5/4 \cdot \ell)$. The three types of strings are as follows:

Type 1: Strings $S \in \mathcal{S}$ such that every length-ℓ substring of S is not strongly periodic.

Type 2: Strings $S \in \mathcal{S}$ containing at least one length-ℓ substring that is not strongly periodic and at least one length-ℓ substring that is strongly periodic.

Type 3: Strings $S \in \mathcal{S}$ such that every length-ℓ substring of S is strongly periodic (in Lemma 8 we show that in this case $\mathrm{per}(S) \le \ell/4$).

These three types are evidently a partition of $\mathcal{S}$ and, before we proceed with the algorithm, we need to show that we can determine the type of a string $S \in \mathcal{S}$ in $\mathcal{O}(|S|)$ time. We start with showing that, in fact, strings of type 3 are exactly strings with period at most $\ell/4$.

▶ **Lemma 8.** *Let S be a string. If $\mathrm{per}(S[j \,..\, j + \ell - 1]) \le \ell/4$ for every j then $\mathrm{per}(S) \le \ell/4$.*

▶ **Lemma 9.** *Given a string S we can determine its type in $\mathcal{O}(|S|)$ time.*

4.1 Type 1 Strings

In this section we show how to solve a restricted instance of the AP problem where every string $S \in \mathcal{S}$ is of type 1, that is, each of its length-ℓ substrings is not strongly periodic, and furthermore $|S| \in [9/8 \cdot \ell, 5/4 \cdot \ell)$ for some $\ell \leq m$. Observe that all (hence at most $1/4 \cdot \ell$) length-ℓ substrings of any $S \in \mathcal{S}$ must be distinct, as otherwise we would be able to find two occurrences of a length-ℓ substring at distance at most $1/4 \cdot \ell$ in S, making the period of the substring at most $1/4 \cdot \ell$ and contradicting the assumption that S is of type 1.

We start with constructing the suffix tree ST of P (our pattern in the EDSM problem) in $\mathcal{O}(m \log m)$ time [53] (note that we are spending $\mathcal{O}(m \log m)$ time and not just $\mathcal{O}(m)$ as to avoid any assumptions on the alphabet). For every explicit node $u \in ST$, we construct a perfect hash function mapping the first letter on every edge outgoing from u to the corresponding edge. This takes $\mathcal{O}(m \log m)$ time [50] and allows us to navigate in ST in constant time per letter. Then, for every $S \in \mathcal{S}$ we check if it occurs in P using the suffix tree in $\mathcal{O}(|S|)$ time, and if not disregard it from further consideration. We want to further partition $\mathcal{S}$ into $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_{\log m}$ that are processed separately. For every $\mathcal{S}_k$, we want to select a set of length-ℓ substrings of P, called the *anchors*, each represented by one of its occurrences in P, such that:

1. The total number of occurrences of all anchors in P is $\mathcal{O}(m/\ell \cdot \log^2 m)$.
2. For every $S \in \mathcal{S}_k$, at least one of its length-ℓ substrings is an anchor.
3. For every $S \in \mathcal{S}_k$, at most $\mathcal{O}(\log^2 m)$ of its length-ℓ substrings are anchors.

We formalize this using the following auxiliary problem, which is a strengthening of the *hitting set* problem: for any collection of m sets over $[n]$, each of size at least k, we can choose a subset of $[n]$ of size $\mathcal{O}(n/k \cdot \log m)$ that nontrivially intersects every set.

Node Selection (NS)
INPUT: A bipartite graph $G = (U, V, E)$ with $\deg(u) \in [d, \alpha \cdot d]$ for every $u \in U$.
OUTPUT: A set of $\mathcal{O}(|V|/d \cdot \log |U|)$ nodes from V such that every node in U has at least one selected neighbor but $\mathcal{O}(\alpha \cdot \log |U|)$ such selected neighbors.

To reduce finding anchors to an instance of the NS problem, we first build a bipartite graph G in which the nodes on the left correspond to strings $S \in \mathcal{S}$, the nodes on the right correspond to distinct length-ℓ substrings of P, and there is an edge connecting a node corresponding to a length-ℓ string H with a node corresponding to a string S when H occurs in S. Using suffix links, we can find the node of the suffix tree corresponding to every length-ℓ substring of S in $\mathcal{O}(|S|)$ total time, so the whole construction takes $\mathcal{O}(m \log m + \sum_{S \in \mathcal{S}} |S|) = \mathcal{O}(m \log m + N)$ time. The size of G is $\mathcal{O}(m + N)$, and the degree of every node on its left belongs to $[1/8 \cdot \ell, 1/4 \cdot \ell)$. We further partition G into a logarithmic number of graphs $G_0, G_1, \ldots, G_{\log m}$ where G_k contains all nodes v on the right of G such that the number of occurrences in P of the corresponding length-ℓ string belongs to $[2^k, 2^{k+1})$. For every node u on the left of G we find k such that at least $1/8 \cdot \ell / \log m$ of its neighbors exist in G_k, add u as a node on the left of G_k, and declare $\mathcal{S}_k$ to consist of all strings $S \in \mathcal{S}$ corresponding to nodes on the left of G_k. By construction, every $S \in \mathcal{S}$ corresponds to a node on the left of exactly one G_k, so we indeed obtain a partition of $\mathcal{S}$. For every $\mathcal{S}_k$ we solve the corresponding instance of the NS problem to obtain its corresponding set of anchors. We can assume that all strings in $\mathcal{S}_k$ are distinct, so there are at most m^2 nodes on the left of G_k, the degree of each such node belongs to $[1/8 \cdot \ell / \log m, 1/4 \cdot \ell]$ and, denoting by m_k the total number of occurrences in P of strings corresponding to nodes on the right of G_k, we have $\sum_k m_k \leq m$ and there are at most $m_k/2^k$ nodes on the right of G_k. At most

$\mathcal{O}((m_k/2^k)/(\ell/\log m)\cdot\log m)$ nodes on the right of G_k are designated as anchors, making the total number of occurrences of all anchors $\mathcal{O}(m/\ell\cdot\log^2 m)$. Also, every $S\in\mathcal{S}_k$ contains an occurrence of at least one anchor, and no more than $\mathcal{O}(\log^2 m)$ such occurrences.

It is not immediately clear that an instance of the NS problem always has a solution. We show that indeed it does, and that it can be efficiently found with a Las Vegas algorithm.

▶ **Lemma 10.** *A solution to an instance of the NS problem always exists and can be found in expected linear time.*

Proof. We independently choose each node of V with probability p to obtain the set X of selected nodes. Then, we check if the size of X is small enough, every node in U has at least one selected neighbor, and $\mathcal{O}(\alpha\cdot\log|U|)$ such selected neighbors. All these checks can be made in linear time in the size of the graph, so to show that a solution exists and can be found in expected linear time, it remains to show that we can adjust p to make the probability of failure equal to a constant less than 1. The expected size of X is obviously $p|V|$, so by Markov's inequality the probability that $|X|>4p|V|$ is at most $1/4$. The probability that a node in U has no neighbors in X is at most $(1-p)^d$. Thus, by union bound the probability that there exists at least one such node is at most $|U|\cdot(1-p)^d\le|U|\cdot e^{-pd}$. Consider a node in U of degree $d'\in[d,\alpha\cdot d]$. Its expected number of selected neighbors is pd'. Thus, by Chernoff's inequality the probability that its number of selected neighbors exceeds $(1+\delta)pd'$ is at most $e^{-\frac{\delta^2}{2+\delta}pd'}$. By setting $\delta=1$, we obtain that the probability of the number of selected neighbors exceeding $2p\alpha d$ is at most $e^{-\frac{1}{3}\alpha pd}$. By union bound, the probability that this happens for at least one node is at most $|U|\cdot e^{-\frac{1}{3}\alpha pd}$.

We choose $p=3\ln(4|U|)/d$. Then, the probability that the size of X exceeds $4p|V|=12\ln(4|U|)/d\cdot|V|=\mathcal{O}(\frac{|V|}{d}\cdot\log|U|)$ is at most $1/4$, the probability that there exists a node in U with no selected neighbor is at most $|U|\cdot e^{-pd}\le 1/4$, and the probability that there exists a node in U with more than $2p\alpha d=\mathcal{O}(\alpha\cdot\log|U|)$ selected neighbors is at most $|U|\cdot e^{-\frac{1}{3}\alpha pd}\le 1/4$, thus the overall probability of failure is at most $3/4$ as required. ◀

In the rest of this section we explain how to compute the bit vector V from the bit vector U after having obtained a set $\mathcal{A}$ of anchors for a set of strings $\mathcal{S}_k$ of total length N_k. For any $S\in\mathcal{S}_k$, since S contains an occurrence of at least one anchor $H\in\mathcal{A}$, for concreteness $S[j\,..\,(j+|H|-1)]=H$, any occurrence of S in P can be generated by choosing some occurrence of H in P, say $P[i\,..\,(i+|H|-1)]=H$, and then checking that $S[1\,..\,(j-1)]=P[(i-j+1)\,..\,(i-1)]$ and $S[(j+|H|)\,..\,|S|]=P[(i+|H|)\,..\,(i+|S|-j)]$. In other words, $S[1\,..\,(j-1)]$ should be a suffix of $P[1\,..\,(i-1)]$ and $S[(j+|H|)\,..\,|S|]$ should be a prefix of $P[(i+|H|)\,..\,|P|]$. In such case, we say that the occurrence of S in P is generated by H. By the properties of $\mathcal{A}$, any occurrence of $S\in\mathcal{S}_k$ is generated by at least one but no more than $\mathcal{O}(\log^2 m)$ anchors. For every $H\in\mathcal{A}$ we create a separate data structure $D(H)$ responsible for setting $V[i+|S|-1]=1$ if $U[i-1]=1$ and $P[i\,..\,(i+|S|-1)]=S$ is generated by H. We separately describe what information is used to initialize each $D(H)$ and how it is later processed to update V.

Initialization. $D(H)$ consists of two compact tries $T(H)$ and $T^r(H)$. For every occurrence of H in P, denoted by $P[i\,..\,(i+|H|-1)]=H$, $T(H)$ should contain a leaf corresponding to $P[(i+|H|)\,..\,|P|]\$$ and $T^r(H)$ should contain a leaf corresponding to $(P[1\,..\,(i-1)])^r\$$, both decorated with position i. Additionally, $D(H)$ stores a list $L(H)$ of pairs of nodes (u,v), where $u\in T^r(H)$ and $v\in T(H)$. Each such pair corresponds to an occurrence of H in a string $S\in\mathcal{S}_k$, $S[j\,..\,(j+|H|-1)]=H$, where u is the node of $T^r(H)$ corresponding to $(S[1\,..\,(j-1)])^r\$$ and v is the node of $T(H)$ corresponding to $S[(j+|H|+1)\,..\,|S|]\$$. We claim that $D(H)$, for all H, can be constructed in $\mathcal{O}(m\log m+N_k)$ total time.

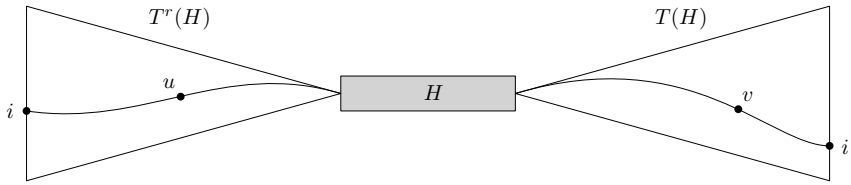


Figure 1 An occurrence of S starting at position i in P is generated by H: (u, v) corresponds to $S[j\,..\,(j+|H|-1)] = H$ and i appears in the subtree rooted at u as well as the subtree rooted at v.

We first construct the suffix tree ST of P and the suffix tree ST^r of P^r in $\mathcal{O}(m \log m)$ time. We augment both trees with a structure for answering *weighted ancestor* (WA) and *lowest common ancestor* (LCA) queries that are defined as follows. For a rooted tree T on n nodes with an integer weight $\mathcal{D}(v)$ assigned to every node u, such that the weight of the root is zero and $\mathcal{D}(u) < \mathcal{D}(v)$ if u is the parent of v, we say that a node v is a weighted ancestor of a node v at depth ℓ, denoted by $\mathrm{WA}_T(u, \ell)$, if v is the highest ancestor of u with weight of at least ℓ. Such queries can be answered in $\mathcal{O}(\log n)$ time after an $\mathcal{O}(n)$-time preprocessing [25]. For a rooted tree T, $\mathrm{LCA}_T(u, v)$ is the lowest node that is an ancestor of both u and v. Such queries can be answered in $\mathcal{O}(1)$ time after an $\mathcal{O}(n)$-time preprocessing [10]. Recall that every anchor H is represented by one of its occurrences in P. Using WA queries, we can access in $\mathcal{O}(\log m)$ time the nodes corresponding to H and H^r, respectively, and extract a lexicographically sorted list of suffixes following an occurrence of H in $P\$$ and a lexicographically sorted list of reversed prefixes preceding an occurrence of H in $P^r\$$ in time proportional to the number of such occurrences. Then, by iterating over the lexicographically sorted list of suffixes and using LCA queries on ST we can build $T(H)$ in time proportional to the length of the list, and similarly for $T^r(H)$. To construct $L(H)$ we start by computing, for every $S \in \mathcal{S}_k$ and $j = 1, \ldots, |S|$, the node of ST^r corresponding to $(S[1\,..\,j])^r$ and the node of ST corresponding to $S([(j+1)\,..\,|S|]$ (the nodes might possibly be implicit). Using suffix links this takes only $\mathcal{O}(|S|)$ time. We also find, for every length-ℓ substring $S[j\,..\,(j+\ell-1)]$ of S, an anchor $H \in \mathcal{A}$ such that $S[j\,..\,(j+\ell-1)] = H$, if any exists. This can be done by finding the nodes (implicit or explicit) of ST that correspond to the anchors, and then scanning over all length-ℓ substrings while maintaining the node of ST corresponding to the current substring using suffix links in $\mathcal{O}(|S|)$ total time. After having determined that $S[j\,..\,(j+\ell-1)] = H$ we add (u, v) to $L(H)$, where u and v are the previously found nodes of ST^r and ST corresponding to $(S[1\,..\,(j-1)])^r$ and $S[(j+\ell)\,..\,|S|]$, respectively. By construction, we have the following property, also illustrated in Figure 1.

▶ **Fact 11.** A string $S \in \mathcal{S}_k$ starts at position $i - j + 1$ in P if and only if, for some anchor $H \in \mathcal{A}$, $L(H)$ contains a pair (u, v) corresponding to $S[j\,..\,(j+|H|-1)] = H$, such that the subtree of $T^r(H)$ rooted at u and that of $T(H)$ rooted at v contain a leaf decorated with i.

Note that the overall size of all lists $L(H)$, when summed up over all $H \in \mathcal{A}$, is $\mathcal{O}(N_k/\ell \cdot \log^2 m)$, because any $S \in \mathcal{S}_k$ contains $\mathcal{O}(\log^2 m)$ occurrences of anchors, and since each S is of length at least ℓ, there are only $\mathcal{O}(N_k/\ell)$ strings in $\mathcal{S}_k$.

Processing. The goal of processing $D(H)$ is to efficiently process all occurrences generated by H. As a preliminary step, we decompose $T^r(H)$ and $T(H)$ into heavy paths. Then, for every pair of leaves $u \in T^r(H)$ and $v \in T(H)$ decorated by the same i, we consider all heavy paths above u and v. Let $p = u_1 - u_2 - \ldots$ be a heavy path above u in $T^r(H)$ and $q = v_1 - v_2 - \ldots$ be a heavy path above v in $T(H)$, where u_1 is the head of p and v_1 is the

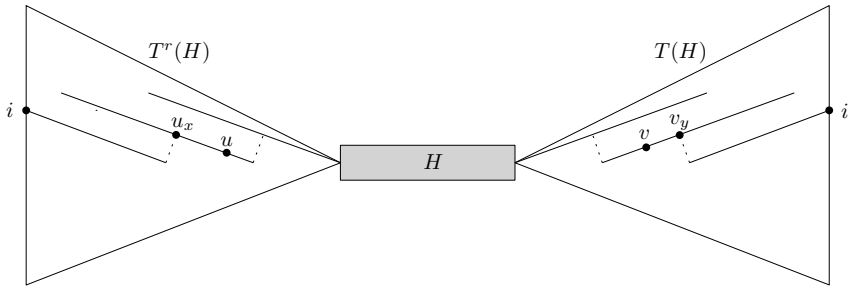


Figure 2 An occurrence of S starting at position i in P corresponds to a triple $(i, \mathcal{L}(u_x), \mathcal{L}(v_y))$ on some auxiliary list.

head of q, respectively. Further, choose the largest x such that u is in the subtree rooted at u_x, and the largest y such that v is in the subtree rooted at v_y (by the choice of p and q, u is in the subtree rooted at u_1 and v is in the subtree rooted at v_1, so this is well-defined). We add $(i, |\mathcal{L}(u_x)|, |\mathcal{L}(v_y)|)$ to an auxiliary list associated with the pair of heavy paths (p, q). In the rest of the processing we work with each such list separately. Notice that the overall size of all auxiliary lists, when summed up over all $H \in \mathcal{A}$, is $\mathcal{O}(m/\ell \cdot \log^4 m)$, because there are at most $\log^2 m$ pairs of heavy paths above u and v decorated by the same i, and the total number of leaves in all trees $T^r(H)$ and $T(H)$ is bounded by the total number of occurrences of all anchors in P, which is $\mathcal{O}(m/\ell \cdot \log^2 m)$. By Fact 11, there is an occurrence of a string $S \in \mathcal{S}_k$ generated by H and starting at position $i - j + 1$ in P if and only if $L(H)$ contains a pair (u, v) corresponding to $S[j\,..\,(j + |H| - 1)] = H$ such that, denoting by p the heavy path containing u in $T^r(H)$ and by q the heavy path containing v in $T(H)$, the auxiliary list associated with (p, q) contains a triple (i, x, y) such that $x \geq |\mathcal{L}(u)|$ and $y \geq |\mathcal{L}(v)|$. This is illustrated in Figure 2. From now on we focus on processing a single auxiliary list associated with (p, q) together with a list of pairs (u, v) such that u belongs to p and v belongs to q.

An auxiliary list can be interpreted geometrically: for every (i, x, y) we create a red point (x, y), and for every (u, v) we create a blue point $(|\mathcal{L}(u)|, |\mathcal{L}(v)|)$. Then, each occurrence of $S \in \mathcal{S}_k$ generated by H corresponds to a pair of points (p_1, p_2) such that p_1 is red, p_2 is blue, and p_1 dominates p_2. We further reduce this to a collection of simpler instances in which all red points already dominate all blue points. This can be done with a divide-and-conquer procedure which is essentially equivalent to constructing a 2D range tree [11]. The total number of points in all obtained instances increases by a factor of $\mathcal{O}(\log^2 m)$, making the total number of red points in all instances $\mathcal{O}(m/\ell \cdot \log^6 m)$, while the total number of blue points is $\mathcal{O}(N_k/\ell \cdot \log^4 m)$. There is an occurrence of a string $S \in \mathcal{S}_k$ generated by H and starting at position $i - j + 1$ in P if and only if some simpler instance contains a red point created for some (i, x, y) and a blue point created for some (u, v) corresponding to $S[j\,..\,(j + |H| - 1)] = H$. In the following we focus on processing a single simpler instance.

To process a simpler instance we need to check if $U[i - j] = 1$, for a red point created for some (i, x, y) and a blue point created for some (u, v) corresponding to $S[j\,..\,(j+|H|-1)] = H$, and if so set $V[i - j + |S|] = 1$. This has a natural interpretation as an instance of BMM: we create an $(5/4 \cdot \ell) \times (5/4 \cdot \ell)$ matrix M such that $M[|S| - j, 5/4 \cdot \ell + 1 - j] = 1$ if and only if there is a blue point created for some (u, v) corresponding to $S[j\,..\,(j+|H|-1)] = H$; then for every red point created for some (i, x, y) we construct a bit vector $U_i = U[(i - 5/4 \cdot \ell)\,..\,(i - 1)]$ (if $i < 5/4 \cdot \ell$, we pad U_i with 0s to make its length always equal to $5/4 \cdot \ell$); calculate $V_i = M \times U_i$; and finally set $V[i + j] = 1$ whenever $V_i[j] = 1$ (and $i + j \leq m$).

▶ **Lemma 12.** *$V_i[k] = 1$ if and only if there is a blue point created for some (u, v) corresponding to $S[j\,..\,(j + |H| - 1)] = H$ such that $U[i - j] = 1$ and $k = |S| - j$.*

The total length of all vectors U_i and V_i is $\mathcal{O}(m \log^6 m)$, so we can afford to extract the appropriate fragment of U and then update the appropriate fragment of V. The bottleneck is computing the matrix-vector product $V_i = M \times U_i$. Naïvely, this might take $\mathcal{O}(N_k/\ell \cdot \log^4 m)$ time, because the total number of 1s in all matrices M is bounded by the total number of blue points. We overcome this by processing together all multiplications concerning the same matrix M. Let $U_{i_1}, U_{i_2}, \ldots, U_{i_s}$ be all bit vectors that need to be multiplied with M, and z a parameter to be determined later. We distinguish between two cases: (i) If $\boldsymbol{s < z}$ we compute the products naïvely by iterating over all 1s in M, and the total computation time, when summed up over all such matrices M, is $\mathcal{O}(N_k/\ell \cdot \log^4 m \cdot z)$; (ii) If $\boldsymbol{s \geq z}$ we partition the bit vectors into $\lceil s/z \rceil \leq s/z + 1$ groups of z (padding the last group with bit vectors containing all 0s). For every group, we create a single matrix whose columns contain all the bit vectors belonging to the group. Thus, we reduce computing all matrix-vector products $M \times U_i$ to computing $\mathcal{O}(s/z)$ matrix-matrix products of the form $M \times M'$, where M' is an $(5/4 \cdot \ell) \times z$ matrix. M' is not necessarily a square matrix, but we can still apply the fast matrix multiplication algorithm to compute $M \times M'$ using the standard trick of decomposing the matrices into square blocks.

▶ **Lemma 13.** *If two $\mathcal{N} \times \mathcal{N}$ matrices can be multiplied in $\mathcal{O}(\mathcal{N}^\omega)$ time, then, for any $\mathcal{N} \geq \mathcal{N}'$, an $\mathcal{N} \times \mathcal{N}$ and an $\mathcal{N} \times \mathcal{N}'$ matrix can be multiplied in $\mathcal{O}((\mathcal{N}/\mathcal{N}')^2 \mathcal{N}'^\omega)$ time.*

By applying Lemma 13, we can compute $M \times M'$ in $\mathcal{O}(\ell^2 z^{\omega-2})$ time (as long as we later verify that $5/4 \cdot \ell \geq z$), so all products $M \times U_i$ can be computed in $\mathcal{O}(\ell^2 z^{\omega-2} \cdot (s/z + 1))$ time. Note that this case can occur only $\mathcal{O}(m/(\ell \cdot z) \cdot \log^6 m)$ times, because all values of s sum up to $\mathcal{O}(m/\ell \cdot \log^6 m)$. This makes the total computation time, when summed up over all such matrices M, $\mathcal{O}(\ell^2 z^{\omega-2} \cdot m/(\ell \cdot z) \cdot \log^6 m) = \mathcal{O}(\ell z^{\omega-3} \cdot m \log^6 m)$.

▶ **Theorem 14.** *An instance of the AP problem where all strings are of type 1 can be solved in expected $\mathcal{O}(m^{1.373} + N)$ time.*

Proof. The total time complexity is first $\mathcal{O}(m + N)$ to construct the graph G and then all graphs G_k, then expected linear time to solve their corresponding instances of the NodeSelection problem, partition $\mathcal{S}$ into $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_{\log m}$ and obtain their corresponding sets of anchors H. The time to initialize all structures $D(H)$ is $\mathcal{O}(m \log m + N_k)$. For every $D(H)$, we obtain in $\mathcal{O}(m/\ell \cdot \log^6 m + N_k/\ell \cdot \log^4 m)$ time a number of simpler instances, and then construct the corresponding Boolean matrices M and bit vectors U_i in additional $\mathcal{O}(m \log^6 m)$ time. Note that some M might be sparse, so we need to represent them as a list of 1s. Then, summing up over all matrices M and both cases, we spend $\mathcal{O}(N_k/\ell \cdot \log^4 m \cdot z + \ell z^{\omega-3} \cdot m \log^6 m)$ time. We would like to assume that $\ell \geq \log^4 m$ so that we can set $z = \ell/\log^4 m$. This is indeed possible, because for any t we can switch to a more naïve approach to process all strings of length at most t in $\mathcal{O}(mt^2 + N_k)$ time: for every possible length $t' \leq t$ scan P with a window of length t', in every step check if the current window $P[i\,..\,(i+t'-1)]$ corresponds to a string $S \in \mathcal{S}$, if so and $U[i-1] = 1$ then set $V[i+t'-1] = 1$. After applying this with $t = \log^4 m$ in $\mathcal{O}(m \log^8 m + N_k)$ time, we can set $z = \ell/\log^4 m$ (so that indeed $5/4 \cdot \ell \geq z$ as required in case $s \geq z$) and the overall time complexity for all matrices M and both cases becomes $\mathcal{O}(N_k + \ell^{\omega-2} \cdot m \log^{6+4(3-\omega)} m)$. Summing up over all values of k and ℓ and taking the initialization into account we obtain $\mathcal{O}(m \log^8 m + m^{\omega-1} \log^{7+4(3-\omega)} m + N) = \mathcal{O}(m^{1.373} + N)$ total time. (We hide $\log^{\mathcal{O}(1)} m$ factors using the fact that $\omega < 2.373$ [30, 54]). ◀

4.2 Wrapping Up

In the full version of the paper we design $\mathcal{O}(m^{1.373} + N)$ and $\mathcal{O}(m^{1.381} + N)$-time algorithms for an instance of the AP problem where all strings are of type 2 and 3, respectively. For type 2 strings, we follow the same ideas as for type 1 strings, except that instead of the NodeSelection problem we select the anchors by applying a periodicity-based argument. For type 3 strings, we need both fast matrix multiplication and FFT. Together with Theorem 14, this gives us Theorem 2.

References

1 A. Abboud, A. Backurs, and V.V. Williams. If the Current Clique Algorithms are Optimal, So is Valiant's Parser. In *56th IEEE Symposium on Foundations Of Computer Science (FOCS)*, pages 98–117, 2015.

2 A. Abboud and V.V. Williams. Popular Conjectures Imply Strong Lower Bounds for Dynamic Problems. In *55th IEEE Symposium on Foundations Of Computer Science (FOCS)*, pages 434–443, 2014.

3 K. Abrahamson. Generalized String Matching. *SIAM J. Comput.*, 16(6):1039–1051, 1987.

4 M. Alzamel, L.A.K. Ayad, G. Bernardini, R. Grossi, C.S. Iliopoulos, N. Pisanti, S.P. Pissis, and G. Rosone. Degenerate string comparison and applications. In *18th International Workshop on Algorithms in Bioinformatics (WABI)*, volume 113 of *LIPIcs*, pages 21:1–21:14, 2018.

5 A. Amir, M. Lewenstein, and E. Porat. Faster algorithms for string matching with k mismatches. *J. Algorithms*, 50(2):257–275, 2004.

6 K. Aoyama, Y. Nakashima, T. I, S. Inenaga, H. Bannai, and M. Takeda. Faster Online Elastic Degenerate String Matching. In *29th Symposium on Combinatorial Pattern Matching (CPM)*, volume 105 of *LIPIcs*, pages 9:1–9:10, 2018.

7 V.L. Arlazarov, E.A. Dinic, M.A. Kronrod, and I.A. Faradžev. On economical construction of the transitive closure of a directed graph. *Soviet Mathematics Doklady*, 11(5):1209–1210, 1970.

8 A. Backurs and P. Indyk. Which Regular Expression Patterns Are Hard to Match? In *57th IEEE Symposium on Foundations Of Computer Science (FOCS)*, pages 457–466, 2016.

9 N. Bansal and R. Williams. Regularity Lemmas and Combinatorial Algorithms. In *50th IEEE Symposium on Foundations Of Computer Science (FOCS)*, pages 745–754, 2009.

10 M.A. Bender and M. Farach-Colton. The LCA Problem Revisited. In *4th Latin American symposium on Theoretical INformatics (LATIN)*, volume 1776 of *Springer LNCS*, pages 88–94, 2000.

11 J.L. Bentley. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM*, 18(9):509–517, 1975.

12 G. Bernardini, N. Pisanti, S.P. Pissis, and G. Rosone. Pattern Matching on Elastic-Degenerate Text with Errors. In *24th International Symposium on String Processing and Information Retrieval (SPIRE)*, pages 74–90, 2017.

13 K. Bringmann, F. Grandoni, B. Saha, and V.V. Williams. Truly Sub-cubic Algorithms for Language Edit Distance and RNA-Folding via Fast Bounded-Difference Min-Plus Product. In *56th IEEE Symposium on Foundations Of Computer Science (FOCS)*, pages 375–384, 2016.

14 K. Bringmann, A. Grønlund, and K.G. Larsen. A Dichotomy for Regular Expression Membership Testing. In *58th IEEE Symposium on Foundations Of Computer Science (FOCS)*, pages 307–318, 2017.

15 T.M. Chan. Speeding Up the Four Russians Algorithm by About One More Logarithmic Factor. In *26th ACM-SIAM Symposium On Discrete Algorithms (SODA)*, pages 212–217, 2015.

16 Y.-J. Chang. Hardness of RNA Folding Problem With Four Symbols. In *27th Symposium on Combinatorial Pattern Matching (CPM)*, volume 54 of *LIPIcs*, pages 13:1–13:12, 2016.

17 K. Chatterjee, B. Choudhary, and A. Pavlogiannis. Optimal Dyck Reachability for Data-dependence and Alias Analysis. In *45th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL)*, pages 30:1–30:30, 2018.

18 A. Cisłak, S. Grabowski, and J. Holub. SOPanG: online text searching over a pan-genome. *Bioinformatics*, page bty506, 2018.

19 P. Clifford and R. Clifford. Simple deterministic wildcard matching. *Inf. Process. Lett.*, 101(2):53–54, 2007.

20 R. Cole and R. Hariharan. Verifying candidate matches in sparse and wildcard matching. In *34th ACM Symposium on Theory Of Computing (STOC)*, pages 592–601, 2002.

21 The Computational Pan-Genomics Consortium. Computational pan-genomics: status, promises and challenges. *Briefings in Bioinformatics*, 19(1):118–135, 2018.

22 M. Crochemore, C. Hancart, and T. Lecroq. *Algorithms on strings*. Cambridge University Press, 2007.

23 M. Crochemore and D. Perrin. Two-Way String Matching. *J. ACM*, 38(3):651–675, 1991.

24 A. Czumaj and A. Lingas. Finding a Heaviest Vertex-Weighted Triangle Is not Harder than Matrix Multiplication. *SIAM J. Comput.*, 39(2):431–444, 2009.

25 M. Farach and S. Muthukrishnan. Perfect Hashing for Strings: formalization and Algorithms. In *7th Annual Symposium on Combinatorial Pattern Matching (CPM)*, volume 1075 of *Springer LNCS*, pages 130–140, 1996.

26 N.J. Fine and H.S. Wilf. Uniqueness Theorems for Periodic Functions. *Proceedings of the American Mathematical Society*, 16(1):109–114, 1965.

27 M.J. Fischer and A.R. Meyer. Boolean Matrix Multiplication and Transitive Closure. In *12th IEEE Symposium on Switching and Automata Theory (SWAT/FOCS)*, pages 129–131, 1971.

28 M.J. Fischer and M.S. Paterson. String matching and other products. In *7th SIAM-AMS Complexity of Computation*, pages 113–125, 1974.

29 M.E. Furman. Application of a method of fast multiplication of matrices in the problem of finding the transitive closure of a graph. *Soviet Mathematics Doklady*, 11(5):1252, 1970.

30 F. Le Gall. Powers of tensors and fast matrix multiplication. In *39th International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 296–303, 2014.

31 P. Gawrychowski and P. Uznański. Towards Unified Approximate Pattern Matching for Hamming and L_1 Distance. In *45th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 107 of *LIPIcs*, pages 62:1–62:13, 2018.

32 R. Grossi, C.S. Iliopoulos, C. Liu, N. Pisanti, S.P. Pissis, A. Retha, G. Rosone, F. Vayani, and L. Versari. On-Line Pattern Matching on Similar Texts. In *28th Symposium on Combinatorial Pattern Matching (CPM)*, volume 78 of *LIPIcs*, pages 9:1–9:14, 2017.

33 M. Henzinger, S. Krinninger, D. Nanongkai, and T. Saranurak. Unifying and Strengthening Hardness for Dynamic Problems via the Online Matrix-Vector Multiplication Conjecture. In *47th ACM Symposium on Theory Of Computing (STOC)*, pages 21–30, 2015.

34 J. Holub, W.F. Smyth, and S. Wang. Fast pattern-matching on indeterminate strings. *J. Discrete Algorithms*, 6(1):37–50, 2008.

35 C.S. Iliopoulos, R. Kundu, and S.P. Pissis. Efficient Pattern Matching in Elastic-Degenerate Texts. In *11th International Conference on Language and Automata Theory and Applications (LATA)*, volume 10168 of *Springer LNCS*, pages 131–142, 2017.

36 P. Indyk. Faster Algorithms for String Matching Problems: Matching the Convolution Bound. In *39th Symposium on Foundations Of Computer Science (FOCS)*, pages 166–173, 1998.

37 A. Itai and M. Rodeh. Finding a Minimum Circuit in a Graph. In *9th ACM Symposium on Theory Of Computing (STOC)*, pages 1–10, 1977.

38 IUPAC-IUB Commission on Biochemical Nomenclature. Abbreviations and symbols for nucleic acids, polynucleotides, and their constituents. *Biochemistry*, 9(20):4022–4027, 1970.

39 A. Kalai. Efficient pattern-matching with don't cares. In *13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 655–656, 2002.

40 D.E. Knuth, J.H. Morris Jr., and V.R. Pratt. Fast Pattern Matching in Strings. *SIAM J. Comput.*, 6(2):323–350, 1977.

41 T. Kociumaka, J. Radoszewski, W. Rytter, and T. Waleń. Internal Pattern Matching Queries in a Text and Applications. In *26th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 532–551, 2015.

42 T. Kopelowitz and R. Krauthgamer. Color-distance oracles and snippets. In *27th Symposium on Combinatorial Pattern Matching (CPM)*, volume 54 of *LIPIcs*, pages 24:1–24:10, 2016.

43 K.G. Larsen, I. Munro, J.S. Nielsen, and S.V. Thankachan. On hardness of several string indexing problems. *Theor. Comput. Sci.*, 582:74–82, 2015.

44 L. Lee. Fast Context-free Grammar Parsing Requires Fast Boolean Matrix Multiplication. *J. ACM*, 49(1):1–15, 2002.

45 J. Matoušek. Computing Dominances in E^n. *Inf. Process. Lett.*, 38(5):277–278, 1991.

46 I. Munro. Efficient Determination of the Transitive Closure of a Directed Graph. *Inf. Process. Lett.*, 1(2):56–58, 1971.

47 G. Navarro. NR-grep: a fast and flexible pattern-matching tool. *Softw., Pract. Exper.*, 31(13):1265–1312, 2001.

48 S.P. Pissis and A. Retha. Dictionary Matching in Elastic-Degenerate Texts with Applications in Searching VCF Files On-line. In *17th International Symposium on Experimental Algorithms (SEA)*, volume 103 of *LIPIcs*, pages 16:1–16:14, 2018.

49 L. Roditty and U. Zwick. On Dynamic Shortest Paths Problems. In *12th European Symposium on Algorithms (ESA)*, volume 3221 of *Springer LNCS*, pages 580–591, 2004.

50 M. Ružić. Constructing Efficient Dictionaries in Close to Sorting Time. In *35th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 5125 of *Springer LNCS*, pages 84–95, 2008.

51 D.D. Sleator and R.E. Tarjan. A Data Structure for Dynamic Trees. *J. Comput. Syst. Sci.*, 26(3):362–391, 1983.

52 L.G. Valiant. General Context-free Recognition in Less Than Cubic Time. *J. Comput. Syst. Sci.*, 10(2):308–315, April 1975.

53 P. Weiner. Linear Pattern Matching Algorithms. In *14th IEEE Annual Symposium on Switching and Automata Theory (SWAT/FOCS)*, pages 1–11, 1973.

54 V.V. Williams. Multiplying matrices faster than Coppersmith-Winograd. In *44th ACM Symposium on Theory Of Computing Conference (STOC)*, pages 887–898, 2012.

55 V.V. Williams and R. Williams. Finding a maximum weight triangle in $n^{3-\delta}$ time, with applications. In *38th ACM Symposium on Theory Of Computing Conference (STOC)*, pages 225–231, 2006.

56 V.V. Williams and R. Williams. Subcubic equivalences between path, matrix and triangle problems. In *51st IEEE Symposium on Foundations Of Computer Science (FOCS)*, pages 645–654, 2010.

57 S. Wu and U. Manber. Agrep – A Fast Approximate Pattern-Matching Tool. In *USENIX Technical Conference*, pages 153–162, 1992.

58 H. Yu. An improved combinatorial algorithm for Boolean matrix multiplication. *Inf. Comput.*, 261(Part):240–247, 2018.

59 U. Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *J. ACM*, 49(3):289–317, 2002.

The Complexity of Approximating the Matching Polynomial in the Complex Plane

Ivona Bezáková
Department of Computer Science, Rochester Institute of Technology, Rochester, NY, USA
ib@cs.rit.edu

Andreas Galanis
Department of Computer Science, University of Oxford, UK
andreas.galanis@cs.ox.ac.uk

Leslie Ann Goldberg
Department of Computer Science, University of Oxford, UK
leslie.goldberg@cs.ox.ac.uk

Daniel Štefankovič
Department of Computer Science, University of Rochester, Rochester, NY, USA
stefanko@cs.rochester.edu

Abstract

We study the problem of approximating the value of the matching polynomial on graphs with edge parameter γ, where γ takes arbitrary values in the complex plane.

When γ is a positive real, Jerrum and Sinclair showed that the problem admits an FPRAS on general graphs. For general complex values of γ, Patel and Regts, building on methods developed by Barvinok, showed that the problem admits an FPTAS on graphs of maximum degree Δ as long as γ is not a negative real number less than or equal to $-1/(4(\Delta-1))$. Our first main result completes the picture for the approximability of the matching polynomial on bounded degree graphs. We show that for all $\Delta \geq 3$ and all real γ less than $-1/(4(\Delta-1))$, the problem of approximating the value of the matching polynomial on graphs of maximum degree Δ with edge parameter γ is #P-hard.

We then explore whether the maximum degree parameter can be replaced by the connective constant. Sinclair et al. showed that for positive real γ it is possible to approximate the value of the matching polynomial using a correlation decay algorithm on graphs with bounded connective constant (and potentially unbounded maximum degree). We first show that this result does not extend in general in the complex plane; in particular, the problem is #P-hard on graphs with bounded connective constant for a dense set of γ values on the negative real axis. Nevertheless, we show that the result does extend for any complex value γ that does not lie on the negative real axis. Our analysis accounts for complex values of γ using geodesic distances in the complex plane in the metric defined by an appropriate density function.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Theory of computation → Problems, reductions and completeness

Keywords and phrases matchings, partition function, correlation decay, connective constant

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.22

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `http://arxiv.org/abs/1807.04930`. The theorem numbering here matches the full version.

Funding *Ivona Bezáková*: Research supported by NSF grant CCF-1319987.
Andreas Galanis: The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) ERC grant agreement no. 334828. The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 22; pp. 22:1–22:13

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

Leslie Ann Goldberg: The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) ERC grant agreement no. 334828. The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein.
Daniel Štefankovič: Research supported by NSF grant CCF-1563757.

1 Introduction

We study the problem of approximating the matching polynomial of a graph. This polynomial has a parameter γ, called the edge activity. A *matching* of a graph G is a set $M \subseteq E(G)$ such that each vertex $v \in V(G)$ is contained in at most one edge in M. We denote by $\mathcal{M}_G$ the set of all matchings of G. The matching polynomial $Z_G(\gamma)$ is given by

$$Z_G(\gamma) = \sum_{M \in \mathcal{M}_G} \gamma^{|M|}.$$

This polynomial is also referred to as the partition function of the *monomer-dimer model* in statistical physics.

Here is what is known about approximating this polynomial. We first describe the case where γ is positive and real. This is a natural case, and is the case where the first complexity-theoretic results were obtained. We next describe the more general case, where γ is a complex number. There are many reasons for considering the more general case. The parameter γ is defined to be complex, rather than real, in the classic paper of Heilmann and Lieb [8]. Furthermore, it has recently been shown [15] that the quantum evolution of a system originally in thermodynamic equilibrium is equivalent to the partition function of the system with a complex parameter. As [15] explains, recent discoveries in physics make it possible to study thermodynamics in the complex plane of physical parameters – so complex parameters are increasingly relevant. As we will see in this paper, it is beneficial to study partition functions with complex parameters even when one is most interested in the real case – the reason is that the generalisation sheds light on "what is really going on" with complexity bottlenecks, and on appropriate potential functions. Here is the summary of known results in both cases.

- **When the edge activity γ is a positive real number:** For any positive real number γ, Jerrum and Sinclair [9, Corollary 4.4] gave an FPRAS for approximating $Z_G(\gamma)$. Using the correlation decay technique, Bayati et al. [3] gave a (deterministic) FPTAS for the same problem for the case in which the degree of the input graph G is at most a constant Δ.
- **When the edge activity γ is a complex number:** Known results are restricted to the case where γ is not a real number less than or equal to $-1/(4(\Delta-1))$. In this case, there is a positive result, due to Patel and Regts [11]. Using a method of Barvinok [1, 2] for approximating a partition function by truncating its Taylor series (in a region where the partition function has no zeroes), Patel and Regts [11, Theorem 1.2] extended the positive result of Bayati et al. to the case in which γ is a complex number that is not a negative real that is less than $-1/(4(\Delta-1))$, see also [2, Section 5.1.7]). Patel and Regts obtained a polynomial time algorithm (rather than a quasi-polynomial time one) by developing clever methods for exactly computing coefficients of the Taylor series.

Our first contribution completes this picture by showing that for all $\Delta \geq 3$ and all real $\gamma < -1/(4(\Delta-1))$ it is actually #P-hard to approximate $Z_G(\gamma)$ on graphs with degree at most Δ. We use the following notation to state our result more precisely. We consider the problems of multiplicatively approximating the norm of $Z_G(\gamma)$, and of computing its sign. Our first theorem shows that, for all $\Delta \geq 3$ and all rational numbers $\gamma < -1/(4(\Delta-1))$, it is #P-hard to approximate $|Z_G(\gamma)|$ on bipartite graphs of maximum degree Δ within a constant factor.

▶ **Theorem 1.** *Let $\Delta \geq 3$ and $\gamma < -\frac{1}{4(\Delta-1)}$ be a rational number. Then, it is #P-hard to approximate $|Z_G(\gamma)|$ within a factor of 1.01 on graphs G of maximum degree Δ, even when restricted to bipartite graphs G with $Z_G(\gamma) \neq 0$.*

The number 1.01 in Theorem 1 is not important. It can be replaced with any constant greater than 1. In fact, for any fixed $\epsilon > 0$, the theorem, together with a standard powering argument, shows that it is #P-hard to approximate $Z_G(\gamma)$ within a factor of $2^{|V(G)|^{1-\epsilon}}$.

Our second theorem shows that it is #P-hard to compute the sign of $Z_G(\gamma)$ on bipartite graphs of maximum degree Δ.

▶ **Theorem 2.** *Let $\Delta \geq 3$ and $\gamma < -\frac{1}{4(\Delta-1)}$ be a rational number. Then, it is #P-hard to decide whether $Z_G(\gamma) > 0$ on graphs G of maximum degree Δ, even when restricted to bipartite graphs G with $Z_G(\gamma) \neq 0$.*

We next explore whether the bound on the maximum degree of G can be relaxed to a restriction on average degree. The notion of average degree that we use is the *connective constant*. Given a graph G, and a vertex v, let $N_G(v,k)$ be the number of k-edge paths in G that start from v. The following definition is taken almost verbatim from [13, 14].[1]

▶ **Definition 3** ([13, 14])**.** *Let $\mathcal{F}$ be a family of finite graphs and let Δ, a and c be positive real numbers. The connective constant of $\mathcal{F}$ is at most Δ with profile (a,c) if, for any graph $G=(V,E)$ in $\mathcal{F}$ and any vertex v in G, it holds that $\sum_{k=1}^{\ell} N_G(v,k) \leq c\Delta^{\ell}$ for all $\ell \geq a \log |V|$.*

Sinclair, Srivastava, Štefankovič and Yin [13, Theorem 1.3] showed that, for fixed Δ, when γ is a positive real, the correlation decay method gives an FPTAS for approximating $Z_G(\gamma)$ on graphs G with connective constant at most Δ (without any bound on the maximum degree of G). The run-time of their algorithm is $(n/\epsilon)^{\mathrm{O}(\sqrt{\gamma\Delta}\log\Delta)}$, where n is the number of vertices of G and ϵ is the relative error.

Our next result shows that, in striking contrast to the bounded-degree case, the algorithmic result of Sinclair et al. cannot be extended to negative reals, even if $\gamma \geq -1/(4(\Delta-1))$. Given positive real numbers a and c and a real number $\Delta > 1$, let $\mathcal{F}_{\Delta,a,c}$ be the set of graphs with connective constant at most Δ and profile (a,c).

▶ **Theorem 4.** *There exist a dense set of values γ on the negative real axis such that the following holds for any real numbers $\Delta > 1$ and all $a, c > 0$.*

1. *It is #P-hard to approximate $|Z_G(\gamma)|$ within a factor 1.01 on graphs $G \in \mathcal{F}_{\Delta,a,c}$,*
2. *it is #P-hard to decide whether $Z_G(\gamma) > 0$ on graphs $G \in \mathcal{F}_{\Delta,a,c}$.*

Both of these results hold even when restricted to bipartite graphs G with $Z_G(\gamma) \neq 0$.

[1] The only difference between Definition 3 and the corresponding definitions in [13, 14] is the addition of the terminology "profile (a,c)" which will be used to state our hardness results in a strong form (the results in [13, 14] were algorithmic which is why this handle on the constants a and c was not required).

The algorithmic contribution of our paper is to show that, despite the hardness result of Theorem 4, correlation decay gives a good approximation algorithm for any complex value γ that does not lie on the negative real axis when the input graph has bounded connective constant. It is interesting that we are able to use correlation decay to get a good approximation for all non-real complex values γ. Our result is the only known approximation in this setting. In particular, it is not known how to obtain such a result using the method of Patel and Regts [11]. In order to describe our result, we use the following notation. Given a complex number x, let $\arg(x)$ denote the principal value of its argument in the range $[0, 2\pi)$ and $|x|$ denote its norm. Our result is the following.

▶ **Theorem 5.** *Let Δ, a and c be positive real numbers and let $\gamma \in \mathbb{C} \setminus \mathbb{R}_{<0}$ be any fixed edge activity. Then there is an algorithm which takes as input an n-vertex graph $G \in \mathcal{F}_{\Delta,a,c}$ and a rational $\epsilon \in (0,1)$ and produces an output $\hat{Z} = Z_G(\gamma)\mathrm{e}^z$ for some complex number z with $|z| \leq \epsilon$. The running time of the algorithm is $(\hat{c}n/\epsilon)^{O\left((1+a+\sqrt{|\hat{\gamma}|\Delta})\log\Delta\right)}$ where $\hat{\gamma} = \frac{2|\gamma|}{1+\cos(\arg\gamma)}$ and $\hat{c} = \max\{1, c\}$.*

Theorem 5 gives an algorithmic result which contrasts with the hardness results of Theorems 1 and 2. It has the following corollary.

▶ **Corollary 6.** *Let Δ, a and c be positive real numbers and let $\gamma \in \mathbb{C} \setminus \mathbb{R}_{<0}$ be any fixed edge activity. Then, for any rational $K > 1$ and any positive rational ρ, there are polynomial-time algorithms to take as input a graph $G \in \mathcal{F}_{\Delta,a,c}$ and approximate $|Z_G(\gamma)|$ within a multiplicative factor of K and $\arg(Z_G(\gamma))$ within an additive error ρ.*

In order to prove Theorem 5, showing correlation decay for complex γ, we use geodesic distances in the complex plane in the metric defined by an appropriate density function. Correlation decay for complex activities has been analysed in the context of the hard-core model (see Harvey, Srivastava and Vondrák [7])[2]. The region in the complex plane in which the authors of [7] worked allowed them to measure distances using the norm instead of requiring geodesic distances. An alternative approach was given by Peters and Regts [12], again in the context of the hard-core model, where they showed contraction within the basin of an attracting fixpoint using the theory of complex dynamical systems.

2 Preliminaries

Let γ be a complex number and $G = (V, E)$ be an arbitrary graph. Recall that $\mathcal{M}_G$ is the set of matchings of G. For a matching $M \in \mathcal{M}_G$, we denote by $\mathsf{ver}(M)$ the set of matched vertices in the matching M. For a vertex u in G, we also define

$$Z_{G,u}(\gamma) := \sum_{M \in \mathcal{M}_G;\, u \in \mathsf{ver}(M)} \gamma^{|M|} \quad \text{and} \quad Z_{G,\neg u}(\gamma) := \sum_{M \in \mathcal{M}_G;\, u \notin \mathsf{ver}(M)} \gamma^{|M|}.$$

Thus, $Z_{G,u}(\gamma)$ is the contribution to the partition function $Z_G(\gamma)$ from those matchings $M \in \mathcal{M}_G$ such that u is matched in M, while $Z_{G,\neg u}(\gamma)$ is the contribution to the partition function $Z_G(\gamma)$ from those matchings $M \in \mathcal{M}_G$ such that u is not matched in M.

We will use the following result about the location of the zeroes of the matching polynomial.

[2] Note that Harvey et al were actually working with the mutivariate hard-core polynomial – this causes interesting complications which will not be relevant for this paper. They also extend their method (for the hard-core polynomial, in their region) to graphs of unbounded degree that have bounded connective constant.

▶ **Theorem 7** ([8], see, e.g., [2, Theorem 5.1.2]). *Let $\Delta \geq 3$ be an integer and G be a graph of maximum degree Δ. Then, for all complex γ that do not lie on the interval $(-\infty, -\frac{1}{4(\Delta-1)})$ of the negative real axis, it holds that $Z_G(\gamma) \neq 0$.*

▶ **Corollary 8.** *Let $\Delta \geq 3$ be an integer and $\gamma > -\frac{1}{4(\Delta-1)}$ be a real number. Then, for all graphs G of maximum degree Δ it holds that $Z_G(\gamma) > 0$.*

For our approximation algorithm of Theorem 5, given a graph $G = (V, E)$ with $Z_G(\gamma) \neq 0$ and a vertex $v \in V$, we will be interested in the quantity

$$p_v(G, \gamma) := Z_{G,\neg v}(\gamma)/Z_G(\gamma).$$

The algorithm will be based on the following result by Godsil.

▶ **Theorem 9** ([6]). *Let $\gamma \in \mathbb{C}\backslash\mathbb{R}_{<0}$. Let $G = (V, E)$ be a graph and let $v \in V$ be one of its vertices. Let $T_{SAW}(v, G)$ be the self-avoiding walk tree of G rooted at v. Then,*

$$p_v(G, \gamma) = p_v(T_{SAW}(v, G), \gamma).$$

3 FPTAS for graphs with bounded connective constant

In this section, we prove Theorem 5. Consider $\gamma \in \mathbb{C}\backslash\mathbb{R}_{<0}$.

We will use the correlation decay technique of Weitz [16], which we adapt for use with complex activities. We review the basic idea behind the technique (see, e.g., [3, 14, 13]). For a graph G (of bounded connective constant), we first express $Z_G(\gamma)$ as a telescoping product

$$Z_G(\gamma) = 1/\prod_{i=1}^{n} p_{v_j}(G_j, \gamma) \tag{1}$$

where $v_1, \ldots, v_n$ is an arbitrary enumeration of the vertices of the graph G and G_j is the graph obtained from G by deleting the vertices $v_1, \ldots, v_j$. In light of (1), we can therefore focus on approximating the value $p_v(G, \gamma)$ for a graph G and vertex v. Using Godsil's Theorem (cf. Theorem 9), it in turn suffices to approximate $p_v(T_{SAW}(v, G), \gamma)$. This might seem as a somewhat simpler task given that $T_{SAW}(v, G)$ is a tree; the caveat however is that the tree $T_{SAW}(v, G)$ is prohibitively large, so in order to be able to perform computations efficiently we need to truncate the tree. The correlation decay technique analyses the approximation error introduced by this truncation process by recursively tracking the error using tree recurrences.

In the case of matchings, for a tree T and a vertex v in T, we can write a recursion for $p_v(T, \gamma)$ as follows. If v is the only vertex in T, then $p_v(T, \gamma) = 1$ (since the only possible matching is the empty set and thus $Z_{T,\neg v}(\gamma) = Z_T(\gamma) = 1$). Otherwise, let $T_1, \ldots, T_d$ be the trees of $T\backslash\{v\}$ and let $v_1, \ldots, v_d$ be the neighbours of v in $T_1, \ldots, T_d$, respectively. Then, we have that

$$Z_{T,\neg v}(\gamma) = \prod_{i=1}^{d} Z_{T_i}(\gamma), \quad Z_T(\gamma) = \prod_{i=1}^{d} Z_{T_i}(\gamma) + \sum_{i=1}^{d} \gamma\, Z_{T_i,\neg v_i}(\gamma) \prod_{j\in\{1,\ldots,d\}, j\neq i} Z_{T_j}(\gamma)$$

and therefore

$$p_v(T, \gamma) = \frac{Z_{T,\neg v}(\gamma)}{Z_T(\gamma)} = \frac{1}{1 + \gamma \sum_{i=1}^{d} \frac{Z_{T_i,\neg v_i}(\gamma)}{Z_{T_i}(\gamma)}} = \frac{1}{1 + \gamma \sum_{i=1}^{d} p_{v_i}(T_i, \gamma)}.$$

Hence, we need to evaluate the recurrence

$$x = F(x_1, \ldots, x_d) \text{ where } F(x_1, \ldots, x_d) = \frac{1}{1 + \gamma \sum_{i=1}^{d} x_i}, \tag{2}$$

with base case $x = 1$.

To show the decay of correlations, one wants to show that after applying the recurrence starting from two different sets of values at $v_1, \ldots, v_d$, the two computed values at v will be "closer" than were the initial values at the v_i's. This leads us to define a notion of distance. Often straightforward distances do not suffice to show decay of correlations, and distances defined via a "potential" function are used. We adapt this notion to the complex plane.

3.1 Metrics for measuring the error in the complex plane

We use a distance metric based on conformal density functions (see [10] for details).

▶ **Definition 10** (Length, Distance, Metric). *Let U be a simply connected open subset of $\mathbb{C}$ and let $\Phi : U \to \mathbb{R}_{>0}$ be a function (called conformal density). The* length with respect to Φ *of a path*[3] *$\eta : [0, 1] \to U$ is defined as*

$$\int_0^1 \Phi(\eta(t)) \left| \frac{\partial}{\partial t} \eta(t) \right| dt.$$

The distance with respect to Φ *between two points $x, y \in U$, denoted $\mathrm{dist}_\Phi(x, y)$, is the infimum of the lengths of the paths η connecting x to y (that is, $\eta(0) = x$ and $\eta(1) = y$). We will refer to the metric induced by the distance function $\mathrm{dist}_\Phi(\cdot, \cdot)$ as the (conformal) metric given by Φ.*

We first quantify one-level correlation decay.

▶ **Lemma 12.** *Let U be a simply connected open subset of $\mathbb{C}$, $\Phi : U \to \mathbb{R}_{>0}$ be a conformal density function, and $\mathrm{dist}_\Phi(\cdot, \cdot)$ be the metric given by Φ. Let p and q be conjugate exponents, that is, $1/p + 1/q = 1$, where $p, q \in \mathbb{R}_{>0} \cup \{\infty\}$.*

Suppose that $d \geq 1$ is an integer and $F : U^d \to U$ is a holomorphic map. Let $x_1, \ldots, x_d \in U$ and $y_1, \ldots, y_d \in U$ and let $x = F(x_1, \ldots, x_d)$ and $y = F(y_1, \ldots, y_d)$. Assume that there exists a real $\alpha \in (0, 1)$ such that for any $z_1, \ldots, z_d \in U$

$$\sum_{i=1}^{d} \left| \Phi(F(z_1, \ldots, z_d)) \frac{\partial F}{\partial z_i}(z_1, \ldots, z_d) \frac{1}{\Phi(z_i)} \right|^p \leq \alpha^p. \tag{3}$$

Then

$$\mathrm{dist}_\Phi(x, y) \leq \alpha \left(\sum_{i=1}^{d} \mathrm{dist}_\Phi(x_i, y_i)^q \right)^{1/q}. \tag{4}$$

Proof. Let $\epsilon > 0$. For $i \in [d]$, let η_i be a path connecting x_i to y_i of length $\ell_i \leq \mathrm{dist}_\Phi(x_i, y_i) + \epsilon$. We assume w.l.o.g. that η_i is re-parameterized to uniform speed (using arc length), that is, for a.e. $t \in [0, 1]$ we have

$$\left| \frac{\partial}{\partial t} \eta_i(t) \right| \Phi(\eta_i(t)) = \ell_i. \tag{5}$$

[3] Following [10], paths are assumed to be continuous and piecewise continuously differentiable.

We now define a path η connecting x to y:

$$\eta(t) := F(\eta_1(t), \ldots, \eta_d(t)).$$

Let L denote the length of η and $F_i(x_1, \ldots, x_d)$ denote the function $\frac{\partial F}{\partial x_i}(x_1, ..., x_d)$ Then, using the triangle inequality and (5), we have

$$\begin{aligned} L &= \int_0^1 \Phi(\eta(t)) \Big| \frac{\partial}{\partial t}\eta(t) \Big| \, dt = \int_0^1 \Phi(\eta(t)) \Big| \sum_{i=1}^d F_i(\eta_1(t), \ldots, \eta_d(t)) \, \frac{\partial \eta_i}{\partial t}(t) \Big| \, dt \\ &\leq \int_0^1 \Phi(\eta(t)) \sum_{i=1}^d \Big| F_i(\eta_1(t), \ldots, \eta_d(t)) \, \frac{\partial \eta_i}{\partial t}(t) \Big| \, dt \qquad (6) \\ &= \int_0^1 \sum_{i=1}^d \Big| \Phi(\eta(t)) F_i(\eta_1(t), \ldots, \eta_d(t)) \frac{1}{\Phi(\eta_i(t))} \, \ell_i \Big| \, dt. \end{aligned}$$

By Hölder's inequality and condition (3), for any $t \in [0, 1]$, we have

$$\sum_{i=1}^d \Big| \Phi(\eta(t)) F_i(\eta_1(t), \ldots, \eta_d(t)) \frac{1}{\Phi(\eta_i(t))} \ell_i \Big| \leq$$

$$\left(\sum_{i=1}^d \Big| \Phi(\eta(t)) F_i(\eta_1(t), \ldots, \eta_d(t)) \frac{1}{\Phi(\eta_i(t))} \Big|^p \right)^{1/p} \left(\sum_{i=1}^d \ell_i^q \right)^{1/q} \leq \alpha \Big(\sum_{i=1}^d \ell_i^q \Big)^{1/q}.$$

Integrating this for t between 0 and 1 and combining with (6), we obtain

$$\mathrm{dist}_\Phi(x, y) \leq L \leq \alpha \Big(\sum_{i=1}^d \ell_i^q \Big)^{1/q}.$$

Taking $\epsilon \to 0$ we obtain

$$\mathrm{dist}_\Phi(x, y) \leq \alpha \left(\sum_{i=1}^d \mathrm{dist}_\Phi(x_i, y_i)^q \right)^{1/q}. \qquad \blacktriangleleft$$

Now, given a rooted tree, our goal will be to bound the correlation decay at the root when we truncate the tree at depth $\Theta(\log n)$. Let T be a finite tree rooted at a vertex ρ and let C be a subset of the leaves of T. Let $U \subseteq \mathbb{C}$. We will have a family of maps $\{F_d\}_{d \geq 1}$ where $F_d : U^d \mapsto U$ will be a symmetric map of arity d (which will be the recurrence applied to a vertex of the tree with d children). Let $\sigma : C \to U$ be an arbitrary assignment of values in U to the vertices of C. Let also $u_0 \in U$ be the "initial" value (u_0 corresponds to the starting point of the recurrences). For a vertex v in T and an initial value $u_0 \in U$, we define the quantity $r_v(C, \sigma, u_0)$ recursively as follows.

$$r_v(C, \sigma, u_0) = \begin{cases} u_0 & \text{if } v \text{ is a leaf of } T \text{ and } v \notin C, \\ \sigma(v) & \text{if } v \in C, \\ F_d(x_1, \ldots, x_d) & \text{otherwise, where } x_i = r_{v_i}(C, \sigma, u_0) \\ & \text{and } v_1, \ldots, v_d \text{ are } v\text{'s children in } T. \end{cases} \qquad (7)$$

We can now study the sensitivity of $r_v(C, \sigma, u_0)$ to the assignment σ. The following lemma is the analogue of [14, Lemma 3] for the complex plane and will be used to apply the correlation decay technique for graphs of bounded connective constant, the proof can be found in the full version.

▶ **Lemma 13.** *Let U be a simply connected open subset of $\mathbb{C}$ and $\Phi : U \to \mathbb{R}_{>0}$ be a conformal density function. For $d = 1, 2, \ldots$, let $F_d : U^d \mapsto U$ be symmetric holomorphic maps. Suppose that there exists a real $\alpha \in (0, 1)$ and conjugate exponents p and q such that for every integer $d \geq 1$ and all $z_1, \ldots, z_d \in U$ it holds that*

$$\sum_{i=1}^{d} \left| \Phi(F_d(z_1, \ldots, z_d)) \frac{\partial F_d}{\partial z_i}(z_1, \ldots, z_d) \frac{1}{\Phi(z_i)} \right|^p \leq \alpha^p. \tag{8}$$

Then, the following holds for any initial value $u_0 \in U$ and any finite tree T rooted at ρ.

Let C be a subset of the leaves of T and consider two arbitrary assignments $\sigma_1 : C \to U$ and $\sigma_2 : C \to U$. Then

$$|r_\rho(C, \sigma_1, u_0) - r_\rho(C, \sigma_2, u_0)| \leq \left(\frac{M}{L}\right) \left(\sum_{v \in C} \alpha^{q \cdot \mathrm{depth}(v)} \right)^{1/q},$$

where $L := \inf_{x \in U} \Phi(x)$, $M := \max_{v \in C} \mathrm{dist}_\Phi(\sigma_1(v), \sigma_2(v))$ and $\mathrm{depth}(v)$ is the distance of v from the root ρ.

3.2 Applying the method for matchings

Suppose that $\gamma \in \mathbb{C} \setminus \mathbb{R}_{\leq 0}$. We will parameterise γ as

$$\gamma = (1/Q)^2, \text{ where we choose } Q \text{ such that } \mathrm{Re}(Q) > 0. \tag{9}$$

Note that, in the choice of Q, we used the assumption that γ is not a negative real number. Let $\mathcal{H}$ be the right complex half-plane, that is, the set of complex x such that $\mathrm{Re}(x) > 0$, and note that $Q \in \mathcal{H}$. We will also transform the space in which the quantities $p_v(G, \gamma)$ live using the map $x \mapsto x/Q$. In the transformed space, the recurrence (2) becomes

$$y = F(y_1, \ldots, y_d) \text{ where } F(y_1, \ldots, y_d) = \frac{1}{Q + \sum_{i=1}^{d} y_i}, \tag{10}$$

where if y corresponds to a leaf then $y = 1/Q$ (we refer to this y as the initial y). Let

$$U = \left\{ y \in \mathbb{C} \mid \mathrm{Re}(y) > 0,\ |y| < 1/\mathrm{Re}(Q) \right\}. \tag{11}$$

The following lemma shows that the set U is closed under application of the recurrence (10).

▶ **Lemma 14.** *Suppose that $y_1, \ldots, y_d \in U$ and $\mathrm{Re}(Q) > 0$. Then, for y given by (10), we have that $y \in U$ as well. In fact, we have that $\mathrm{Re}(y) \geq \frac{\mathrm{Re}(Q)}{\left(|Q| + \frac{d}{\mathrm{Re}(Q)}\right)^2}$.*

We next go on to show the required contraction properties for an appropriate function Φ. This is largely based on arguments from [13] from the real case, which we can adapt to the complex plane to obtain the following.

▶ **Lemma 16.** *Let Δ be a positive real number, $\gamma \in \mathbb{C} \setminus \mathbb{R}_{<0}$, and Q, U be given from (9) and (11), respectively. Consider the function $\Phi : U \mapsto \mathbb{R}_{>0}$ given by $\Phi(y) = \frac{1}{\mathrm{Re}(y)(2/\mathrm{Re}(Q) - \mathrm{Re}(y))}$ and let $\hat{\gamma} = \frac{2|\gamma|}{1 + \cos(\arg\gamma)}$,*

$$D = \max\{\Delta, \frac{3}{4\hat{\gamma}}\},\ \ p = 1/(1 - \frac{1}{\sqrt{1 + 4\hat{\gamma}D}}),\ \ q = \frac{p}{p-1},\ \ \alpha = \frac{1}{D^{1/q}}\Big(1 - \frac{2}{1 + \sqrt{1 + 4\hat{\gamma}D}}\Big). \tag{12}$$

Then, the following holds for all integer $d \geq 1$.

Consider the map $F : U^d \mapsto U$ given by $F(y_1, \ldots, y_d) = \frac{1}{Q + \sum_{i=1}^{d} y_i}$. Then, for arbitrary $y_1, \ldots, y_d \in U$ we have

$$\sum_{i=1}^{d} \left| \Phi(F(y_1, \ldots, y_d)) \frac{\partial F}{\partial y_i}(y_1, \ldots, y_d) \frac{1}{\Phi(y_i)} \right|^p \leq \alpha^p. \tag{13}$$

Based on the above lemmas, we can now give a proof sketch of Theorem 5.

Proof Sketch of Theorem 5. If γ is a non-negative real number, then the result follows from [13, Theorem 1.3]. So we focus on the case where γ is not real. Using the telescoping expansion of $Z_G(\gamma)$ described in (1), it suffices to give an algorithm that on an input graph $G \in \mathcal{F}_{\Delta,a,c}$, a vertex v in G and rational $\delta > 0$ outputs in time $(\hat{c}n/\delta)^{O\left((1+a+\sqrt{|\hat{\gamma}|\Delta})\log\Delta\right)}$ a quantity $\tilde{p}$ which satisfies $\tilde{p} = p_v(G, \gamma)\mathrm{e}^z$ for some complex number z with $|z| \leq \delta$.

Let $T = T_{SAW}(v, G)$ be the self-avoiding walk tree rooted at v, then by Theorem 9 we have that $p_v(G, \gamma) = p_v(T, \gamma)$, so it suffices to approximate $p_v(T, \gamma)$. For this, we apply the general framework of Lemma 13 to the recurrence in (10) using the contraction properties proved in Lemma 16. More precisely, we first truncate the tree T at logarithmic depth ℓ to obtain a tree T' and we output $\hat{p} = p_v(T', \gamma)$ as our approximation to $p_v(T, \gamma)$. Note that T' has size at most $c\Delta^\ell$ since G has connective constant at most Δ. We then invoke Lemmas 13 and 16 to show that the absolute error between $p_v(T, \gamma)$ and $p_v(T', \gamma)$ decays as $(\Delta^{1/q}\alpha)^\ell$ where $\alpha < 1/\Delta^{1/q}$ is the constant in Lemma 16. By taking $\ell = \Theta(\log n)$, we can therefore make the absolute error as small as an inverse polynomial in n. The absolute error can then be translated to the desired relative error between $\hat{p}$ and $p_v(G, \gamma)$ using the bound in Lemma 14. ◀

4 Proof of hardness results

Let $\gamma_0 = -1/10$ and $\mathcal{G}$ be the set of graphs of maximum degree 3. It is well-known [5, Theorem 3] that the problem of *exactly* computing $Z_G(\gamma_0)$ given an input graph $G \in \mathcal{G}$ is #P-hard. Moreover, by Corollary 8 we have that $Z_G(\gamma_0) > 0$ for all graphs $G \in \mathcal{G}$.

Using an oracle on graphs H of maximum degree Δ for either approximating $Z_H(\gamma)$ multiplicatively or deciding the sign of $Z_H(\gamma)$, we will design a polynomial time algorithm to exactly compute the ratio $\frac{Z_G(\gamma_0)}{Z_{G-e^*}(\gamma_0)}$ for an arbitrary graph $G \in \mathcal{G}$ and an arbitrary edge e^* of G; note that this ratio is well-defined since $Z_{G-e^*}(\gamma_0) > 0$. With such a subroutine at hand, we can compute $Z_G(\gamma_0)$ using self-reducibility techniques; namely, let $e_1, e_2, \ldots, e_m$ be an enumeration of the edges of G and let G_i be the graph where the edges $e_i, \ldots, e_m$ are deleted (note that $G_{m+1} = G$ and G_1 is the empty graph). Then, we have that $Z_G(\gamma_0) = \prod_{i=1}^{m} \frac{Z_{G_{i+1}}(\gamma_0)}{Z_{G_i}(\gamma_0)}$. This yields the #P-hardness results of Theorems 1 and 2.

The most difficult part of designing the subroutine is constructing graph gadgets that have the effect of "changing" the edge activity γ to any desired activity, perhaps with some small error. It is actually important to make the error exponentially small relative to the size of the graph. To formalise these gadget constructions, we will need some definitions. Let $G = (V, E)$ be a graph and $u, v \in V$. Analogously to the notation $Z_{G,u}(\gamma)$ and $Z_{G,\neg u}(\gamma)$ of Section 2, we let $Z_{G,u,v}(\gamma)$ be the contribution to the partition function $Z_G(\gamma)$ from those matchings $M \in \mathcal{M}_G$ such that both u, v are matched in M, and we define $Z_{G,u,\neg v}(\gamma), Z_{G,\neg u,v}(\gamma), Z_{G,\neg u,\neg v}(\gamma)$ similarly.

▶ **Definition 17.** *Fix a real number γ. Given γ, the graph $G = (V, E)$ is said to* implement *the edge activity $\gamma' \in \mathbb{R}$ with* accuracy $\epsilon > 0$ *if there are vertices u, v in G such that $Z_{G,\neg u,\neg v}(\gamma) \neq 0$ and*

1. *u, v have degree one in G and $(u, v) \notin E$,*
2. $\left|\frac{Z_{G,u,\neg v}(\gamma)}{Z_{G,\neg u,\neg v}(\gamma)}\right| \leq \epsilon$, $\left|\frac{Z_{G,\neg u,v}(\gamma)}{Z_{G,\neg u,\neg v}(\gamma)}\right| \leq \epsilon$,
3. $\left|\frac{Z_{G,u,v}(\gamma)}{Z_{G,\neg u,\neg v}(\gamma)} - \gamma'\right| \leq \epsilon$.

We call u, v the terminals of G. If both of Items 2 and 3 hold with $\epsilon = 0$, we say that G implements the edge activity γ' (perfectly).

▶ **Definition 18.** *Let α be a rational number and write $\alpha = p/q$, where p, q are integers such that $\gcd(p, q) = 1$. Then, the* size *of α, denoted by $\text{size}(\alpha)$, is given by $1 + \log(|p| + |q|)$. For $\alpha_1, \ldots, \alpha_t \in \mathbb{Q}$, we denote by $\text{size}(\alpha_1, \ldots, \alpha_t)$ the total of the sizes of $\alpha_1, \ldots, \alpha_t$.*

Our key lemma for designing the subroutine is the following.

▶ **Lemma 19.** *Let $\Delta \geq 3$ be an integer and $\gamma < -\frac{1}{4(\Delta-1)}$ be a rational number.*

There is an algorithm which, on input rational $\gamma' \leq 0$ and $\epsilon > 0$, outputs in $poly(\text{size}(\gamma', \epsilon))$ time a bipartite graph G of maximum degree at most Δ with terminals u, v in the same part of the vertex partition of G so that G implements γ' with accuracy ϵ.

In turn, to prove Lemma 19 it will be simpler to first construct graph gadgets that implement "vertex" activities.

▶ **Definition 22.** *Fix a real number γ. Given γ, the graph $G = (V, E)$ is said to* implement the vertex activity $\lambda \in \mathbb{R}$ *with* accuracy $\epsilon > 0$ *if there is vertex u in G such that*

1. *u has degree one in G,*
2. *$Z_G(\gamma) \neq 0$ and $\left|\frac{Z_{G,\neg u}(\gamma)}{Z_G(\gamma)} - \lambda\right| \leq \epsilon$.*

We call u the terminal of G. If Item 2 holds with $\epsilon = 0$, we say that G implements λ (perfectly).

Our main lemma about implementing vertex activities is as follows.

▶ **Lemma 23.** *Let $\Delta \geq 3$ be an integer and $\gamma < -\frac{1}{4(\Delta-1)}$ be a rational number.*

There is an algorithm which, on input a rational number λ and $\epsilon > 0$, outputs in $poly(\text{size}(\lambda, \epsilon))$ time a bipartite graph G of maximum degree at most Δ that implements the vertex activity λ with accuracy ϵ.

In order to obtain the exponential precision of Lemma 23, we will first show how to implement vertex activities with arbitrarily small constant precision, as formalised in the following lemma.

▶ **Lemma 24.** *Let $\Delta \geq 3$ be an integer and $\gamma < -\frac{1}{4(\Delta-1)}$ be a real number.*

For every $\lambda \in \mathbb{R}$ and $\epsilon > 0$, there is a bipartite graph G of maximum degree at most Δ that implements the vertex activity λ with accuracy ϵ.

To prove Lemma 24, we will need to consider two cases for the value of γ. Namely, for an integer $\Delta \geq 3$, the following subset of the negative reals will be relevant:

$$\mathcal{B}_\Delta = \left\{\gamma \in \mathbb{R} \mid \gamma = -\tfrac{1}{4(\Delta-1)(\cos\theta)^2} \text{ for some } \theta \in (0, \pi/2) \text{ that is a rational multiple of } \pi\right\}. \tag{14}$$

If $\gamma \notin \mathcal{B}_\Delta$ then we show that we can use $(\Delta-1)$-ary trees of appropriate height to obtain the constant accuracy of Lemma 24. The situation is more complicated for $\gamma \in \mathcal{B}_\Delta$ since the $(\Delta-1)$-ary tree is not as effective; neverthless, it can still be used to show that we can implement perfectly the edge activity $\gamma' = -1$ (though in some cases we have to work a bit harder, cf. Lemmas 25 and 27 of the full version). To make use of $\gamma' = -1$, we show in Lemma 26 of the full version that for every rational number λ, there exists a tree of maximum degree $\Delta = 3$ that implements the vertex activity λ and this can then be propagated to get the constant accuracy of Lemma 24 for $\gamma \in \mathcal{B}_\Delta$.

We then bootstrap Lemma 24 to obtain the exponential precision required in Lemma 23, based on the "contracting maps that cover" technique of [4]. The key is to build a finite collection of maps $\Phi_i : x \mapsto \frac{1}{1+\gamma(\lambda_i+x)}$ for different values of λ_i (obtained from Lemma 24) and to apply these iteratively to amplify precision on an appropriately chosen interval of the real axis; the details of the construction as well as the choice of the λ_i's (see Lemma 30 below) depend heavily on the fact that we are working with the matching polynomial. Using the maps Φ_i, Lemma 23 can then be proved using a careful analysis depending on the value of λ (relative to the interval) and, once this is in place, we have everything we need to prove Lemma 19, see the full version for details.

The proof of the hardness results for graphs with bounded connective constant (Theorem 4) can be obtained by adapting our arguments above. Namely, we let $S = \bigcup_{d\geq 3} \mathcal{B}_d$. Then, for $\gamma \in S$, as we discussed earlier, there exists a tree that implements the edge activity $\gamma' = -1$. We can then modify the tree so that the terminals of the final tree are at distance ℓ, for arbitrarily large ℓ. The key now is that we can attach the new tree to the edges of a target graph to modify the edge activity and at the same time reduce its connective constant (since there is just one path connecting the terminals of the tree gadget). Since S is dense on the negative real axis, we therefore obtain Theorem 4 by applying the hardness results of Theorems 1 and 2.

We conclude by giving the deferred construction of the maps Φ_i which are used to obtain the exponential precision of Lemma 23. The proof of the following lemma establishes important properties of the maps that enable them to bootstrap precision. The lemma also gives an algorithm that can be used to implement any "target" y with exponential precision.[4]

▶ **Lemma 30.** *Let $\gamma < 0$ be a rational number. Then, there exist rationals x_0 and $r, \delta > 0$ and reals $\lambda_1^*, \ldots, \lambda_t^*$ (for some positive integer t) such that the following holds for all rational $\lambda_1, \ldots, \lambda_t$ satisfying $|\lambda_i - \lambda_i^*| \leq \delta$ for $i \in [t]$.*

Let $I := [x_0 - r, x_0 + r]$ and, for $i \in [t]$, consider the map $\Phi_i : x \mapsto \frac{1}{1+\gamma(\lambda_i+x)}$ for $x \neq -(1+\gamma\lambda_i)/\gamma$. There is an algorithm which, on input (i) a starting point $y_0 \in I \cap \mathbb{Q}$, (ii) a target $y \in I \cap \mathbb{Q}$, and (iii) a rational $\epsilon > 0$, outputs in poly$(\text{size}(y_0, y, \epsilon))$ time a number $\hat{y} \in I \cap \mathbb{Q}$ and a sequence $i_1, i_2, \ldots, i_k \in [t]$ such that

$$\hat{y} = \Phi_{i_k}(\Phi_{i_{k-1}}(\cdots \Phi_{i_1}(y_0)\cdots)) \text{ and } |\hat{y} - y| \leq \epsilon.$$

Proof. Let x_1, x_2 be rationals such that $\gamma x_1 x_2 = -1$ and $x_1 \neq \pm x_2$. Let λ be such that $1 + \gamma\lambda = -\gamma(x_1 + x_2)$. Then, the fixpoints of the map $\Phi : x \mapsto \frac{1}{1+\gamma(\lambda+x)}$ are x_1 and x_2, and at least one of the two points is attracting.[5] Denote by x_0 the attracting fixpoint of Φ, so

[4] The "target" y corresponds to a vertex activity λ – the only difference is that a transformation between them has been applied for technical reasons, see proof of Lemma 23 in the full version for details.

[5] To see this, note that $\Phi(x) = x$ is equivalent to $x(1+\gamma\lambda) + \gamma x^2 = 1$ and therefore x_1 and x_2 are (the only) fixpoints of Φ. Moreover, we have that $\Phi'(x) = -\frac{\gamma}{(1+\gamma(\lambda+x))^2}$ and hence $\Phi'(x_1) = -\gamma x_1^2$,

that x_0 satisfies $\Phi(x_0) = x_0$ and $0 < |\Phi'(x_0)| < 1$. By Lemma 21 of the full version, we can compute $\eta > 0$ such that for all $x \in [x_0 - \eta, x_0 + \eta]$ and all $\lambda' \in [\lambda - \eta, \lambda + \eta]$ it holds that

$$1 + \gamma(\lambda' + x) \neq 0 \text{ and } \left| \frac{\gamma}{\big(1+\gamma(\lambda'+x)\big)^2} - \frac{\gamma}{\big(1+\gamma(\lambda+x_0)\big)^2} \right| \leq \frac{1}{2} \min\big\{|\Phi'(x_0)|, 1 - |\Phi'(x_0)|\big\}. \quad (15)$$

Let $r := \frac{|\Phi'(x_0)|}{4}\eta$, $\delta := (r/4)$ and let $\lambda_1^*, \ldots, \lambda_t^*$ form a δ-covering of the interval $[\lambda - \eta/2, \lambda + \eta/2]$. Let $\lambda_1, \ldots, \lambda_t$ be arbitrary rationals satisfying $|\lambda_i - \lambda_i^*| \leq \delta$. For $i \in [t]$ consider the maps $\Phi_i : x \mapsto \frac{1}{1+\gamma(\lambda_i+x)}$. Finally, let I be the interval $[x_0 - r, x_0 + r]$. We will show
Property 1: The maps $\{\Phi_i\}_{i\in[t]}$ are contracting on the interval I, and
Property 2: $I \subseteq \Phi_1(I) \cup \cdots \cup \Phi_t(I)$.
Once these two properties of the maps $\{\Phi_i\}_{i\in[t]}$ are proved, the algorithm in the statement of the lemma and its analysis are almost identical to those in [4, Proof of Lemmas 12 & 26]. The only difference here is that the maps $\{\Phi_i\}_{i\in[t]}$ have different expressions. The fact that we need about the expression of the maps is that, for $i \in [t]$ and for every rational x, $\Phi_i^{-1}(x)$ can be computed in time $poly(\text{size}(x, \lambda_i, \gamma))$. This is clear since $\Phi_i^{-1}(x) = \frac{1}{\gamma}(\frac{1}{x} - 1) - \lambda_i$.

Proof of Property 1. Fix $i \in [t]$. We will show that Φ_i is contracting on the interval I. Observe that $r < \eta/4$ since $|\Phi'(x_0)| < 1$ and therefore $\delta < \eta/4$ as well. Then, we have by the triangle inequality that

$$|\lambda_i - \lambda| \leq |\lambda_i - \lambda_i^*| + |\lambda_i^* - \lambda| \leq \delta + \eta/2 < \eta.$$

Therefore, we can apply (15) to $\lambda' = \lambda_i$ and $x \in I$. Observe that $\Phi'(x) = -\gamma/\big(1+\gamma(\lambda_i+x)\big)^2$ and $\Phi'(x_0) = -\gamma/\big(1+\gamma(\lambda_i+x_0)\big)^2$ and hence we obtain that for all $x \in I$ it holds that

$$|\Phi_i'(x)| \leq \frac{1}{2}(1 + |\Phi'(x_0)|) < 1.$$

It follows that the maps Φ_i are contracting on the interval I for all $i \in [t]$. ◀

Proof of Property 2. It suffices to consider an arbitrary $y \in I$ and show that there exists $j \in [t]$ such that $\Phi_j^{-1}(y) \in I$. To do this, we set J to be the interval $[x_0 - \eta/2, x_0 + \eta/2]$ and consider the map Φ on the interval J. Then, (15) for $\lambda' = \lambda$ and $x \in J$ gives that

$$0 < \frac{1}{2}|\Phi'(x_0)| \leq |\Phi'(x)|,$$

and therefore, by the Mean Value Theorem, for $z, w \in J$ we have that

$$\frac{1}{2}|\Phi'(x_0)| \cdot |z - w| \leq |\Phi(z) - \Phi(w)|. \quad (16)$$

We thus have that

$$|\Phi(x_0 + \eta/2) - x_0| = |\Phi(x_0 + \eta/2) - \Phi(x_0)| \geq \eta|\Phi'(x_0)|/4 = r,$$
$$|\Phi(x_0 - \eta/2) - x_0| = |\Phi(x_0 - \eta/2) - \Phi(x_0)| \geq \eta|\Phi'(x_0)|/4 = r.$$

Since Φ is monotonically increasing and continuous on the interval J, we therefore obtain that $I \subseteq \Phi(J)$. Therefore, for arbitrary $y \in I$ it holds that $\Phi^{-1}(y) \in J$ and hence from (16) applied to $z = \Phi^{-1}(y)$ and $w = \Phi^{-1}(x_0)$, we obtain that

$$|\Phi^{-1}(y) - x_0| = |\Phi^{-1}(y) - \Phi^{-1}(x_0)| \leq (2/|\Phi'(x_0)|)(y - x_0) \leq \eta/2.$$

$\Phi'(x_2) = -\gamma x_2^2$. Therefore $|\Phi'(x_1)| \neq |\Phi'(x_2)|$ and $1 = |\gamma x_1 x_2| = \sqrt{|\Phi'(x_1)||\Phi'(x_2)|}$. Therefore either $|\Phi'(x_1)| < 1$ or $|\Phi'(x_2)| < 1$.

Since $\lambda_1^*, \ldots, \lambda_t^*$ is a δ-covering of the interval $[\lambda - \eta/2, \lambda + \eta/2]$, it follows that there exists $j \in [t]$ such that

$$\left|\lambda + \Phi^{-1}(y) - x_0 - \lambda_j^*\right| \leq \delta = r/4.$$

Now, observe that $\Phi_j^{-1}(y) = \frac{1}{\gamma}\left(\frac{1}{y} - 1\right) - \lambda_j$ and $\Phi^{-1}(y) = \frac{1}{\gamma}\left(\frac{1}{y} - 1\right) - \lambda$, so we have that

$$\begin{aligned}\left|\Phi_j^{-1}(y) - x_0\right| &= \left|\frac{1}{\gamma}\left(\frac{1}{y} - 1\right) - \lambda_j - x_0\right| = \left|\lambda + \Phi^{-1}(y) - x_0 - \lambda_j\right| \\ &\leq \left|\lambda + \Phi^{-1}(y) - x_0 - \lambda_j^*\right| + \left|\lambda_j - \lambda_j^*\right| \leq r/4 + r/4 = r/2.\end{aligned}$$

It follows that $y \in \Phi_j(I)$ and therefore, since y was arbitrary, we have that $I \subseteq \Phi_1(I) \cup \cdots \cup \Phi_t(I)$. ◀

This completes the proof of Properties 1 and 2, and hence the proof of Lemma 30. ◀

References

1. A. Barvinok. Computing the Permanent of (Some) Complex Matrices. *Foundations of Computational Mathematics*, 16(2):329–342, 2016.
2. A. Barvinok. *Combinatorics and Complexity of Partition Functions*. Algorithms and Combinatorics. Springer International Publishing, 2017.
3. M. Bayati, D. Gamarnik, D. A. Katz, C. Nair, and P. Tetali. Simple Deterministic Approximation Algorithms for Counting Matchings. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 122–127, 2007.
4. I. Bezáková, A. Galanis, L. A. Goldberg, and D. Štefankovič. Inapproximability of the Independent Set Polynomial in the Complex Plane. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*, pages 1234–1240, 2018. Full version available from `arxiv:1711.00282`.
5. J.-Y. Cai, S. Huang, and P. Lu. From Holant to #CSP and Back: Dichotomy for Holant^c Problems. *Algorithmica*, 64(3):511–533, 2012.
6. C. D. Godsil. Matchings and Walks in Graphs. *Journal of Graph Theory*, 5(3):285–297, 1981.
7. Nicholas J. A. Harvey, Piyush Srivastava, and Jan Vondrák. Computing the Independence Polynomial: from the Tree Threshold down to the Roots. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 1557–1576, 2018.
8. O. J. Heilmann and E. H. Lieb. Theory of Monomer-Dimer Systems. *Communications in Mathematical Physics*, 25(3):190–232, 1972.
9. M. Jerrum and A. Sinclair. Approximating the Permanent. *SIAM J. Comput.*, 18(6):1149–1178, 1989.
10. D. Kraus and O. Roth. Conformal Metrics. *CoRR*, abs/0805.2235, 2008. `arXiv:0805.2235`.
11. V. Patel and G. Regts. Deterministic Polynomial-Time Approximation Algorithms for Partition Functions and Graph Polynomials. *SIAM Journal on Computing*, 46(6):1893–1919, 2017.
12. H. Peters and G. Regts. On a Conjecture of Sokal Concerning Roots of the Independence Polynomial. *The Michigan Mathematical Journal*, 2019. To appear. URL: `https://projecteuclid.org/euclid.mmj/1541667626#info`.
13. A. Sinclair, P. Srivastava, D. Štefankovič, and Y. Yin. Spatial Mixing and the Connective Constant: Optimal Bounds. *Probability Theory and Related Fields*, 168(1):153–197, 2017.
14. A. Sinclair, P. Srivastava, and Y. Yin. Spatial Mixing and Approximation Algorithms for Graphs with Bounded Connective Constant. In *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, FOCS '13, pages 300–309, 2013.
15. B.-B. Wei, S.-W. Chen, H.-C. Po, and R.-B. Liu. Phase Transitions in the Complex Plane of Physical Parameters. *Nature Scientific Reports*, 4:5202; DOI: 10.1038/srep05202, 2014.
16. D. Weitz. Counting Independent Sets Up to the Tree Threshold. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, STOC '06, pages 140–149, New York, NY, USA, 2006. ACM.

Finding Tutte Paths in Linear Time

Therese Biedl
David R. Cheriton School of Computer Science, University of Waterloo, Canada
https://cs.uwaterloo.ca/~biedl/
biedl@uwaterloo.ca

Philipp Kindermann
Lehrstuhl für Informatik I, Universität Würzburg, Germany
https://go.uniwue.de/pkinderm
philipp.kindermann@uni-wuerzburg.de

Abstract

It is well-known that every planar graph has a *Tutte path*, i.e., a path P such that any component of $G-P$ has at most three attachment points on P. However, it was only recently shown that such Tutte paths can be found in polynomial time. In this paper, we give a new proof that 3-connected planar graphs have Tutte paths, which leads to a linear-time algorithm to find Tutte paths. Furthermore, our Tutte path has special properties: it visits all exterior vertices, all components of $G-P$ have exactly three attachment points, and we can assign distinct representatives to them that are interior vertices. Finally, our running time bound is slightly stronger; we can bound it in terms of the degrees of the faces that are incident to P. This allows us to find some applications of Tutte paths (such as binary spanning trees and 2-walks) in linear time as well.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases planar graph, Tutte path, Hamiltonian path, 2-walk, linear time

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.23

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at [5]: http://arxiv.org/abs/1812.04543.

Funding *Therese Biedl*: Supported by NSERC.

1 Introduction

A *Tutte path* is a well-known generalization of Hamiltonian paths that allows to visit only a subset of the vertices of the graph, as long as all remaining vertices are in components with at most three attachment points. (Detailed definitions are below.) They have been studied extensively, especially for planar graphs, starting from Tutte's original result:

▶ **Theorem 1** (Tutte [19]). *Let G be a 2-connected planar graph with distinct vertices X, Y on the outer face. Let α be an edge on the outer face. Then G has a Tutte path from X to Y that uses edge α.*

We refer to the recent work by Schmid and Schmidt [15] for a detailed review of the history and applications of Tutte paths. It was long not known how to compute a Tutte path in less than exponential time. A breakthrough was achieved by Schmid and Schmidt in 2015 [13, 14], when they showed that one can find a Tutte path for 3-connected planar graphs in polynomial time. In 2018, the same authors then argued that Tutte paths can be found in polynomial time even for 2-connected planar graphs [15]. For both papers, the main insight is to prove the existence of a Tutte path by splitting the graph into non-overlapping subgraphs to recurse on; the split can be found in linear time and therefore the running time becomes quadratic.


In this paper, we show that Tutte paths can be computed in linear time. To do so, we give an entirely different proof of the existence of a Tutte path for 3-connected planar graph. This proof is very simple if the graph is triangulated, but requires more care when faces have larger degrees. Our path (and also the one by Schmid and Schmidt [13, 14]) comes with a system of distinct representatives, i.e., an injective assignment from the components of $G \setminus P$ to vertices of P that are attachment points. Such representatives are useful for various applications of Tutte paths.

Our proof for 3-connected planar graphs is based on a Hamiltonian-path proof by Asano, Kikuchi and Saito [1] that was designed to give a linear-time algorithm; with arguments much as in their paper we can therefore find the Tutte path and its representatives in linear time. Since 3-connected planar graphs are (as we argue) the bottleneck in finding Tutte paths, this shows that the path of Theorem 1 can be found in linear time.

1.1 Preliminaries

We assume familiarity with graphs, see, e.g., Diestel [7]. Throughout this paper, $G = (V, E)$ denotes a graph with n vertices and m edges. We assume that G is *planar*, i.e., can be drawn in 2D without edge crossings. A planar drawing of G splits $\mathbb{R}^2$ into connected regions called *faces*; the unbounded region is the *outer face* while all others are called *interior faces*. A vertex/edge is called *exterior* if it is incident to the outer face and *interior* otherwise. We assume throughout that G is *plane*, i.e., one particular abstract drawing of G has been fixed (by giving the clockwise order of edges around each vertex and the edges that are on the outer face). Any subgraph of G *inherits* this planar embedding, i.e., uses the induced order of edges and as outer face the face that contained the outer face of G. The following notion will be convenient: Two vertices v and w are *interior-face-adjacent* (in a plane graph G) if there exists an interior face that is incident to both v and w. We will simply write *face-adjacent* since we never consider adjacency via the outer face.

Nooses and connectivity. For a fixed planar drawing of G, let a *noose* be a simple closed curve $\mathcal{N}$ that goes through vertices and faces and crosses no edge except at endpoints. A noose can be described as a cyclic sequence $\langle x_0, f_1, x_1, \ldots, f_s, x_s{=}x_0\rangle$ of vertices and faces such that f_i contains x_{i-1} and x_i, and hence is independent of the chosen drawing. Frequently, the choice of faces will be clear from context or irrelevant; we then say that $\mathcal{N} = \langle x_0, \ldots, x_s{=}x_0\rangle$ *goes through* $\{x_1, \ldots, x_s\}$. The subgraph *inside/outside* $\mathcal{N}$ is the graph induced by the vertices that are on or inside/outside $\mathcal{N}$. The subgraph *strictly inside/outside* is obtained from this by deleting the vertices on $\mathcal{N}$.

A graph G is *connected* if for any two vertices v, w there is a path from v to w in G. A *cutting k-set* in G is a set $S = \{x_1, \ldots, x_k\}$ of vertices such that $G \setminus S$ has more connected components than G. We call it a *cutting pair* for $k = 2$ and a *cutting triplet* for $k = 3$. A graph G is called *k-connected* if it has no cutting $(k-1)$-set. Since we are only studying planar graphs, it will be convenient to use a characterization of connectivity via nooses. Consider a noose $\mathcal{N}$ that goes through $\{x_1, \ldots, x_k\}$ (and no other vertices), and there are vertices both strictly inside and strictly outside $\mathcal{N}$. Then clearly $S = \{x_1, \ldots, x_k\}$ is a cutting k-set. Vice versa, in a planar graph, any cutting k-set S for $k = 1, 2, 3$ gives rise to a noose $\mathcal{N}$ through S that has vertices both strictly inside and strictly outside. A *strict cut component* C of S is a subgraph strictly inside a noose $\mathcal{N}$ through some of the vertices of S such that C contains at least one vertex not in S and is inclusion-minimal among all such nooses. In particular, a strict cut component C contains no vertices or edges of S. A *cut component* C^+ is obtained from a strict cut component C by re-inserting those vertices of S that have neighbours of C, as well as the edges from them to C.

Hamiltonian paths and Tutte paths. A *Hamiltonian path* is a path that visits every vertex exactly once. To generalize it to Tutte paths, we need more definitions. Fix a path P in the graph. A *P-bridge* C is a cut component of P; its *attachment points* are its vertices on P.[1] A *Tutte path* is a path P such that any P-bridge C has at most three attachment points, and if C contains exterior edges, then it has at most two attachment points. Our Tutte paths for 3-connected graphs will be such that no P-bridges contain exterior edges, so the second restriction holds automatically.

A Tutte path with a *system of distinct representatives* (SDR), also called a *T_{SDR}-path* for short, is a Tutte path P together with an injective assignment σ from the P-bridges to vertices in P such that for every P-bridge C vertex $\sigma(C)$ is an attachment point of C.

Given a path P in a plane graph, we denote by $F(P)$ the set of all interior faces that contain at least one vertex of P.

1.2 From 3-connected to 2-connected

In this section, we show that, to find the path of Theorem 1 efficiently, it suffices to consider 3-connected planar graphs.

We re-prove Theorem 1, presuming it holds for 3-connected planar graphs, by induction on the number of vertices with an inner induction on the number of exterior vertices. Say we want to find a Tutte path from X to Y that uses exterior edge $\alpha = (U, W)$, where X, Y are exterior vertices. In the base case, G is 3-connected and we are done. So assume that G has cutting pairs. If edge (X, Y) does not exist, then add it in such a way that α stays exterior, and find a Tutte path P in the resulting graph recursively (it has fewer exterior vertices). Since $\{X, Y\} \neq \{U, W\}$ (because $(U, W) \in G$ while $(X, Y) \notin G$), path P visits at least one vertex other than X, Y, and so cannot use edge (X, Y). So it is also a Tutte path of G.

Now, assume that (X, Y) exists. Repeatedly split the graph at any cutting pair $\{u, v\}$ into cut components $C_1, \ldots, C_k$, and store the *3-connected components* $C_1^+, \ldots, C_k^+$ (induced by the cut components with an additional virtual edge between the cutting pairs) in a so-called *SPQR-tree* [6, 9], which additionally creates one leaf node for every edge of G. This can be done in linear time [10].

Root the SPQR-tree at the node of edge (X, Y). For each 3-connected component C^+ other than the root, set $\{X_C, Y_C\}$ to be the cutting pair that C^+ has in common with its parent component, and observe that these two vertices are necessarily exterior in C since X, Y are exterior in G; see Figure 1.

If C^+ has only these two vertices, then let P_C be the path (X_C, Y_C). Otherwise, define an edge $\alpha_C \neq (X_C, Y_C)$ of C^+ as follows: If the node of α is a descendant of C^+, then let α_C be the virtual edge of C^+ that it shares with the child that leads to this descendant. Note that α_C is a virtual edge, and it is necessarily on the outer face of C since α is on the outer face of G. Otherwise (α is not in a descendant of C^+) choose α_C to be an arbitrary exterior edge of C other than (X_C, Y_C). Let P_C be a Tutte path that begins at X_C, ends at Y_C, and uses edge α_C; we know that this exists since C^+ is either a triangle or a 3-connected graph.

Now obtain the Tutte path P of G by repeatedly substituting paths of 3-connected components. Specifically, initiate P as the virtual copy of edge (X, Y) that was added when we created the node for (X, Y). For as long as P contains a virtual edge (u, v), let C^+ be the child component at this virtual edge and observe that $\{X_C, Y_C\} = \{u, v\}$. *Substitute P_C in place of edge (u, v) of P*, i.e., set P to be $X \overset{P}{\rightsquigarrow} u/v \overset{P_C}{\rightsquigarrow} v/u \overset{P}{\rightsquigarrow} Y$. Note that, if C^+ is not

[1] Our definition of P-bridge considers only the *proper* P-bridges [19] that contain at least one vertex.

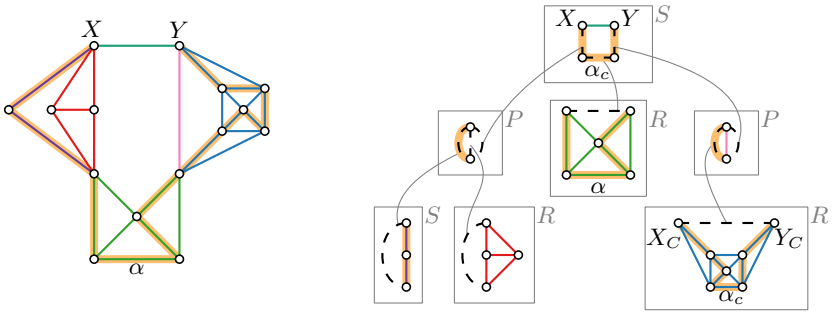


Figure 1 A 2-connected graph, its SPQR-tree (leaf nodes are omitted), and its Tutte path.

the singleton-edge (u, v), then P_C contains α_C, which is a virtual edge. This means that the process repeats until we have substituted the real edges from the leaves of the SPQR-trees. In particular (due to our choice of α_C), we will substitute the paths from all components between (X, Y) and α, which means that α is an edge of the final path P as required.

Observe that for some 3-connected components we do not substitute their paths; these become P-bridges with two attachment points. There may also be some P-bridges within each 3-connected components, but these have at most three attachment points since we used a Tutte path for each component. So the result is the desired Tutte path. Since we compute one Tutte path per 3-connected component, and this can be done in time proportional to the size of the component, the overall running time is linear.

2 Tutte paths in 3-connected planar graphs

For triangulated planar graphs, one can quite easily find a T_{SDR}-path by removing the interiors of all separating triangles, and finding for the resulting 4-connected planar graph a Hamiltonian path using the approach of Asano, Kikuchi, and Saito [1]. It is not hard to see that we can assign representatives to all separating triangles, possibly after expanding the path using the substitution trick described below. (We omit the details for space reasons.)

For 3-connected planar graphs that are not triangulated, we use the same approach, but must generalize many definitions from Asano, Kikuchi, and Saito [1] and add quite a few cases because now face-adjacent vertices are not necessarily adjacent. To keep the proof self-contained, we re-phrase everything from scratch.[2]

We need a few definitions. The *outer stellation* of a planar graph G is the graph obtained by adding a vertex in the outer face and connecting it to all exterior vertices. A planar graph G is called *internally 3-connected* if its outer stellation is 3-connected. Note that this implies that G is 2-connected, any cutting pair is *exterior* (i.e., has both vertices on the outer face) and has only two cut components. In the following, we endow G with k *corners*, which are k vertices $X_1, \ldots, X_k$ that appear in this order on the outer face. Usually, $k = 3$ or 4, but occasionally we allow larger k. A *side* of such a graph is the outer face path between two consecutive corners that does not contain any other corners. The *corner stellation* G^{s} is obtained by adding a vertex in the outer face and connecting it to the corners. We say

[2] Indeed, due to attempts to simplify the notations similar as done in [4], the reader familiar with [1] may barely see the correspondence between the proof and [1]. Roughly, their Condition (W) corresponds to $c3c(X, W, Y)$, their Case 1 is our Case 3a, and their Case 3 combines our Case 2 with our Case 4a (but resolves it in a symmetric fashion).

that G is *corner-3-connected with respect to corners* $X_1, \ldots, X_k$ (abbreviated to "G *satisfies* $c3c(X_1, \ldots, X_k)$") if G^s is 3-connected. Figure 2a illustrates this condition. It is easy to show that G satisfies $c3c(X_1, \ldots, X_k)$ if and only if $k \geq 3$, G is internally 3-connected, and no cutting pair $\{v, w\}$ of G has both v and w on one side of G.

For ease of proof, we make the induction hypothesis stronger than just having a T_{SDR}-path, by restricting which vertices *must* be visited and which vertices *must not* be representatives. A T_{int}*-path* is a T_{SDR}-path P that visits all exterior vertices, and where representative $\sigma(C)$ is interior, for all P-bridges C. The goal of the remainder of this section is to prove the following result (which immediately implies Theorem 1 for 3-connected graphs[3]):

▶ **Lemma 2.** *Let G be a plane graph with distinct vertices X, Y on the outer face. Let $(U, W) \neq (X, Y)$ be an edge on the outer face. If G satisfies $c3c(X, U, W, Y)$, then it has a T_{int}-path that begins at X, ends at Y, and contains (U, W).*

We need a second result for the induction. Let a T_{end}*-path* be a T_{SDR}-path P that visits all exterior vertices, and where representative $\sigma(C)$ is interior or the last vertex of P, for all P-bridges C.

▶ **Lemma 3.** *Let G be a plane graph with distinct vertices X, Y on the outer face. Let $(U, W) \neq (X, Y)$ be an edge on the outer face. If G satisfies $c3c(X, U, W, Y)$ and*

($\sqsupset$) *(W, Y) and (Y, X) are edges,*

then G has a T_{end}-path P that begins at X, ends at Y, and uses (U, W) and (W, Y).[4] *Further, if Y is the representative of a P-bridge C, then C has W and Y as attachment points.*

See Figure 5c for a graph that satisfies ($\sqsupset$).

We assume throughout that X, U, W, Y are enumerated in ccw order along the outer face, the other case can be resolved by reversing the planar embedding.

The following trick will help shorten the proof: If graph G satisfies ($\sqsupset$), then Lemma 3 implies Lemma 2. Namely, if Lemma 3 holds, then we have a T_{end}-path P from X to Y through (U, W) and (W, Y). If this is not a T_{int}-path, then some P-bridge C has Y as representative, and by assumption also has W as attachment point. It must have a third attachment point u, otherwise $\{W, Y\}$ would be a cutting pair within one side of G, contradicting corner-3-connectivity. It has no more attachment points since P is a Tutte path, so $\{W, Y, u\}$ is a cutting triplet. We apply the *substitution trick* described below (and useful in other situations as well), which replaces (W, Y) with a path through C that does not use u. Thus, C no longer needs a representative and we obtain a T_{int}-path.

The substitution trick. This trick can be applied whenever we have an edge $e = (w, y)$ used by some T_{SDR}-path P, and a P-bridge C that resides inside a noose through some cutting triplet $\{u, w, y\}$ for some vertex u. Define $C^+ = G[C] \cup \{(u, w), (w, y)\} \setminus \{(u, y)\}$, where edges are added only if they did not exist in $G[C]$.[5]

[3] Theorem 1 allows $(U, W) = (X, Y)$, but then holds trivially since using edge (X, Y) as path satisfies all conditions. We require $(U, W) \neq (X, Y)$ since we want not just a Tutte path but a T_{int}-path, and the single-edge path (X, Y) would allow only exterior vertices as representatives.

[4] This lemma is a special case of the "Three Edge Lemma" [17], which states that for any three edges on the outer face there exists a Tutte cycle containing them all. However, it cannot simply be obtained from it since we require restrictions on the location of representatives.

[5] We apply the substitution trick even when $V(C^+) = V(G)$ and G has a triangular outer face; not adding edge (w, y) will ensure that C^+ has fewer interior vertices and induction can be applied.

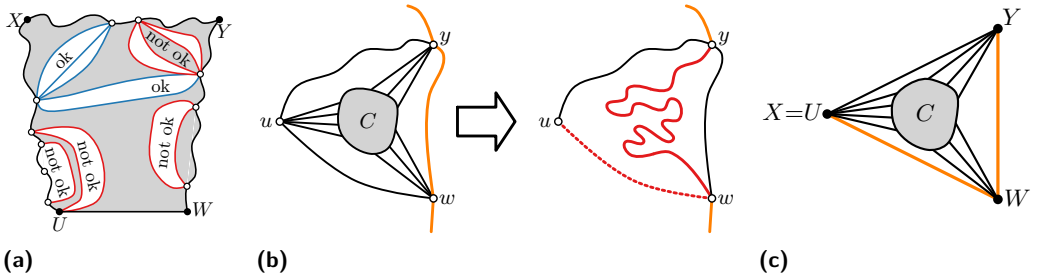


Figure 2 (a) Corner-3-connectivity $c3c(X, U, W, Y)$, (b) the substitution trick, and (c) Case 1.

C^+ satisfies $c3c(u, v, w)$, else there would have been a cutting pair in G that was interior or within one side. Hence, by induction, C^+ has a T_{int}-path P_{C^+} from u to y that uses edge (u, w). It does not use the edge (u, y) since P_{C^+} begins at u with edge (u, w). So $P_{C^+} \setminus (u, w)$ is a path in C from w to y that does not visit u. Substitute this in place of edge (w, y) of P; see Figure 2b. We claim that the resulting path P' is a T_{int}-path. We prove a more general statement in the full version [5], but roughly speaking, combining paths preserves T_{int}-paths because every P'-bridge can inherit its representative from P or P_{C^+}, and no vertex is used twice as representative since P_{C^+} does not use $\{u, v, w\}$ as representatives.

2.1 Proof of Lemma 2 and Lemma 3

We prove the two lemmas simultaneous by induction on the number of vertices of G, with an inner induction on the number of interior vertices. The base case is $n = 3$ where G is a triangle, but the same construction works whenever the outer face is a triangle (see below). For the induction step, we need the notation S_{xy}, which is the outer face path from x to y in ccw direction. In particular, the four sides are S_{XU}, S_{UW}, S_{WY}, and S_{YX}. We sometimes name sides as suggested by Figure 2a, so S_{XU}, S_{UW}, S_{WY}, and S_{YX} are the *left/bottom/right/top side*, respectively.

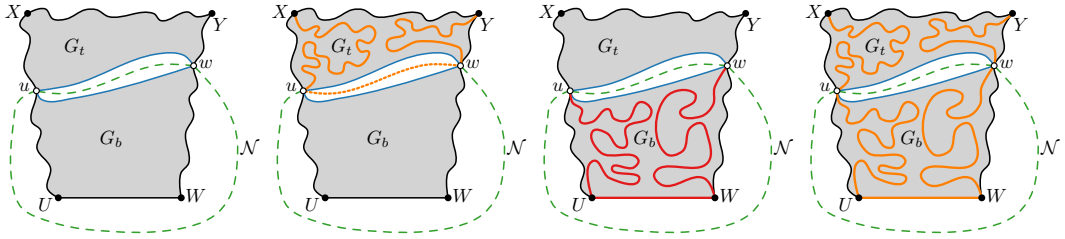
2.1.1 Case 1: The outer face is a triangle

Figure 2c illustrates this case. We know that $X \neq Y$ and $U \neq W$, so we must have $X = U$ or $W = Y$. For Lemma 3, we know that ($\sqsupset$) holds, which forces $W \neq Y$, hence $X = U$. For Lemma 2, we may assume $X = U$ by symmetry, for otherwise we reverse the planar embedding, find a path from Y to X that uses (W, U) (with this, we have $X' = U'$) and then reverse the result.

So $X = U$. Define P to be $\langle X{=}U, W, Y\rangle$ and observe that this is a T_{end}-path, because the unique P-bridge C (if any) has attachment points $\{U, W, Y\}$, and we can assign Y to be its representative. So Lemma 3 holds. Since condition ($\sqsupset$) is satisfied, this implies Lemma 2.

2.1.2 Case 2: G has a cutting pair $\{u, w\}$ with u and w on the left and right side

Figure 3 illustrates this case. Let $\mathcal{N}$ be a noose through u and w along a common interior face f^* and then going through the outer face. Let G_t and G_b be the subgraphs inside and outside $\mathcal{N}$, named such that G_b contains the bottom side. Let G_t^+/G_b^+ be the graphs obtained from G_t/G_b by adding (u, w) if not in the graph yet. We add (u, v) even if it did not exist in G (we will ensure that the final path does not use it).



(a) Case 2 **(b)** The path P_t **(c)** The path P_b **(d)** The path P

Figure 3 (a) Case 2, (b)–(d) proof of Lemma 2 for Case 2.

We first show Lemma 2. One can easily verify that G_t satisfies $c3c(X, u, w, Y)$ since its outer face is a simple cycle; see the full version [5]. Apply induction and find a T_{int}-path P_t of G_t^+ from X to Y that uses edge (u, w). Now apply a modified substitution trick to (u, w). Namely, by induction, there is a T_{int}-path P_b of G_b^+ from u to w that uses edge (U, W). Substitute P_b into P_t in place of (u, w) to get P. Path P uses (U, W) since P_b does. It does not use (u, w) since we removed this from P_t, and P_b starts at u, ends at w, and visits (U, W) in between. So after inheriting representatives from P_b and P_t we obtain a T_{int}-path P in G.

To prove Lemma 3, note that exactly one of G_t^+ and G_b^+ contains (W, Y); use a T_{end}-path for this subgraph and create P as above. Only one graph uses Y as representative, and one easily shows that P is a T_{end}-path.

2.1.3 Case 3: G has a cutting pair $\{y, w\}$ with y and w on the top and right side, respectively. Furthermore, there is an interior face f^* containing y and w that does not contain Y.

For later applications, we first want to point out that if G has a cutting pair $\{y, w\}$ on the top and right side for which (y, w) is an edge, then such a face f^* always exists, because there are two interior faces containing y and w, and not both can contain Y.

Figure 4 illustrates this case. We know that $w \neq Y \neq y$, else $\{y, w\}$ would be a cutting pair within one side. We may assume $y \neq X$; else we can use Case 2. Hence, the top side contains at least three vertices X, y, Y, so $(\sqsupset)$ does not hold and we have to prove only Lemma 2.

We choose $\{y, w\}$ such that w is as close to W as possible (along the right side). The face f^* containing y, w may have multiple edges on the top side; let y be the one that is as close to Y as possible. Define G_b, G_b^+, G_t, G_t^+ to be as in Case 2. Since the outer face of G_b^+ is a simple cycle, it satisfies $c3c(X, U, W, w, y)$. But since we chose w to be as close to W as possible, it also satisfies $c3c(X, U, W, y)$. Namely, assume for contradiction that some cutting pair $\{y', w'\}$ exists along the side $S_{Ww} \cup (w, y)$ of G_b^+; see Figure 4a. Since there is no cutting pair within S_{Ww}, it must have the form $\{y, w'\}$ for some $w' \neq w$ on S_{Ww}. As f^* does not contain Y, neither can any face containing $\{y, w'\}$, so $\{y, w'\}$ could have been used for Case 3, contradicting our choice of w.

By induction, we can find a T_{int}-path P_b of G_b^+ from X to y that includes the edge (U, W). The plan is to combine P_b with a path through G_t, but we must distinguish some cases.

Case 3a: P_b does not contain (y, w) or $(y, w) \in G$. Observe that G_t^+ satisfies $c3c(y, w, Y)$. By induction, find a T_{int}-path P_t in G_t^+ from Y to w that uses edge (y, w). Append the reverse of $P_t \setminus (y, w)$ to P_b to obtain a T_{int}-path; see Figure 4b.

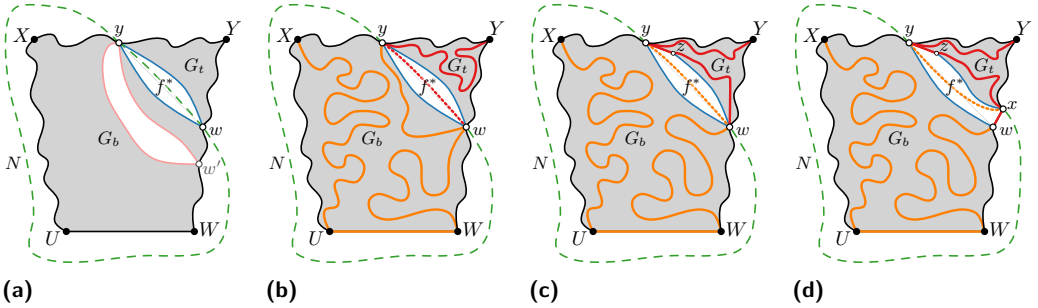


Figure 4 Case 3: (a) G_b^+ satisfies $c3c(X, U, W, y)$, (b) Case 3a, (c) Case 3b-1, (d) Case 3b-2.

Case 3b: P_b contains (y, w) and $(y, w) \notin G$. In this case, we must remove (y, w) from the path and hence use a subpath in G_t to reach vertex y. This requires further subcases. Let π_f be the path along f^* from y to w that becomes part of the the outer face of G_t. Let (y, z) be the edge incident to y on π_f.

Case 3b-1: π_f contains no vertex on the outer face of G other than y and w. See Figure 4c. The outer face of G_t is then a simple cycle and G_t satisfies $c3c(w, y, Y)$. By induction, we can find a T_{int}-path P_t in G_t that begins at Y, ends at w, and uses (y, z).

Case 3b-2: π_f contains a vertex $x \neq y, w$ on the outer face of G. See Figure 4d. Since x is on f^*, it cannot be on the top side by choice of y. So $x \in S_{wY} \setminus Y$. In fact, x must be the neighbor of w on both S_{wY} and π_f, else there would be a cutting pair within the right side. Set G'_t to be the graph inside a noose through y and x that has Y inside. Since π_f has no vertices other than y, x, w on the outer face of G, graph G'_t has a simple cycle as outer face, so it satisfies $c3c(Y, y, z, x)$. By induction, we can find a T_{int}-path P'_t of G_t that begins at Y, ends at x, and uses (y, z). We append (w, x) to obtain P_t.

In both cases, we obtain a path P_t that begins at Y, ends at w, and visits all of G_t. Appending the reverse of this to $P_b \setminus (y, w)$ gives the T_{int}-path.

2.1.4 Case 3′: G has a cutting pair $\{y, w\}$ with y and w on the top and left side, respectively. Furthermore, there is an interior face f^* containing y and w that does not contain X.

This is handled symmetrically to Case 3.

2.1.5 Case 4: None of the above

In this case, we split G into one big graph G_0 and (possibly many) smaller graphs $G_1, \ldots, G_s$, recurse in G_0, and then substitute T_{int}-paths of $G_1, \ldots, G_s$ or use them as P-bridges.

We need two subcases, but first give some steps that are common to both. Let Y_X be the neighbor of Y on the top side. Define a *B-necklace* (for $B \in \{U, W\}$) to be a noose $\mathcal{N}_0 : \langle Y_X{=}x_0, f_1, x_1, \ldots, x_{s-1}, f_s, x_s{=}B, f_o\rangle$, (where f_o is the outer face) for which x_i is face-adjacent to at least one vertex on $S_{WY} \setminus \{B\}$ for $1 \leq i \leq s-1$. See also Figure 5. We say that the necklace is *simple* if it contains no vertex twice, and *interior* if every x_i (for $0 < i < s$) is an interior vertex. One can argue that if none of the previous cases applies, then there always exists a simple interior B-necklace (see the full version [5]).

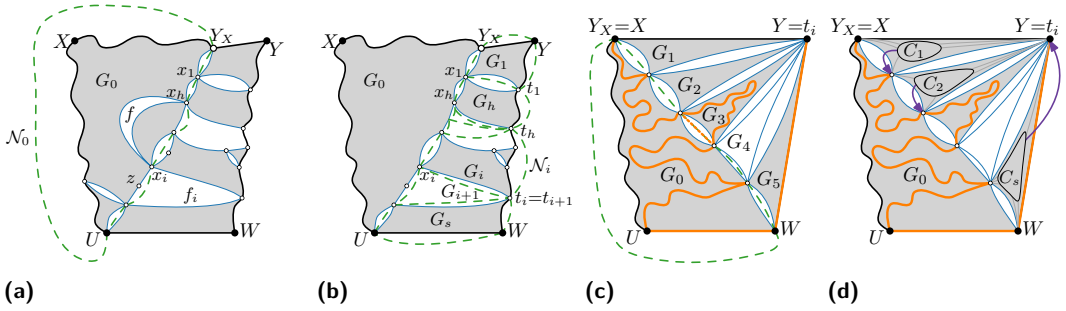


Figure 5 Case 4. (a) A simple interior U-necklace that is not leftmost due to face f (which yields a cutting pair $\{x_h, x_i\}$), and since it could include vertex z. (b) The graphs $G_1, \ldots, G_s$. (c–d) Case 4a. The path P^+ after using the substitution trick and (d) assignment of the representatives.

Route $\mathcal{N}_0$ through the outer face such that the left side is in its interior, and let G_0 (the "left graph") be the graph inside $\mathcal{N}_0$. We say that $\mathcal{N}_0$ is *leftmost* if (among all simple interior B-necklaces) its left graph G_0 is smallest, and (among all simple interior B-necklaces whose left graph is G_0) it contains the most vertices of G_0. Fix a leftmost B-necklace $\langle x_0, \ldots, x_s \rangle$.

▷ Claim 4. If (x_i, x_{i+1}) is not an edge for some $0 \le i < s$, then the face f_i of $\mathcal{N}_0$ contains no vertex of $S_{WY} \setminus \{B\}$.

Proof. If (x_i, x_{i+1}) is not an edge of G, then both paths from x_i to x_{i+1} on f_i contain at least one other vertex. One of them, say z, is inside $\mathcal{N}_0$. If f_i contains a vertex of $S_{WY} \setminus \{B\}$, then x_i and z are face-adjacent, z and x_{i+1} are face-adjacent, and z has a neighbor on $S_{WY} \setminus \{B\}$, so $x_0, \ldots, x_i, z, x_{i+1}, \ldots, x_s$ is a simple interior B-necklace with the same left graph but containing more vertices of G_0. Hence $\mathcal{N}_0$ is not leftmost, a contradiction. ◁

For $i = 0, \ldots, s-1$, let t_i be the vertex on $S_{WY} \setminus \{B\}$ that is face-adjacent to x_i and closest to Y (along the right side) among all such vertices. Set $t_s = W$ if $x_s = U$, and $t_s = Y$ otherwise. For $0 < i \le s$, define $\mathcal{N}_i$ to be the noose through $\langle x_{i-1}, x_i, t_i, t_{i-1} \rangle$ such that the left side is outside $\mathcal{N}_i$. For $0 < i \le s$ let G_i be the graph inside $\mathcal{N}_i$ (i.e., a cut component of $\{x_{i-1}, x_i, t_{i-1}, t_i\}$); see Figure 5b.

Let G^+ be the graph obtained from G by adding *virtual edges* (x_i, x_{i-1}) and (t_i, t_{i-1}) (for $i = 1, \ldots, s$) whenever these two vertices are distinct and the edge did not exist in G. Let G_0^+ be the graph obtained from G_0 by likewise adding virtual edges $(x_0, x_1), \ldots, (x_{s-1}, x_s)$. This makes the outer face of G_0 a simple cycle, so G_0^+ satisfies $c3c(X, U, B{=}x_s, \ldots, x_0{=}Y_X)$. We distinguish two cases.

Case 4a: ($\sqsupset$) holds, i.e., (X, Y) and (W, Y) are edges. We only have to prove Lemma 3 since this implies Lemma 2. Consider Figs. 5c and d. Let $\langle x_0{=}Y_X{=}X, x_1, \ldots, x_s{=}W \rangle$ be a leftmost W-necklace. By $S_{WY} = (W, Y)$, we have $t_i = Y$ for all i. Since $x_0 = Y_X = X$, we have that G_0^+ satisfies $c3c(X{=}x_0, x_1, \ldots, x_s{=}W, U)$. But observe that G_0^+ has no cutting pair $\{x_h, x_i\}$ with $0 \le h < i \le s$, for otherwise the face f containing x_h and x_i could be used as a shortcut and $\mathcal{N}_0$ was not leftmost (see Figure 5a). So G_0^+ actually satisfies $c3c(X, W, U)$. Use induction to obtain a T_{int}-path P_0 from X to W in G_0^+ that uses edge (U, W). Then $P^+ = P_0 \cup (W, Y)$ is a path in G^+ that contains (U, W), and (W, Y).

Fix some $i = 1, \ldots, s$. If P^+ used edge (x_{i-1}, x_i) and it was virtual, then by Claim 4 f_i contains no vertex of S_{WY}, which means that the interior of G_i is non-empty. Apply the substitution trick to remove (x_{i-1}, x_i) from P^+, replacing it with a path through G_i.

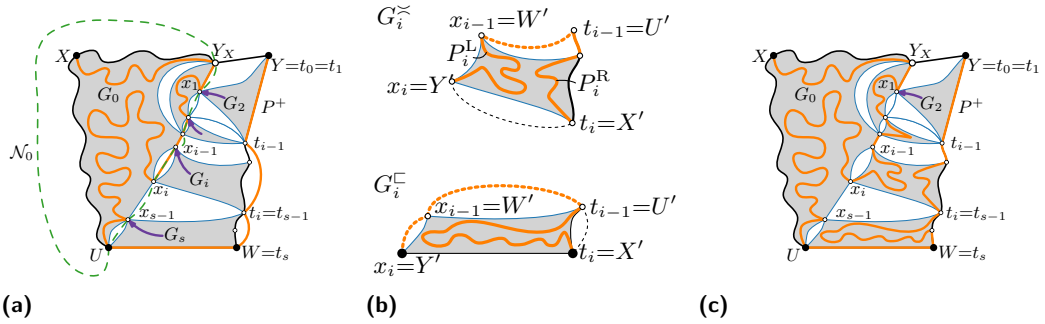


Figure 6 Case 4b. (a) Construction of P^+ with representatives; (b) $G_i^{\asymp}$ (Case 4b-2) and $G_i^{\sqsubset}$ (Case 4b-3) used to substitute virtual edges; (c) P^+ with representatives after all substitutions.

Otherwise, we keep G_i as a P^+-bridge. We let its representative be x_i if $1 \leq i < s$, and Y if $i = s$. Observe that this representative is interior or Y, and was not used by P_0 since P_0 was a T_{int}-path. So we obtain a T_{end}-path with the desired properties.

Case 4b: ($\sqsupset$) does not hold. We must prove only Lemma 2 and may therefore by symmetry assume that $X \neq U$. We claim that this implies that $\deg(Y_X) \geq 3$. For if $\deg(Y_X) = 2$, then its neighbors form a cutting pair, which by corner-3-connectivity means that Y_X is a corner, hence $Y_X = X$. Since $X \neq U$, the two neighbors of Y_X are then Y and a vertex on the left side, and we could have applied Case 2. So $\deg(Y_X) \geq 3$. Let (Y_X, x_1) be the edge at Y_X that comes after (Y_X, Y) in clockwise order (see Figure 6a). Note that x_1 is face-adjacent to Y. It must be an interior vertex, for otherwise by $\deg(Y_X) \geq 3$ edge (Y_X, x_1) is a cutting pair that we could have used for Case 3 or 3′.

Let $\mathcal{N}_0 = \langle x_0{=}Y_X, x_1, \ldots, x_s{=}U\rangle$ be a simple interior U-necklace; see Figure 6a. We use a U-necklace that is leftmost among all U-necklaces that contain x_1. Note that Claim 4 holds for $\mathcal{N}_0$ even with this restriction, since (x_0, x_1) is an edge. We know that G_0^+ satisfies $c3c(X, Y_X, x_1, \ldots, x_s{=}U)$. But observe that G_0^+ has no cutting pair $\{x_h, x_i\}$ for $1 \leq h < i \leq s$, for otherwise (as in Figure 5a) $\mathcal{N}_0$ would not be the leftmost necklace that uses x_1. So G_0^+ actually satisfies $c3c(X, Y_X, x_1, U)$.

Use induction to obtain a T_{int}-path P_0 in G_0 from U to X through edge (x_1, x_0). Append the path $\langle U, W, t_s, \ldots, t_0{=}Y\rangle$ to the reverse of P_0 to obtain path P^+. This path begins at X, ends at Y, and contains (U, W). Any P^+-bridge is either a P_0-bridge (and receives a representative there) or is G_i for some $1 \leq i \leq s$. For $i > 1$, assign x_{i-1} as representative to G_i. Graph G_1 has an empty interior by choice of x_1 and needs no representative.

There are two reasons why we cannot always use P^+ for the result. First, it may use virtual edges and hence not be a path in G. Second, some P^+-bridge G_i may have four attachment points. Both are resolved by expanding P^+ via paths through the G_i's. Fix one i with $1 \leq i \leq s$ and consider the following cases:

Case 4b-1: (x_{i-1}, x_i) is virtual and used by P^+, and $t_{i-1} = t_i$. By Claim 4, the interior of graph G_i is non-empty and inside the separating triplet $\{x_{i-1}, x_i, t_i\}$. Replace (x_{i-1}, x_i) by a path through G_i with the substitution trick; see graph G_3 in Figure 6.

Case 4b-2: (x_{i-1}, x_i) is virtual and used by P^+, and $t_{i-1} \neq t_i$. See Figure 6b(top). We want to replace both (x_{i-1}, x_i) and (t_{i-1}, t_i) (which is always used by P^+) with a path through G_i. Let $G_i^{\asymp}$ be the graph G_i with (t_i, x_i) and (t_{i-1}, x_{i-1}) added. The outer face of $G_i^{\asymp}$ is a simple cycle since f_i contains no vertex of the right side by Claim 4, so $G_i^{\asymp}$ satisfies $c3c(t_i, t_{i-1}, x_{i-1}, x_i)$. By induction, find a T_{int}-path P_i in $G_i^{\asymp}$ from t_i to x_i

that uses the edge (t_{i-1}, x_{i-1}). So removing (t_{i-1}, x_{i-1}) from P_i splits it into two paths: path P_i^{R} connects t_i to t_{i-1}, and path P_i^{L} connects x_{i-1} to x_i. (No other split is possible by planarity.) Neither path uses the added edge (t_i, x_i) since it connects the ends of P_i. Use P_i^{R} to replace (t_{i-1}, t_i) and P_i^{L} to replace (x_{i-1}, x_i) in P^+.

Case 4b-3: Subgraph G_i has a non-empty interior and $t_i \neq t_{i-1}$. See Figure 6b(bottom). In this case, G_i is a P^+-bridge with four attachment points, a violation of Tutte path properties. If Case 4b-2 applied to G_i, then G_i is no longer a bridge of the resulting path and we are done. Otherwise, we do a substitution that uses a different supergraph of G_i. Let $G_i^{\sqsubset}$ be G_i with edges from path $\langle t_{i-1}, x_{i-1}, x_i, t_i\rangle$ added if not already in G_i. This graph satisfies $c3c(t_i, t_{i-1}, x_{i-1}, x_i)$ and satisfies condition ($\sqsupset$) if we set $X' = t_i$, $U' = t_{i-1}, W' = x_{i-1}$, and $Y' = x_i$. So we can find a T_{end}-path P_i' of $G_i^{\sqsubset}$ from t_i to x_i that uses (t_{i-1}, x_{i-1}) and (x_{i-1}, x_i). Thus, P_i' ends with $\langle t_{i-1}, x_{i-1}, x_i\rangle$ and $P_i' \setminus \{(t_{i-1}, x_{i-1}), (x_{i-1}, x_i)\}$ is a path from t_{i-1} to t_i in G_i that does not visit x_{i-1} or x_i. Substitute this path in place of edge (t_{i-1}, t_i) in P^+. Note that one P_i'-bridge C may use x_i as its representative, but if so, then it also has x_{i-1} as attachment point. We set x_{i-1} (which was G_i's representative and is no longer needed as such) to be the representative of C.

Case 4b-4: $t_{i-1} \neq t_i$ and (t_{i-1}, t_i) is virtual. Since P^+ always uses edge (t_{i-1}, t_i), we must replace this edge with a path through G_i. This is done automatically because Case 4b-3 applies. Namely, if (t_{i-1}, t_i) is virtual, then there is at least one vertex between t_{i-1} and t_i on the right side. This vertex is exterior in G and hence neither x_i nor x_{i-1}. So it is strictly inside $\mathcal{N}_i$, hence G_i has a non-empty interior and (by $t_{i-1} \neq t_i$) Case 4b-3 applies.

After doing these substitutions, there are no virtual edges in the path, no bridges have four attachment points, every bridge has an interior vertex as representative, and no vertex was used twice as representative; see Figure 6c. This ends the proof of Lemma 2 and 3.

2.2 Linear time complexity

It should be clear that our proof is algorithmic. The main bottlenecks for its running time are to determine which case to apply (i.e., whether there is a cutting pair) and to find the B-necklace. Both can be done in linear time, by computing all cutting pairs [6, 9] and by finding a leftmost path in the subgraph induced by vertices that are face-adjacent to $S_{WY} \setminus B$. This would yield quadratic running time overall. For triangulated planar graphs, this is easily reduced to linear: cutting pairs correspond to interior edges where both ends are exterior, and the necklace can be found, as in [1], with a left-first search that only advances neighbors of $S_{WY} \setminus B$. But for graphs that are not triangulated we need a few extra data structures. We sketch only some ideas for this here; details are in the full version [5].

Globally, we keep track of the corners X, U, W, and Y. For each *interior vertex* w and every *side* S_{ab}, we keep a list $\mathcal{V}(w, S_{ab})$ of faces that contain w as well as a vertex on S_{ab}. In these lists, we can look up quickly whether an interior vertex is face-adjacent to a side. Also, each face knows for each side which vertices it has on it. Finally, for each *pair of sides* S_{ab} and S_{cd}, we store a list $\mathcal{P}(S_{ab}; S_{cd})$ of faces that are incident to a vertex on S_{ab} and a (different) vertex on S_{cd}, i.e., faces that connect cutting pairs.

This allows to test for Case 2 and Case 3 easily ("is $\mathcal{P}(S_{XU}, S_{WY})$ resp. $\mathcal{P}(S_{WY}, S_{YX})$ non-empty?"), and Case 1 and Case 4 are easily determined from the planar embedding. We keep $\mathcal{P}(S_{WY}, S_{YX})$ in an order such that its first entry is the appropriate cutting pair in Case 3. To find a necklace, we *scan* the faces incident to $x_1, \ldots, x_s$. More precisely, we

consider (for vertex x_i, presuming we know face f_i already) each face f in ccw order after f_i, and along face f each vertex w in ccw order after x_i, until we find vertex B (then we are done) or a vertex that is face-adjacent to a vertex in $S_{WY} \setminus B$ (then this is x_{i+1} and $f_{i+1} = f$ and we repeat). The running time for this is proportional to the degrees of vertices and faces that were scanned. We also need to update the data structures when recursing into a subgraph; here, we scan along all vertices (and their incident faces) that were in some necklace along which we cut the graph, or that became newly exterior.

A few crucial insights are needed to bound the running time. First, by corner-3-connectivity every face has at most two vertices on each side. In particular, the above data structures have linear size. Second, we need to scan vertices and faces only if they become incident to a side that they were not previously incident to. Finally, once a vertex or face is incident to a side, it remains incident to it forever (though the side may change role, e.g. from "left" to "top"). This means that every vertex and face is scanned only a constant number of times, because there are only four sides to have incidences with. In fact, we only scan vertices and faces that are incident to the outer face in some subgraph, which means that they will be incident to the path P that we compute, and we have the following:

▶ **Theorem 5.** *The Tutte path P for Theorem 1, Lemma 2 or Lemma 3 can be found in linear time. More specifically, the running time is $O(\sum_{f \in F(P)} \deg(f))$.*

3 Applications

A number of interesting properties of planar 3-connected graphs can be derived easily from the existence of T_{SDR}-paths. In particular, every planar 3-connected graph has a spanning tree of maximum degree 3 [2] (a concept known in the literature as a *3-tree*, but we prefer to use the term *binary spanning tree* to avoid confusion with maximal graphs of treewidth 3). Secondly, every planar 3-connected graph has a *2-walk*, i.e., a walk that visits every vertex at least once and at most twice [8]. In the full version [5], we show that, using Lemma 2, these can be found in linear time; this was known for binary spanning trees [16, 3], but for 2-walks the previous best running time was $O(n^3)$ [15].

▶ **Theorem 6.** *Let G be a 3-connected plane graph with exterior vertex X. Then G has a binary spanning tree T that can be found in linear time. Moreover, when rooting T at X, a vertex v has two children only if it is an interior vertex and part of a cutting triplet $\{v, w, x\}$ of G; one of the subtrees of v contains exactly the vertices interior to $\{v, w, x\}$.*

▶ **Theorem 7.** *Let G be a 3-connected plane graph with exterior vertex X. Then G has a 2-walk P that can be found in linear time. Moreover, P visits X exactly once, and it visits a vertex v twice only if v is part of a separating triplet.*

4 Outlook

In this paper, we improved on a very recent result that shows that Tutte paths in planar graphs can be found in quadratic time. We gave a different existence proof which leads to a linear-time algorithm. For 3-connected planar graphs, we obtain not only a Tutte path, but furthermore endow it with a system of distinct representatives, none of which is on the outer face. With this, we can also find 2-walks and binary spanning trees in 3-connected planar graphs in linear time.

The main remaining questions concern how to find Tutte path in other situations or with further restriction. For example, Thomassen [18] and later Sanders [12] improved Tutte's result and showed that we need not restrict the ends of the Tutte path to lie on the outer face. These paths can be found in quadratic time [15]. But our proof does not seem to carry over to the result by Sanders, because the ends of the path crucially must coincide with corners of the graph. Can we find such a path in linear time?

Furthermore, the existence of Tutte paths has been studied for other types of surfaces (see, e.g., Kawarabayashi and Ozeki [11] and the references therein). Can these Tutte paths be found in polynomial time, and preferably, linear time?

References

1. Takao Asano, Shunji Kikuchi, and Nobuji Saito. A Linear Algorithm for Finding Hamiltonian Cycles in 4-Connected Maximal Planar Graphs. *Discrete Applied Mathematics*, 7:1–15, 1985. doi:10.1016/0166-218X(84)90109-4.
2. David W. Barnette. Trees in Polyhedral Graphs. *Canadian Journal of Mathematics*, 18:731–736, 1966. doi:10.4153/CJM-1966-073-4.
3. Therese Biedl. Trees and Co-trees with Bounded Degrees in Planar 3-Connected Graphs. In R. Ravi and Inge Li Gørtz, editors, *Scandinavian Symposium and Workshops on Algorithms Theory (SWAT'14)*, volume 8503 of *Lecture Notes in Computer Science*, pages 62–73. Springer-Verlag, 2014. doi:10.1007/978-3-319-08404-6_6.
4. Therese Biedl and Martin Derka. 1-String B_2-VPG-Representations of Planar Graphs. *Journal on Computational Geometry*, 7(2):191–215, 2016. doi:10.20382/jocg.v7i2a8.
5. Therese Biedl and Philipp Kindermann. Finding Tutte Paths in Linear Time. *Arxiv report*, abs/1812.04543, 2018. arXiv:1812.04543.
6. Giuseppe Di Battista and Roberto Tamassia. Incremental Planarity Testing. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS'89)*, pages 436–441. IEEE Computer Society, 1989. doi:10.1109/SFCS.1989.63515.
7. Reinhard Diestel. *Graph Theory, 5th Edition*, volume 173 of *Graduate Texts in Mathematics*. Springer, 2016. URL: http://diestel-graph-theory.com/.
8. Zhicheng Gao and R. Bruce Richter. 2-Walks in Circuit Graphs. *Journal of Combinatorial Theory, Series B*, 62(2):259–267, 1994. doi:10.1006/jctb.1994.1068.
9. Carsten Gutwenger and Petra Mutzel. A Linear Time Implementation of SPQR-Trees. In Joe Marks, editor, *Proceedings of the 8th International Symposium on Graph Drawing (GD'00)*, volume 1984 of *Lecture Notes in Computer Science*, pages 77–90. Springer, 2000. doi:10.1007/3-540-44541-2_8.
10. John E. Hopcroft and Robert E. Tarjan. Dividing a graph into triconnected components. *SIAM Journal on Computing*, 2(3):135–158, 1973. doi:10.1137/0202012.
11. Ken-ichi Kawarabayashi and Kenta Ozeki. 4-Connected Projective-Planar Graphs Are Hamiltonian-Connected. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'13)*, pages 378–395. Society for Industrial and Applied Mathematics, 2013. doi:10.1016/j.jctb.2012.11.004.
12. Daniel P. Sanders. On Paths in Planar Graphs. *Journal of Graph Theory*, 24(4):341–345, 1997. doi:10.1002/(SICI)1097-0118(199704)24:4<341::AID-JGT6>3.0.CO;2-O.
13. Andreas Schmid and Jens M. Schmidt. Computing 2-Walks in Polynomial Time. In Ernst W. Mayr and Nicolas Ollinger, editors, *Proceedings of the 32nd International Symposium on Theoretical Aspects of Computer Science (STACS'15)*, volume 30 of *LIPIcs*, pages 676–688. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.STACS.2015.676.
14. Andreas Schmid and Jens M. Schmidt. Computing 2-Walks in Polynomial Time. *ACM Transactions on Algorithms*, 14(2):22:1–22:18, 2018. doi:10.1145/3183368.

15 Andreas Schmid and Jens M. Schmidt. Computing Tutte Paths. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *Proceedings of the 45th International Colloquium on Algorithms, Languages and Programming (ICALP'18)*, volume 107 of *LIPIcs*, pages 98:1–98:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.98.

16 Willy-Bernhard Strothmann. *Bounded-Degree Spanning Trees.* Ph.d. thesis, Universität Paderborn, Heinz Nixdorf Institut, Theoretische Informatik, Paderborn, 1997. ISBN 3-931466-34-5. URL: https://www.hni.uni-paderborn.de/pub/468.

17 Robin Thomas and Xingxing Yu. 4-Connected Projective-Planar Graphs Are Hamiltonian. *Journal of Combinatorial Theory, Series B*, 62:114–132, 1994. doi:10.1006/jctb.1994.1058.

18 Carsten Thomassen. A Theorem on Paths in Planar Graphs. *Journal of Graph Theory*, 7(2):169–176, 1983. doi:10.1002/jgt.3190070205.

19 William T. Tutte. Bridges and Hamiltonian Circuits in Planar Graphs. *Aequationes Mathematicae*, 15(1):1–33, 1977. doi:10.1007/BF01837870.

Approximate Counting of k-Paths: Deterministic and in Polynomial Space

Andreas Björklund
Lund University, Lund, Sweden
andreas.bjorklund@cs.lth.se

Daniel Lokshtanov
University of California, Bergen, Santa Barbara, USA
daniello@ucsb.edu

Saket Saurabh
The Institute of Mathematical Sciences, HBNI, Chennai, India
saket@imsc.res.in

Meirav Zehavi
Ben-Gurion University, Beersheba, Israel
meiravze@bgu.ac.il

Abstract

A few years ago, Alon et al. [ISMB 2008] gave a *simple randomized* $\mathcal{O}((2e)^k m\epsilon^{-2})$-time exponential-space algorithm to approximately compute the number of paths on k vertices in a graph G up to a multiplicative error of $1 \pm \epsilon$. Shortly afterwards, Alon and Gutner [IWPEC 2009, TALG 2010] gave a *deterministic* exponential-space algorithm with running time $(2e)^{k+\mathcal{O}(\log^3 k)} m \log n$ whenever $\epsilon^{-1} = k^{\mathcal{O}(1)}$. Recently, Brand et al. [STOC 2018] provided a speed-up at the cost of reintroducing randomization. Specifically, they gave a *randomized* $\mathcal{O}(4^k m\epsilon^{-2})$-time exponential-space algorithm. In this article, we revisit the algorithm by Alon and Gutner. We modify the foundation of their work, and with a novel twist, obtain the following results.

- We present a *deterministic* $4^{k+\mathcal{O}(\sqrt{k}(\log^2 k+\log^2 \epsilon^{-1}))} m \log n$-time *polynomial-space* algorithm. This *matches* the running time of the best known deterministic polynomial-space algorithm for *deciding* whether a given graph G has a path on k vertices.
- Additionally, we present a *randomized* $4^{k+\mathcal{O}(\log k(\log k+\log \epsilon^{-1}))} m \log n$-time *polynomial-space* algorithm. While Brand et al. make non-trivial use of exterior algebra, our algorithm is very simple; we only make elementary use of the probabilistic method.

Thus, the algorithm by Brand et al. runs in time $4^{k+o(k)}m$ whenever $\epsilon^{-1} = 2^{o(k)}$, while our deterministic and randomized algorithms run in time $4^{k+o(k)} m \log n$ whenever $\epsilon^{-1} = 2^{o(k^{\frac{1}{4}})}$ and $\epsilon^{-1} = 2^{o(\frac{k}{\log k})}$, respectively. Prior to our work, no $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$-time *polynomial-space* algorithm was known. Additionally, our approach is embeddable in the classic framework of divide-and-color, hence it immediately extends to approximate counting of graphs of bounded *treewidth*; in comparison, Brand et al. note that their approach is limited to graphs of bounded *pathwidth*.

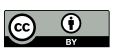
2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases parameterized complexity, approximate counting, k-Path

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.24

Category Track A: Algorithms, Complexity and Games

Funding *Saket Saurabh*: This work is supported by the European Research Council (ERC) via grant LOPPRE, reference 819416.
Meirav Zehavi: This work is supported by the Israel Science Foundation individual research grant no. 1176/18.



46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 24; pp. 24:1–24:15

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

1 Introduction

The objective of the #k-Path problem is to compute the number of k-paths – that is, (simple) paths on k vertices – in a given graph G. Unfortunately, this problem is #W[1]-hard [19], which means that it is unlikely to be solvable in time $f(k)n^{\mathcal{O}(1)}$ for any computable function f of k. Nevertheless, this problem is long known to admit an FPT-approximation scheme (FPT-AS), that is, an $f(k, \epsilon^{-1})n^{\mathcal{O}(1)}$-time algorithm that approximately computes the number of k-paths in a given graph G up to a multiplicative error of $1 \pm \epsilon$. More than 15 years ago, Arvind and Raman [6] utilized the classic method of *color coding* [5] to design a *randomized* exponential-space FPT-AS for #k-Path with running time $k^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ whenever $\epsilon^{-1} \leq k^{\mathcal{O}(k)}$. A few years afterwards, the development and use of applications in computational biology to detect and analyze *network motifs* have already become common practice [34, 37, 36, 18, 24]. Roughly speaking, a network motif is a small pattern whose number of occurrences in a given network is substantially larger than its number of occurrences in a random network. Due to their tight relation to network motifs, #k-Path and other cases of the #Subgraph Isomorphism problem became highly relevant to the study of gene transcription networks, protein-protein interaction (PPI) networks, neural networks and social networks [31]. In light of these developments, Alon et al. [2] revisited the method of color coding to attain a running time whose dependency on k is *single-exponential* rather than *slightly super-exponential*. Specifically, they designed a *simple randomized* $\mathcal{O}((2e)^k m\epsilon^{-2})$-time exponential-space FPT-AS for #k-Path, which they employed to analyze PPI networks of unicellular organisms. In particular, their algorithm has running time $2^{\mathcal{O}(k)}m$ whenever $\epsilon^{-1} \leq 2^{\mathcal{O}(k)}$.

The first *deterministic* FPT-AS for #k-Path was found in 2007 by Alon and Gutner [4]; this algorithm has an exponential space complexity and running time $2^{\mathcal{O}(k \log\log k)}m \log n$ whenever $\epsilon^{-1} = 2^{o(\log k)}$. Shortly afterwards, Alon and Gutner [3] improved upon their previous work, and designed a deterministic exponential-space FPT-AS for #k-Path with running time $(2e)^{k+\mathcal{O}(\log^3 k)}m \log n$ whenever $\epsilon^{-1} = k^{\mathcal{O}(1)}$. For close to a decade, this algorithm has remained the state-of-the-art. In contrast, during this decade, the k-Path problem (the decision version of #k-Path) has seen several improvements that were considered to be breakthroughs at their time [14, 26, 8, 10, 21]. In 2016, Koutis and Williams [27] conjectured that #k-Path admits an FPT-AS with running time $2^k n^{\mathcal{O}(1)}$. Recently, at the cost of reintroducing randomization, Brand et al. [13] provided a speed-up towards the resolution of this conjecture. Specifically, they gave an algebraic *randomized* $\mathcal{O}(4^k m\epsilon^{-2})$-time exponential-space algorithm. In the context of Parameterized Complexity in general, and the k-Path problem in particular, the power of randomization is an issue of wide interest [1]. Specifically for the k-Path problem, an algebraic randomized $2^k n^{\mathcal{O}(1)}$-time algorithm has been found already a decade ago [38], and since then, the existence of a deterministic algorithm that exhibits the same time complexity has been repeatedly posed as a major open problem in the field. Both Koutis and Williams conjectured this question to have an affirmative answer in several venues [38, 28, 27]. Clearly, this question is simpler than the one of the design of a deterministic FPT-AS for #k-Path with running time $2^k n^{\mathcal{O}(1)}$.

In this article, we modify the foundation of the work of Alon and Gutner [4, 3], and with a novel twist, obtain the following results (see Theorem 21 and Corollary 10).

- First, we present a *randomized* $4^{k+\mathcal{O}(\log k(\log k+\log \epsilon^{-1}))}m \log n$-time *polynomial-space* algorithm. While Brand et al. [13] make non-trivial use of exterior algebra, *our randomized algorithm is very simple*: we only make elementary use of the probabilistic method.[1]

[1] Of course, simplicity is a subjective matter, which may depend on the background of the reader.

- Additionally, we present a *deterministic* $4^{k+\mathcal{O}(\sqrt{k}(\log^2 k+\log^2 \epsilon^{-1}))}m \log n$-time *polynomial-space* algorithm. In particular, *without compromising time complexity, we attain both the properties of having a polynomial space complexity and being deterministic simultaneously.* In fact, even though we deal with #k-Path, the running time of our algorithm *matches* the best known running time of a deterministic polynomial-space algorithm for k-Path (the decision version of #k-Path) [14].

Thus, the algorithm by Brand et al. [13] runs in time $4^{k+o(k)}m$ whenever $\epsilon^{-1} = 2^{o(k)}$, while our deterministic and randomized algorithms run in time $4^{k+o(k)}m \log n$ whenever $\epsilon^{-1} = 2^{o(k^{\frac{1}{4}})}$ and $\epsilon^{-1} = 2^{o(\frac{k}{\log k})}$, respectively.

Prior to our work, no $c^k n^{\mathcal{O}(1)}$-time *polynomial-space* (even randomized) algorithm for #k-Path was known *for any constant* c. The design of polynomial-space parameterized algorithms is an active research area in Parameterized Complexity. Even (sometimes) at a notable compromise of time complexity, the property of having polynomial space complexity is sought (see, e.g., [20, 30, 29, 7, 23]). Indeed, algorithms with high space complexity are in practice more constrained because the amount of memory is not easily scaled beyond hardware constraints whereas time complexity can be alleviated by allowing for more time for the algorithm to finish. Furthermore, algorithms with low space complexity are typically easier to parallelize and more cache-friendly.

Additionally, our approach is embeddable in the classic framework of divide-and-color, hence it immediately extends to approximate counting of graphs of bounded treewidth;in comparison, Brand et al. [13] note that their approach is limited to graphs of bounded pathwidth. Similarly, we can approximately count various other objects such as q-dimensional p-matchings, q-set p-packings, graph motifs, and more:

▶ **Theorem 1.** *The following problems admit deterministic* $4^{k+\mathcal{O}(\sqrt{k}(\log^2 k+\log^2 \frac{1}{\epsilon}))}n^{\mathcal{O}(1)}$*-time (resp. randomized* $4^{k+\mathcal{O}(\log^2 k)}(\frac{1}{\epsilon})^{\mathcal{O}(\log k)}n^{\mathcal{O}(1)}$*-time) FPT-ASs with polynomial space complexity:* (i) #Subgraph Isomorphism *for* k*-vertex subgraphs of treewidth* $\mathcal{O}(1)$*;* (ii) #q-Dimensional p-Matching *with* $k = (q-1)p$*;* (iii) #q-Set p-Packing *with* $k = qp$*;* (iv) #Graph Motif *and* #Module Motif *with* $k = 2p$ *where* p *is the motif size;* (v) #p-Internal Out-Branching *with* $k = 2p$*;* (vi) #Partial Cover *for* k*-element solutions.*[2]

Towards the design of our algorithms, our first conceptual contribution is the introduction of the notion of an *approximate parsimonious splitter*. While a randomized construction of such an object is simple, we do not know how (or whether it is even possible) to compute it deterministically within the size and time bounds that we require. We believe that this gap in knowledge of derandomization is the main reason why, for close to a decade, no progress has been made upon the result by Alon and Gutner [4, 3]. Here, our second conceptual contribution comes into play. We show that for recursive procedures, a weaker object that can only split so called nice sets suffices, since the recursion itself can keep track on the "niceness" of sets. We believe that both the concept of approximate parsimonious splitters as well as our approach of how to weaken a randomized object (to efficiently compute it deterministically) at the cost of simple bookkeeping might find further applications in the future. Our ideas and methods are discussed in more detail in Section 3.

[2] For problems (i) and (iv), the basis 4 is replaced by the basis 4.001 (or, more precisely, $4+\delta$ for any fixed constant $\delta > 0$).

Table 1 State-of-the-art of #k-Path and k-Path.

Ref.	Time	Counting	Deterministic	Poly. Space	Extension
[14]	$4^{k+o(k)}n^{\mathcal{O}(1)}$	No	Yes	Yes	Treewidth $\mathcal{O}(1)$
[40]	$2.597^k n^{\mathcal{O}(1)}$	No	Yes	No	Treewidth $\mathcal{O}(1)$
[38]	$2^k n^{\mathcal{O}(1)}$	No	No	Yes	Treewidth $\mathcal{O}(1)$
[10]	$1.657^k n^{\mathcal{O}(1)}$	No	No	Yes	No Extension
[3]	$(2e)^{k+o(k)}n^{\mathcal{O}(1)}$	Yes	Yes	No	Treewidth $\mathcal{O}(1)$
[13]	$4^k n^{\mathcal{O}(1)}$	Yes	No	No	Pathwidth $\mathcal{O}(1)$
This Paper	$4^{k+o(k)}n^{\mathcal{O}(1)}$	Yes	Yes	Yes	Treewidth $\mathcal{O}(1)$

Related Work. The algorithms by Alon et al. [2] and Alon and Gutner [4, 3], just like our algorithms, extend to approximate counting of graphs of bounded treewidth. (This remark is also made by Alon and Gutner [4, 3].) In what follows, we briefly review works related to exact counting and decision from the viewpoint of Parameterized Complexity. Since these topics are not the focus of our work, the survey is illustrative rather than comprehensive.

The problem of counting the number of subgraphs of a graph G that are isomorphic to a graph H – that is, #Subgraph Isomorphism with Pattern H – admits a dichotomy: If the vertex cover number of H is bounded, then it is FPT [39], and otherwise it is #W[1]-hard [16]. The #W[1]-hardness of #k-Path, originally shown by Flum and Grohe [19], follows from this dichotomy. By using the "meet in the middle" approach, the #k-Path problem and, more generally, #Subgraph Isomorphism with Pattern H where H has bounded *pathwidth* and k vertices, was shown to admit an $n^{\frac{k}{2}+\mathcal{O}(1)}$-time algorithm [9]. Later, Björklund et al. [12] showed that $\frac{k}{2}$ is not a barrier (which was considered to be the case at that time) by designing an $n^{0.455k+\mathcal{O}(1)}$-time algorithm. Recently, a breakthrough that resulted in substantially faster running times took place: Curticapean et al. [15] showed that #Subgraph Isomorphism with Pattern H is solvable in time $\ell^{\mathcal{O}(\ell)}n^{0.174\ell}$ where ℓ is the number of edges in H; in particular, this algorithm solves #k-Path in time $k^{\mathcal{O}(k)}n^{0.174k}$.

The k-Path problem (on both directed and undirected graphs) is among the most extensively studied parameterized problems [17, 22]. After a long sequence of works in the past three decades, the current best known parameterized algorithms for k-Path have running times $1.657^k n^{\mathcal{O}(1)}$ (randomized, polynomial space, undirected only) [10, 8] (extended in [11]), $2^k n^{\mathcal{O}(1)}$ (randomized, polynomial space) [38], $2.597^k n^{\mathcal{O}(1)}$ (deterministic, exponential space) [40, 21, 35], and $4^{k+o(k)}n^{\mathcal{O}(1)}$ (deterministic, polynomial space) [14]. The $1.657^k n^{\mathcal{O}(1)}$-time algorithm of Björklund et al. [10, 8] crucially relies on the symmetric structure of undirected k-paths. However, all other algorithms above directly extend to the detection of subgraphs of bounded treewidth. In particular, if the running time of the algorithm is $c^k n^{\mathcal{O}(1)}$, then the running time of the extension is $c^k n^{t+\mathcal{O}(1)}$ where t is the treewidth of the sought graph. To ensure that the constant c remains the same when dealing with the two deterministic algorithms (of [40, 21, 35] and [14]), the "division into small trees" trick by Fomin et al. [21] can be used; for the randomized algorithm (of [38]), no trick is required.

2 Preliminaries

For the sake of readability, we ignore ceiling and floor signs. Given a graph G, we let $V(G)$ and $E(G)$ denote the vertex set and edge set of G, respectively. For a positive integer k, a *k-path* in G is a (simple) path on k vertices in G; in case G is directed, the path is directed as well. We let $n = |V(G)|$ and $m = |E(G)|$. For a subset $U \subseteq V(G)$, $G[U]$ denotes the subgraph of U induced by G, and $G - U = G[V(G) \setminus U]$.

For a function $f : A \to B$ and subsets $A' \subseteq A$ and $B' \subseteq B$, define $f(A') = \{f(a) : a \in A'\}$ and $f^{-1}(B') = \{a \in A : f(a) \in B'\}$. For two functions $f : A \to B$ and $g : B \to C$, the notation $g \circ f : A \to C$ refers to function composition. For two tuples $X = (x_1, x_2, \ldots, x_p)$ and $Y = (y_1, y_2, \ldots, y_q)$, denote their concatenation by $X \diamond Y = (x_1, x_2, \ldots, x_p, y_1, y_2, \ldots, y_q)$. By standard Chernoff bounds, we have the following bounds.

▶ **Proposition 2** ([32]). *Let $X_1, \ldots, X_n$ be independent random variables, each assigned a value in $\{0, 1\}$. Let $X = \sum_{i=1}^{n} X_i$, and let $\mu = E[X]$ denote the expected value of X. Then, for any $0 \leq \delta \leq 1$, it holds that* (i) $\Pr(X \leq (1-\delta)\mu) \leq e^{-\frac{\delta^2\mu}{2}}$ *and* (ii) $\Pr(X \geq (1+\delta)\mu) \leq e^{-\frac{\delta^2\mu}{3}}$.

Universal Families. For any $k \in \mathbb{N}$, a *k-set* is a set of size k. Given a universe U, denote $\binom{U}{k} = \{S \subseteq U : |S| = k\}$. Given a family $\mathcal{F}$ over U and two subsets $A, B \subseteq U$, denote $\mathcal{F}[A, B] = \{F \in \mathcal{F} : A \subseteq F, B \cap F = \emptyset\}$. Next, we present the definition of a universal family.

▶ **Definition 3** (Universal Family [33, 21]). *Let $n, p, q \in \mathbb{N}$. A family $\mathcal{F}$ of sets over a universe U of size n is an* (n, p, q)-universal family *if for each pair of disjoint sets $A \in \binom{U}{p}$ and $B \in \binom{U}{q}$, there is a set $F \in \mathcal{F}$ that contains A and is disjoint from B, that is, $\mathcal{F}[A, B] \neq \emptyset$.*

In the classic setting by Naor et al. [33], $p = q$. However, as shown by Fomin et al. [21], cases where $p \neq q$ are also of interest. Specifically, the following well-known proposition asserts that small representative families can be computed efficiently.

▶ **Proposition 4** ([33, 21]). *Let $n, p, q \in \mathbb{N}$, and $k = p + q$. Let U be a universe of size n. Then, an (n, p, q)-universal family $\mathcal{F}$ of sets over U of size $\mathcal{O}(\binom{k}{p} \log n)$ can be computed with success probability $1 - 1/n$ in time $\mathcal{O}(\binom{k}{p} n \log n)$. Additionally, an (n, p, q)-universal family $\mathcal{F}$ of sets over U of size $\binom{k}{p} 2^{o(k)} \log n$ can be computed (deterministically) in time $\binom{k}{p} 2^{o(k)} n \log n$. Both computations can enumerate the sets in $\mathcal{F}$ with polynomial delay.*

Observe that the constructions above are essentially optimal since any (n, p, q)-universal family must be of size at least $\binom{k}{p}$. We later extend Definition 3 to be approximately parsimonious, and show how to compute approximate parsimonious universal families.

3 Overview of Our Ideas and Methods

In this section, we discuss our main ideas and methods. Additionally, we present a simplified version of one of our applications in detail.

3.1 Approx. Parsimonious Universal Family: Randomized Construction

For any pair of disjoint sets $A \in \binom{U}{p}$ and $B \in \binom{U}{q}$, Definition 3 guarantees that $\mathcal{F}[A, B] \neq \emptyset$. However, the number of sets in $\mathcal{F}[A, B]$ can be arbitrary. In our applications, the number of sets in $\mathcal{F}[A, B]$ will be tightly linked to the number of solutions whose "first half" is in A and whose "second half" is in B; thus, to avoid over-counting some solutions, we need all families $\mathcal{F}[\cdot, \cdot]$ to be *roughly* of the same size. For this purpose, let us first extend Definition 3 to be approximately parsimonious.

▶ **Definition 5** (δ-Parsimonious Universal Family). *Let $n, p, q \in \mathbb{N}$ and $0 < \delta < 1$. Denote $k = p + q$. A family $\mathcal{F}$ of sets over a universe U of size n is a* δ-parsimonious (n, p, q)-universal family *if there exists $T = T(n, p, q, \delta) > 0$ such that for each pair of disjoint sets $A \in \binom{U}{p}$ and $B \in \binom{U}{q}$, it holds that $(1-\delta) \cdot T \leq |\mathcal{F}[A, B]| \leq (1+\delta) \cdot T$.*

We call the value T above a *correction factor*, and suppose it to be given along with the family $\mathcal{F}$. Our randomized computation of a δ-parsimonious (n,p,q)-universal family is based on the probabilistic method, inspired by [33, 21]. Specifically, we prove the following.

▶ **Theorem 6.** *Let $n,p,q \in \mathbb{N}$ and $0 < \delta < 1$, and denote $k = p+q$. Let U be a universe of size n. A δ-parsimonious (n,p,q)-universal family $\mathcal{F}$ of sets over U of size $t = \mathcal{O}\left(\frac{k^k}{p^p q^q} \cdot k \log n \cdot \frac{1}{\delta^2}\right)$,[3] can be computed with success probability at least $1 - 1/n^{100k}$ in time $\mathcal{O}(t \cdot n)$. In particular, the sets in $\mathcal{F}$ can be enumerated with delay $\mathcal{O}(n)$.*

We note that the choice of 100 is arbitrary; it can be replaced by the choice of any fixed constant c. Crucially, we gain the extra property of being δ-parsimonious while essentially having the same time complexity and upper bound on the size of the output as in the non-parsimonious construction.

3.2 Warm Up Application: Simple Randomized FPT-AS for #k-Path

Before we delve into more technical and less intuitive definitions related to our deterministic construction, we find it important to understand the relation between Definition 5 and #k-Path. For this purpose, we present a simple randomized polynomial-space FPT-AS for #k-Path. The dependency of the time complexity on n is made almost linear in Section 3.3). While the improved algorithm is still short and simple, it is somewhat less intuitive and hence presented separately later. For the sake of illustration, suppose that G is undirected.

Algorithm. Let $\widehat{\epsilon} = \ln(1+\epsilon)$ and $\epsilon' = \widehat{\epsilon}/(k-1)$. Our algorithm is a recursive algorithm, denoted by $\mathcal{A}$. Each call to $\mathcal{A}$ is of the form $\mathcal{A}(G',k')$ where G' is an induced subgraph of G and $k' \in \{1,\ldots,k\}$. For all $u,v \in V(G')$, the call $\mathcal{A}(G',k')$ should output an integer $a_{u,v}$ that approximates the number of k'-paths with endpoints u and v in G'. The initial call to the algorithm is with $G' = G$ and $k' = k$, and the final output is $(\sum_{u,v \in V(G)} a_{u,v})/2$.

We turn to describe a call $\mathcal{A}(G',k')$. In the basis, where $k' = 1$, we return $a_{v,v} = 1$ for all $v \in V(G')$, and $a_{u,v} = 0$ for all $u,v \in V(G')$ (with $u \neq v$).

Now, suppose that $k' \geq 2$. By Theorem 6, for an ϵ'-parsimonious $(n, k'/2, k'/2)$-universal family $\mathcal{F}$ of sets over $V(G)$, we can enumerate the sets $F \in \mathcal{F}$ with delay $\mathcal{O}(n)$. For each set $F \in \mathcal{F}$, we proceed as follows. We first perform two recursive calls: *(i)* we call $\mathcal{A}$ with $(G'[F], k'/2)$; *(ii)* we call $\mathcal{A}$ with $(G' - F, k'/2)$. For any $u,v \in F \cap V(G')$, let $b^F_{u,v}$ denote the number returned by the first call. Similarly, for any $u,v \in V(G') \setminus F$, let $c^F_{u,v}$ denote the number returned by the second call. Then, for all $u \in F$ and $v \in V(G') \setminus F$, define $a^F_{u,v} = \sum_{\substack{\{p,q\} \in E(G') \\ \text{s.t. } p \in F, q \notin F}} b^F_{u,p} \cdot c^F_{q,v}$.

Let T be the correction factor of $\mathcal{F}$. After all sets $F \in \mathcal{F}$ were enumerated, for all $u,v \in V(G')$, we output $a_{u,v}$ calculated as follows: $a_{u,v} = \frac{1}{T} \cdot \sum_{\substack{F \in \mathcal{F} \\ \text{s.t. } u \in F, v \notin F}} a^F_{u,v}$. Note that we do not store all the values $a^F_{u,v}$ simultaneously, but we merely store one such value at a time and delete it immediately after $a^F_{u,v}/T$ is added. This completes the description of $\mathcal{A}$.

[3] Note that as $p+q=k$, the value $\frac{k^k}{p^p q^q}$ is upper bounded by 2^k rather than being of the magnitude of k^k.

Analysis. The main part of the analysis is done in the proof of the following lemma.

▶ **Lemma 7.** *For some fixed constant $\eta > 0$, any call $\mathcal{A}(G', k')$ has polynomial space complexity and running time $\eta^{\log k'} 4^{k'} k'^{\log k'} (\log n)^{\log k'} mn^2 (\frac{1}{\epsilon'^2})^{\log k'}$. Additionally, if all constructions of approximate universal families were successful, then for all $u, v \in V(G')$, the number $a_{u,v}$ returned by $\mathcal{A}(G', k')$ satisfies $(1-\epsilon')^{k'-1} x_{u,v} \leq a_{u,v} \leq (1+\epsilon')^{k'-1} x_{u,v}$ where $x_{u,v}$ is the number of k'-paths with endpoints u and v in G'.*

Proof. Let $k' = k/2^d$. We choose $\eta = 10 \max\{\lambda, \tau\}$, where λ and τ are fixed constants defined later. The proof is by backwards induction of d. In the basis ($k' = 1$), the claim is trivial. Now, let $d \leq \log_2 k - 1$, and suppose that the claim holds for $d+1$. Clearly, $\mathcal{A}(G', k')$ has a polynomial space complexity. By Theorem 6, for a fixed constant $\lambda > 0$ (that is independent of η),

$$|\mathcal{F}| \leq \lambda \cdot \frac{k'^{k'}}{(k'/2)^{k'/2}(k'/2)^{k'/2}} \cdot k' \log n \cdot \frac{1}{\epsilon'^2} = \lambda \cdot 2^{k'} \cdot k' \log n \cdot \frac{1}{\epsilon'^2}.$$

Moreover, by the inductive hypothesis, for a fixed constant $\tau > 0$, the running time of $\mathcal{A}(G', k')$ is upper bounded by $|\mathcal{F}| \cdot \left(2 \cdot \eta^{\log \frac{k'}{2}} 4^{\frac{k'}{2}} (\frac{k'}{2})^{\log \frac{k'}{2}} (\log n)^{\log \frac{k'}{2}} mn^2 (\frac{1}{\epsilon'^2})^{\log \frac{k'}{2}} + \tau mn^2\right)$. Note that τ is independent of η. By choosing $\eta = 10 \max\{\lambda, \tau\}$, this means that the running time of $\mathcal{A}(G', k')$ is upper bounded by

$$\begin{aligned}
& |\mathcal{F}| \cdot \left(2 \cdot \eta^{\log \frac{k'}{2}} 4^{\frac{k'}{2}} (\frac{k'}{2})^{\log \frac{k'}{2}} (\log n)^{\log \frac{k'}{2}} mn^2 (\frac{1}{\epsilon'^2})^{\log \frac{k'}{2}} + \tau mn^2\right) \\
\leq\ & \frac{\eta}{10} 2^{k'} k' \log n \frac{1}{\epsilon'^2} \cdot \left(2 \cdot \eta^{\log k' - 1} 2^{k'} k'^{\log k' - 1} (\log n)^{\log k' - 1} mn^2 (\frac{1}{\epsilon'^2})^{\log k' - 1} + \frac{\eta}{10} mn^2\right) \\
\leq\ & \eta^{\log k'} 4^{k'} k'^{\log k'} (\log n)^{\log k'} mn^2 (\frac{1}{\epsilon'^2})^{\log k'}.
\end{aligned}$$

This completes the proof of the first item of the claim.

Towards the proof of the second item of the claim, suppose that all constructions of approximate universal families were successful, and consider some $u, v \in V(G')$. Let $x^{\widehat{G}}_{p,q}$ denote the number of $k'/2$-paths with endpoints p and q in $\widehat{G}$. By the inductive hypothesis, we have that

$$\begin{aligned}
a_{u,v} &= \frac{1}{T} \cdot \sum_{\substack{F \in \mathcal{F} \\ \text{s.t. } u \in F, v \notin F}} a^F_{u,v} = \frac{1}{T} \cdot \sum_{\substack{F \in \mathcal{F} \\ \text{s.t. } u \in F, v \notin F}} \left(\sum_{\substack{\{p,q\} \in E(G') \\ \text{s.t. } p \in F, q \notin F}} b^F_{u,p} \cdot c^F_{q,v} \right) \\
&\leq \frac{1}{T} \cdot \sum_{\substack{F \in \mathcal{F} \\ \text{s.t. } u \in F, v \notin F}} \left(\sum_{\substack{\{p,q\} \in E(G') \\ \text{s.t. } p \in F, q \notin F}} (1+\epsilon')^{\frac{k'}{2}-1} x^{G'[F]}_{u,p} \cdot (1+\epsilon')^{\frac{k'}{2}-1} x^{G'-F}_{q,v} \right) \\
&= \frac{1}{T} \cdot (1+\epsilon')^{k'-2} \cdot \sum_{\substack{F \in \mathcal{F} \\ \text{s.t. } u \in F, v \notin F}} \left(\sum_{\substack{\{p,q\} \in E(G') \\ \text{s.t. } p \in F, q \notin F}} x^{G'[F]}_{u,p} \cdot x^{G'-F}_{q,v} \right)
\end{aligned}$$

Let $\mathcal{P}_{u,v}$ denote the set of k'-paths in G' with endpoints u and v. In addition, for any subset $F \subseteq V(G')$, let $\mathcal{P}_{u,v}[F]$ denote the set of paths $P \in \mathcal{P}_{u,v}$ where the $k'/2$ vertices on P closest to u (including u) belong to F and the other $k'/2$ vertices on P do not belong to F. Thus,

$$a_{u,v} \leq (1+\epsilon')^{k'-2} \cdot \frac{\sum_{F \in \mathcal{F}} |\mathcal{P}_{u,v}[F]|}{T}.$$

Since $\mathcal{F}$ is an ϵ'-parsimonious $(n, k'/2, k'/2)$-universal family, for any path $P \in \mathcal{P}_{u,v}$ it holds that the number of sets $F \in \mathcal{F}$ such that $P \in \mathcal{P}_{u,v}[F]$ is upper bounded by $(1+\epsilon')T$. Thus,

$$a_{u,v} \leq (1+\epsilon')^{k'-2} \cdot \frac{(1+\epsilon')T|\mathcal{P}_{u,v}|}{T} = (1+\epsilon')^{k'-1} x_{u,v}.$$

Symmetrically, we derive that $(1-\epsilon')^{k'-1} x_{u,v} \leq a_{u,v}$. This completes the proof. ◀

We now conclude the following theorem.

▶ **Theorem 8.** *There is a randomized $(4^{k+o(k)}mn^2 + mn^{2+o(1)})(\frac{1}{\epsilon})^{\mathcal{O}(\log k)}$-time polynomial-space algorithm that, given a graph G, a positive integer k and an accuracy value $0 < \epsilon < 1$, outputs a number y that (with high probability, say, at least $9/10$) satisfies $(1-\epsilon)x \leq y \leq (1+\epsilon)x$ where x is the number of k-paths in G. In particular, if $\frac{1}{\epsilon} = 2^{o(k/\log k)}$, then the running time is $4^{k+o(k)}mn^2 + mn^{2+o(1)}$.*

Proof. By Lemma 7 with $G' = G$ and $k' = k$, we know that the total running time of $\mathcal{A}(G,k)$ is bounded by $4^{k+\mathcal{O}(\log^2 k)}(\log n)^{\log k} mn^2 (\frac{1}{\epsilon'})^{\log k}$ and uses polynomial space. Additionally, if all constructions of approximate universal families were successful, then for all $u, v \in V(G)$, the number $a_{u,v}$ computed by $\mathcal{A}(G,k)$ satisfies $(1-\epsilon')^{k-1} x_{u,v} \leq a_{u,v} \leq (1+\epsilon')^{k-1} x_{u,v}$ where $x_{u,v}$ is the number of k-paths with endpoints u and v in G.

If $\log n \leq 2^{\sqrt{k}}$, then $(\log n)^{\log k} \leq 2^{o(k)}$. Otherwise, when $\log n > 2^{\sqrt{k}}$, it holds that $k < \log^2 \log n$. It follows that $4^{k+\mathcal{O}(\log^2 k)}(\log n)^{\log k} \leq 4^{\log^2 \log n + \mathcal{O}(\log\log\log n)}(\log n)^{2\log\log\log n} \leq n^{\mathcal{O}(\frac{\log^2 \log n}{\log n})} \leq n^{o(1)}$. In addition, by Taylor series $\ln(1+x) = \sum_{n=1}^{\infty}(-1)^{n+1}\frac{x^n}{n}$, it follows that $\epsilon/2 \leq \epsilon - \epsilon^2/2 \leq \ln(1+\epsilon) = \widehat{\epsilon} \leq \epsilon$, which means that $(\frac{1}{\epsilon'})^{\log k} = 2^{\mathcal{O}(\log^2 k)}(\frac{1}{\epsilon})^{\mathcal{O}(\log k)}$. Thus, $4^{k+\mathcal{O}(\log^2 k)}(\log n)^{\log k} mn^2 (\frac{1}{\epsilon'})^{\log k} = (4^{k+o(k)}mn^2 + mn^{2+o(1)})(\frac{1}{\epsilon})^{\mathcal{O}(\log k)}$.

We now claim that with high probability, all constructions of approximate universal families were successful. By Theorem 6, the probability that a single construction is successful is at least $1 - 1/n^{100k}$. Thus, the probability that all constructions are successful is at least $(1-1/n^{100k})^{\mu}$ where μ is the number of constructions. Clearly, the number of constructions is upper bounded by the running time of $\mathcal{A}$. In turn, we can assume w.l.o.g. that the upper bound proven on this running time is, in itself, upper bounded by n^k, since otherwise the problem can be solved exactly by brute force within it. Thus, $\mu \leq n^k$. From this, we know that the probability that all constructions are successful is at least $(1-1/n^{100k})^{n^k}$. As n grows larger, this value approaches 1. In particular, the success probability can be assumed to be at least $9/10$ (otherwise n is a fixed constant), which proves our claim.

Thus, we know that for all $u, v \in V(G)$, it holds that $(1-\epsilon')^{k-1} x_{u,v} \leq a_{u,v} \leq (1+\epsilon')^{k-1} x_{u,v}$. Substituting ϵ' by $\widehat{\epsilon}$, we have that for all $u, v \in V(G)$, it holds that $(1-\widehat{\epsilon})x_{uv} \leq (1-\frac{\widehat{\epsilon}}{k-1})^{k-1} x_{u,v} \leq a_{u,v} \leq (1+\frac{\widehat{\epsilon}}{k-1})^{k-1} x_{u,v} \leq e^{\widehat{\epsilon}} x_{u,v}$. Since $(1-\epsilon) \leq (1-\widehat{\epsilon})$ and $e^{\widehat{\epsilon}} = (1+\epsilon)$, we have that for all $u, v \in V(G)$, it holds that $(1-\epsilon)x_{u,v} \leq a_{u,v} \leq (1+\epsilon)x_{u,v}$. Thus,

$$\begin{aligned} y &= \left(\sum_{u,v \in V(G)} a_{u,v}\right)/2 \leq \left(\sum_{u,v \in V(G)} (1+\epsilon)x_{u,v}\right)/2 \\ &= (1+\epsilon)\left(\sum_{u,v \in V(G)} x_{u,v}\right)/2 = (1+\epsilon)x. \end{aligned}$$

Symmetrically, we obtain that $(1-\epsilon)x \leq y$. This completes the proof. ◀

3.3 Improved Randomized FPT-AS for #k-Path

As our improved randomized FPT-AS is less intuitive, we first discuss the intuition behind it. Here, in addition to G' and k', every call to the recursive algorithm $\mathcal{A}$ is given an assignment $\alpha' : V(G) \setminus V(G') \to \mathbb{N}_0$ of a non-negative integer to each vertex outside G'. Roughly speaking, for each vertex $v \in V(G) \setminus V(G')$, the value $\alpha'(v)$ is an approximation of the number of $\widehat{k}$-paths that end at v and are completely contained in $G - U$ for a certain integer $\widehat{k} \in \{1, 2, \ldots, k - k'\}$ and a subset $U \subseteq V(G)$ that contains $V(G')$. In particular, given that now the goal of each call is to output such an assignment for $G - (U \setminus V(G')$ (a precise definition of the goal of a call is given in the formal description of the algorithm), we do not need to consider every pair of vertices $u, v \in V(G')$ and compute a value $a_{u,v}$; instead, we only compute one value per vertex. Additionally, recall that in the previous algorithm in order to compute $a_{u,v}$, we considered every edge $\{p, q\} \in E(G')$ while computing $a^F_{u,v}$ and hence divided our task into the computation of $k'/2$-paths between u and p in one recursive call and $k'/2$-paths between q and u in the other. Here, we do not store the two endpoints of paths, but their "middle". More precisely, the flow of information differs: to compute the assignment we need to output in the current call, we perform one recursive call to which the assignment α' is given as input; this call will return an assignment that "handles" the first $\widehat{k} + k'/2$ vertices on the paths being counted, and be sent as input to the second recursive call to handle the next $k'/2$ vertices.

Algorithm. Let $\widehat{\epsilon} = \ln(1 + \epsilon)$ and $\epsilon' = \widehat{\epsilon}/(k - 1)$. We add a new vertex s to G and connect it to all vertices in G. Thus, rather than counting the number of k-paths in the former graph G, we can count the number of $(k + 1)$-paths with s as an endpoint in the new graph G. In what follows, we focus on this goal.

Our algorithm is a recursive algorithm, denoted by $\mathcal{A}$. Each call to $\mathcal{A}$ is of the form $\mathcal{A}(G', k', \alpha')$ where G' is an induced subgraph of G, $k' \in \{1, \ldots, k\}$, and $\alpha' : V(G) \setminus V(G') \to \mathbb{N}_0$. The call $\mathcal{A}(G', k', \alpha')$ should output an assignment $\alpha : V(G') \to \mathbb{N}_0$ with the following property: For each vertex $v \in V(G')$, it holds that $\alpha(v)$ approximates the following number:

$$\sum_{\substack{\{p,q\} \in E(G) \\ \text{s.t. } p \notin V(G'), q \in V(G')}} \alpha'(p) \cdot x_{q,v},$$

where $x_{q,v}$ is the number of k'-paths in G' between q and v.

The initial call to the algorithm is with $G' = G - \{s\}$, $k' = k$, and $\alpha'(s) = 1$. The final output is $\sum_{v \in V(G) \setminus \{s\}} \alpha(v)$.

We turn to describe a call $\mathcal{A}(G', k', \alpha')$. In the basis, where $k' = 1$, we return an assignment $\alpha : V(G') \to \mathbb{N}_0$ defined as follows: For each vertex $v \in V(G')$, define

$$\alpha(v) = \sum_{\substack{u \notin V(G') \\ \text{s.t. } \{u,v\} \in E(G)}} \alpha'(u).$$

Now, suppose that $k' \geq 2$. By Theorem 6, for an ϵ'-parsimonious $(n, k'/2, k'/2)$-universal family $\mathcal{F}$ of sets over $V(G)$, we can enumerate the sets $F \in \mathcal{F}$ with delay $\mathcal{O}(n)$. For each set $F \in \mathcal{F}$, we proceed as follows. We first recursively call $\mathcal{A}$ with $(G'[F], k'/2, \alpha')$ where α' is extended to assign 0 to every vertex in $V(G') \setminus F$. Let $\widehat{\alpha}_F$ be the output of this call, and extend it to assign 0 to every vertex in $V(G) \setminus V(G')$. Then, we recursively call $\mathcal{A}$ with $(G' - F, k'/2, \widehat{\alpha}_F)$. Let α_F be the output of this recursive call.

Let T be the correction factor of $\mathcal{F}$. After all sets $F \in \mathcal{F}$ were enumerated, the output $\alpha : V(G') \to \mathbb{N}_0$ is computed as follows. For all $v \in V(G')$, we calculate

$$\alpha(v) = \left(\sum_{\substack{F \in \mathcal{F} \\ \text{s.t. } v \notin F}} \alpha_F(v) \right) / T.$$

Note that we do not store all the assignments α_F simultaneously, but we merely store one such assignment at a time and delete it immediately after $\alpha_F(v)/T$, for every $v \in V(G')$, is added. This completes the description of $\mathcal{A}$.

Correctness. The proof of correctness of our algorithm roughly follows the same lines as the proof of correctness of Theorem 8. Due to space constraints, we omit the details, and conclude this section with the statement of our result.

▶ **Theorem 9.** *There is a randomized $(4^{k+o(k)}m + mn^{o(1)})(\frac{1}{\epsilon})^{\mathcal{O}(\log k)}$-time polynomial-space algorithm that, given a graph G, a positive integer k and an accuracy value $0<\epsilon<1$, outputs a number y that (with high probability) satisfies $(1-\epsilon)x \leq y/2 \leq (1+\epsilon)x$ where x is the number of k-paths in G. In particular, if $\frac{1}{\epsilon} = 2^{o(k/\log k)}$, then the running time is $4^{k+o(k)}mn^{o(1)}$.*

Additionally, we can obtain the following corollary. (This corollary does not follow directly from Theorem 9, but requires a simple preliminary step to shrink the universe; due to space constraints, the details are omitted.)

▶ **Corollary 10.** *There is a randomized $4^{k+\mathcal{O}(\log^2 k)}m \log n(\frac{1}{\epsilon})^{\mathcal{O}(\log k)}$-time polynomial-space algorithm that, given a graph G, a positive integer k and an accuracy value $0<\epsilon<1$,'outputs a number y that (with high probability) satisfies $(1-\epsilon)x \leq y/2 \leq (1+\epsilon)x$ where x is the number of k-paths in G. In particular, if $\frac{1}{\epsilon} = 2^{o(k/\log k)}$, then the running time is $4^{k+o(k)}m \log n$.*

3.4 Approx. Parsimonious Universal Family: Deterministic Construction

We do not know how to deterministically construct small δ-parsimonious universal families. Indeed, the best construction that we are aware of is the one based on bipartite Paley graphs (see Theorem 11.9 in the book by Jukna [25] and the historical notes behind the result). This construction leads to families of size $4^{k+o(k)}$ for $p = q = \frac{k}{2}$, whereas we would like size $2^{k+o(k)}$. Instead, we provide an efficient deterministic computation of a small δ-parsimonious universal family that is suitable for handling so called "nice pairs". The crucial point is that with respect to our applications, this relaxed construction suffices. In this section, we present the definition of this relaxation, its construction and main property. Due to space constraints, the proofs of the two lemmas and the theorem stated in this section are omitted.

To simplify the following definitions, we introduce the following notation. To see the intuition behind this notation in the context of applications, throughout this section h can be thought of as a function that reduces the size of the universe from n to z, f can be thought of as a function that splits the reduced universe into t parts, and $\overline{\mathbf{p}}$ can be thought of as a function that tells us that each part has k/t "useful" elements (e.g., vertices of paths to be counted in a certain recursive call) among which either p_i or $(k/t) - p_i$ were "exhausted".

▶ **Definition 11.** *Let $n, p, q, t, z \in \mathbb{N}$, and $k = p+q$. Let U be a universe of size n. A function $\overline{\boldsymbol{p}} : \{1, 2, \ldots, t\} \to \{0, 1, \ldots, k/t\}$ such that $\sum_{i=1}^{t} p_i = p$, is called* (p, q, t)-compatible. *When $\overline{\boldsymbol{p}}$ is clear from context, for each $i \in \{1, 2, \ldots, t\}$, denote $p_i = \overline{\boldsymbol{p}}(i)$ and $q_i = (k/t) - p_i$.*

A triple $(h, f, \overline{\boldsymbol{p}})$ is called (n, p, q, t, z)-compatible *if $h : U \to \{1, 2, \ldots, z\}$, $f : \{1, 2, \ldots, z\} \to \{1, 2, \ldots, t\}$, and $\overline{\boldsymbol{p}}$ is (p, q, t)-compatible. (The universe U will be clear from context.)*

We begin by defining what is a nice pair.

▶ **Definition 12** (Nice Pair). *Let $n, p, q, t, z \in \mathbb{N}$. Let U be a universe of size n. Let $(h, f, \overline{\boldsymbol{p}})$ be (n, p, q, t, z)-compatible. A pair (A, B) is* nice *(with respect to $(h, f, \overline{\boldsymbol{p}})$) if $A \in \binom{U}{p}$ and $B \in \binom{U}{q}$ are disjoint sets, and the following conditions hold.*

1. *The function h is injective when restricted to $A \cup B$.*
2. *For each $i \in \{1, 2, \ldots, t\}$, it holds that $|\{u \in A : f(h(u)) = i\}| = p_i$ and $|\{u \in B : f(h(u)) = i\}| = (k/t) - p_i$.*

Towards the definition of a δ-parsimonious universal family for nice pairs, we first present a weaker definition of this notion where we have a triple $(h, f, \overline{\mathbf{p}})$ at hand.

▶ **Definition 13** (Specific δ-Parsimonious Universal Family for Nice Pairs). *Let $n, p, q, t, z \in \mathbb{N}$. Let U be a universe of size n. Let $(h, f, \overline{\boldsymbol{p}})$ be (n, p, q, t, z)-compatible. Let $0 < \delta < 1$. A family $\mathcal{F}$ of sets over $\{1, \ldots, z\}$ is a* δ-parsimonious $(h, f, \overline{\mathbf{p}})$-universal family (for nice pairs) *if there exists $T = T(h, f, \overline{\boldsymbol{p}}, \delta) > 0$ such that for every nice pair (A, B), it holds that $(1 - \delta) \cdot T \leq |\mathcal{F}[h(A), h(B)]| \leq (1 + \delta) \cdot T$.*

Before we show how to extend Definition 13 to the notion useful for applications, we argue that small δ-parsimonious $(h, f, \overline{\mathbf{p}})$-universal families can be computed "efficiently".

▶ **Lemma 14.** *Let $p, q, t, z \in \mathbb{N}$, and denote $k = p + q$ and $s = k/t$. Let $(h, f, \overline{\boldsymbol{p}})$ be (n, p, q, t, z)-compatible. Let $0 < \delta < 1$. A δ-parsimonious $(h, f, \overline{\boldsymbol{p}})$-universal family $\mathcal{F}$ of sets over $\{1, \ldots, z\}$ of size $\ell = \mathcal{O}\left(\binom{k}{p} \cdot (k \cdot \log z \cdot \frac{\mathcal{O}(1)}{\delta})^{2t}\right)$ can be computed in time $\ell \cdot z^{s+1} s^{\mathcal{O}(1)} t$. In particular, the sets in $\mathcal{F}$ can be enumerated with delay $z^{s+1} s^{\mathcal{O}(1)} t$.*

Towards the definition of our general construction, we need to present the definitions of a balanced splitter and a balanced hash family. Constructions of such a splitter and a family were given by Alon and Gutner [4, 3].

▶ **Definition 15** (Definition 2.2 [4]). *Suppose that $1 \leq \ell \leq k \leq n$ and $0 < \epsilon < 1$, and let H be a family of functions from $\{1, \ldots, n\}$ to $\{1, \ldots, \ell\}$. For a set $S \in \binom{\{1,\ldots,n\}}{k}$, let $\mathtt{split}_H(S)$ denote the number of functions $h \in H$ that split H into equal size parts, that is, $|h^{-1}(i) \cap S| = k/\ell$. Then, H is an* ϵ-balanced (n, k, ℓ)-splitter *if there exists $T = T(n, k, \ell, \epsilon) > 0$ such that for every set $S \in \binom{\{1,\ldots,n\}}{k}$, we have $(1 - \epsilon)T \leq \mathtt{split}_H(S) \leq (1 - \epsilon)T$.*

▶ **Definition 16** (Definition 2.1 [4]). *Suppose that $1 \leq k \leq \ell \leq n$ and $0 < \epsilon < 1$. A family H of functions from $\{1, \ldots, n\}$ to $\{1, \ldots, \ell\}$ is an* (ϵ, k)-balanced family of hash functions *if there exists $T = T(n, k, \ell, \epsilon) > 0$ such that for every set $S \in \binom{\{1,\ldots,n\}}{k}$, the number of functions in H that are injective when restricted to S is between $(1 - \epsilon)T$ and $(1 + \epsilon)T$.*

We are now ready to define our general derandomization tool.

▶ **Definition 17** ((General) δ-Parsimonious Universal Family for Nice Pairs). *Let $n, p, q \in \mathbb{N}$ and $0 < \delta < 1$, and denote $k = p + q$, $z = \frac{2k^2}{\epsilon}$, $t = \sqrt{k}$, $s = k/t = \sqrt{k}$, and $\epsilon = \delta/3$. Let U be a universe of size n. A* δ-parsimonious (n, p, q)-universal tuple (for nice pairs) *is a tuple $(H, S, \{\mathcal{F}^{h,f,\overline{\boldsymbol{p}}}\}|_{h \in H, f \in S, \overline{\boldsymbol{p}}})$[4] that satisfies the following conditions.*

- *H is an (ϵ, k)-balanced family of hash functions from $\{1, \ldots, n\}$ to $\{1, \ldots, z\}$ (with correction factor T_H).*
- *S is an ϵ-balanced (z, k, t)-splitter (with correction factor T_S).*
- *For every hash function $h \in H$, splitter $f \in S$ and (p, q, t)-compatible function $\overline{\boldsymbol{p}}$, it holds that $\mathcal{F}^{h,f,\overline{\boldsymbol{p}}}$ is a δ-parsimonious $(h, f, \overline{\boldsymbol{p}})$-universal family (with correction factor $T_{\overline{\boldsymbol{p}}}$).*

[4] The enumeration is over every (p, q, t)-compatible $\overline{\mathbf{p}}$.

By *enumerating the quadruples of* $(H, S, \{\mathcal{F}^{h,f,\overline{p}}\}|_{h\in H, f\in S,\overline{p}})$, we refer to the enumeration of every quadruple $(h, f, \overline{\mathbf{p}}, F)$ such that $h \in H$, $f \in S$ and $F \in \mathcal{F}^{h,f,\overline{\mathbf{p}}}$. We remark that below, for the sake of brevity, when we write $k, z, t, s, \epsilon, T_H, T_S$ and $T_{\overline{\mathbf{p}}}$, we refer to the notations given in Definition 17. Let us now state our construction.

▶ **Theorem 18.** *Let $n, p, q \in \mathbb{N}$ and $0 < \delta < 1$. Denote $k = p + q$. Let U be a universe of size n. A δ-parsimonious (n, p, q)-universal tuple $(H, S, \{\mathcal{F}^{h,f,\overline{p}}\}|_{h\in H, f\in S,\overline{p}})$ with ℓ quadruples can be computed in time $\frac{k^{\mathcal{O}(1)} n \log n}{\delta^{\mathcal{O}(1)}} + \ell \cdot \Delta$. In particular, after preprocessing time $\frac{k^{\mathcal{O}(1)} n \log n}{\delta^{\mathcal{O}(1)}}$, the quadruples of $(H, S, \{\mathcal{F}^{h,f,\overline{p}}\}|_{h\in H, f\in S,\overline{p}})$ can be enumerated with delay Δ. Here,*

$$\begin{aligned} \ell &= \binom{k}{p} \cdot 2^{\mathcal{O}(\sqrt{k}(\log^2 k + \log^2 \frac{1}{\delta}))} \cdot \log n, \text{ and} \\ \Delta &= 2^{\mathcal{O}(\sqrt{k}(\log k + \log \frac{1}{\delta}))}. \end{aligned}$$

In order to state the property of a δ-parsimonious (n, p, q)-universal tuple that makes it useful for applications, we need one last definition.

▶ **Definition 19.** *Let $n, p, q \in \mathbb{N}$ and $0 < \delta < 1$. Let U be a universe of size n. Furthermore, let $(H, S, \{\mathcal{F}^{h,f,\overline{p}}\}|_{h\in H, f\in S,\overline{p}})$ be a δ-parsimonious (n, p, q)-universal tuple. Finally, let $A \in \binom{U}{p}$ and $B \in \binom{U}{q}$ be disjoint sets. We say that the pair (A, B)* fits *a quadruple $(h, f, \overline{\boldsymbol{p}}, F)$ of $(H, S, \{\mathcal{F}^{h,f,\overline{p}}\}|_{h\in H, f\in S,\overline{p}})$ if (A, B) is nice with respect to $(h, f, \overline{\boldsymbol{p}})$, and $h(A) \subseteq F$ and $f \cap h(B) = \emptyset$.*

Finally, we state the promised property.

▶ **Lemma 20.** *Let $n, p, q \in \mathbb{N}$ and $0 < \delta < 1$. Let U be a universe of size n. Furthermore, let $(H, S, \{\mathcal{F}^{h,f,\overline{p}}\}|_{h\in H, f\in S,\overline{p}})$ be a δ-parsimonious (n, p, q)-universal tuple. Then, there exist $T = T(n, p, q, \delta) > 0$ and for every $\overline{\boldsymbol{p}}$ that is (p, q, t)-compatible, $T_{\overline{\boldsymbol{p}}} = T_{\overline{\boldsymbol{p}}}(n, p, q, \delta) > 0$, such that for any $A \in \binom{U}{p}$ and $B \in \binom{U}{q}$ that are disjoint, the following conditions hold.*

1. *The number of triples $(h, f, \overline{\boldsymbol{p}})$ with respect to whom (A, B) is nice, where $h \in H$, $f \in S$ and $\overline{\boldsymbol{p}}$ is (p, q, t)-compatible, is between $(1 - \delta)T$ and $(1 + \delta)T$.*
2. *For any triple $(h, f, \overline{\boldsymbol{p}})$ with respect to whom (A, B) is nice, where $h \in H$, $f \in S$ and $\overline{\boldsymbol{p}}$ is (p, q, t)-compatible, the number of quadruples $(h, f, \overline{\boldsymbol{p}}, F)$ of $(H, S, \{\mathcal{F}^{h,f,\overline{p}}\}|_{h\in H, f\in S,\overline{p}})$ that fit (A, B) is between $(1 - \delta)T_{\overline{\boldsymbol{p}}}$ and $(1 + \delta)T_{\overline{\boldsymbol{p}}}$.*

3.5 Deterministic FPT-AS for #k-Path

Our deterministic FPT-AS builds upon the scheme of our second randomized FPT-AS, but it is more technical. Due to space constraints, the full details of the description of the algorithm and its proof of correctness is omitted. Here, we only discuss the main idea that underlies the design of this algorithm. Like our previous algorithm, this algorithm (denoted by $\mathcal{A}$) is recursive. However, in addition to G', k' and α', every call to $\mathcal{A}$ is also given two tuples $\mathcal{R}$ and $\mathcal{W}$. The number of elements in $\mathcal{R}$ and $\mathcal{W}$ equals the depth d of the current recursive call in the recursion tree.

Roughly speaking, every element in $\mathcal{R}$ is a quadruple $(h_i, f_i, \overline{\mathbf{p}}_i, \sigma_i)$ where *(i)* the triple $(h_i, f_i, \overline{\mathbf{p}}_i)$ corresponds to the interpretation preceding Definition 11, and *(ii)* $\sigma_i \in \{\texttt{left}, \texttt{right}\}$ indicates whether we should count paths that consist of $\overline{\mathbf{p}}_i(j)$ (in case $\sigma_i = \texttt{left}$) or $s_i - \overline{\mathbf{p}}_i(j)$ (in case $\sigma_i = \texttt{right}$) vertices of the j-th part of the reduced universe split by f_i. Thus, we "keep track" of all triples considered along the current recursion branch. The reason why we have to store this information is to ensure that, in the current recursive call, we only count paths P whose vertex set has the following property:

when we will return to the i-th recursive call, the partition (A, B) of $V(P)$ where A consists of the first $\widehat{k}$ vertices of P (for a certain $\widehat{k} \in \{1, 2, \ldots, k\}$ that depends on the location of this i-th call in the recursion tree) is nice with respect to $(h_i, f_i, \overline{\mathbf{p}}_i)$, see Definition 12. This simple (though perhaps slightly tedious) bookkeeping sidesteps the fact that Lemma 20 only suits nice pairs.

The tuple $\mathcal{W}$ is meant to keep track of how many vertices the paths that we currently count have used "so far" from the j-th part of the universe split by f_i for every choice of i and j. For this purpose, $\mathcal{W}$ is defined to have the form $(\overline{\mathbf{w}}_1, \overline{\mathbf{w}}_2, \ldots, \overline{\mathbf{w}}_d)$ such that for each $i \in \{1, 2, \ldots, d\}$, the following condition holds: For each $j \in \{1, 2, \ldots, t_i\}$, if $\sigma_i = \texttt{left}$ then $\overline{\mathbf{w}}_i(j) \leq \overline{\mathbf{p}}_i(j)$, and otherwise $\overline{\mathbf{w}}_i(j) \leq s_i - \overline{\mathbf{p}}_i(j)$. Here, $s_i = \sqrt{(k/2^i)}$ is the number of vertices the paths that we currently count should use (in total) from each part split by f_i.

Accordingly, the objective of a call $\mathcal{A}(G', k', \alpha', \mathcal{R}, \mathcal{W})$ is to output an assignment $\alpha : V(G') \to \mathbb{N}_0$ with the following property: For each vertex $v \in V(G')$, it holds that $\alpha(v)$ approximates $\sum_{\substack{\{p,q\} \in E(G) \\ \text{s.t. } p \notin V(G'), q \in V(G')}} \alpha'(p) \cdot |\mathcal{P}_{q,v}^{G',k',\mathcal{R},\mathcal{W}}|$. Roughly speaking, $\mathcal{P}_{q,v}^{G',k',\mathcal{R},\mathcal{W}}$ is the collection of all k'-paths in G' with endpoints q and v that "comply" with the constraints imposed by $\mathcal{R}$ and $\mathcal{W}$. (Due to space constraints, the formal definition is omitted.)

We conclude this section with the formal statement of our main result.

▶ **Theorem 21.** *There is a deterministic $4^{k+\mathcal{O}(\sqrt{k}(\log^2 k + \log^2 \frac{1}{\epsilon}))} m \log n$-time polynomial-space algorithm that, given a graph G, a positive integer k and an accuracy value $0 < \epsilon < 1$, outputs a number y that satisfies $(1-\epsilon)x \leq y/2 \leq (1+\epsilon)x$ where x is the number of k-paths in G. In particular, if $\frac{1}{\epsilon} = 2^{o(k^{\frac{1}{4}})}$, then the running time is $4^{k+o(k)} m \log n$.*

Due to space constraints, the discussion on extensions and other applications is omitted.

References

1. Randomization in Parameterized Complexity. `www.dagstuhl.de/de/programm/kalender/semhp/?semnr=17041`.
2. Noga Alon, Phuong Dao, Iman Hajirasouliha, Fereydoun Hormozdiari, and Süleyman Cenk Sahinalp. Biomolecular network motif counting and discovery by color coding. In *Proceedings 16th International Conference on Intelligent Systems for Molecular Biology (ISMB), Toronto, Canada, July 19-23, 2008*, pages 241–249, 2008. `doi:10.1093/bioinformatics/btn163`.
3. Noga Alon and Shai Gutner. Balanced Hashing, Color Coding and Approximate Counting. In *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, pages 1–16, 2009. `doi:10.1007/978-3-642-11269-0_1`.
4. Noga Alon and Shai Gutner. Balanced families of perfect hash functions and their applications. *ACM Trans. Algorithms*, 6(3):54:1–54:12, 2010. `doi:10.1145/1798596.1798607`.
5. Noga Alon, Raphael Yuster, and Uri Zwick. Color-Coding. *J. ACM*, 42(4):844–856, 1995. `doi:10.1145/210332.210337`.
6. Vikraman Arvind and Venkatesh Raman. Approximation Algorithms for Some Parameterized Counting Problems. In *Algorithms and Computation, 13th International Symposium, ISAAC 2002 Vancouver, BC, Canada, November 21-23, 2002, Proceedings*, pages 453–464, 2002. `doi:10.1007/3-540-36136-7_40`.
7. André Berger, László Kozma, Matthias Mnich, and Roland Vincze. A time- and space-optimal algorithm for the many-visits TSP. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1770–1782, 2019. `doi:10.1137/1.9781611975482.106`.

8 Andreas Björklund. Determinant Sums for Undirected Hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014. doi:10.1137/110839229.

9 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Counting Paths and Packings in Halves. In *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, pages 578–586, 2009. doi:10.1007/978-3-642-04128-0_52.

10 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *J. Comput. Syst. Sci.*, 87:119–139, 2017. doi:10.1016/j.jcss.2017.03.003.

11 Andreas Björklund, Vikram Kamat, Lukasz Kowalik, and Meirav Zehavi. Spotting Trees with Few Leaves. *SIAM J. Discrete Math.*, 31(2):687–713, 2017. doi:10.1137/15M1048975.

12 Andreas Björklund, Petteri Kaski, and Lukasz Kowalik. Counting Thin Subgraphs via Packings Faster than Meet-in-the-Middle Time. *ACM Trans. Algorithms*, 13(4):48:1–48:26, 2017. doi:10.1145/3125500.

13 Cornelius Brand, Holger Dell, and Thore Husfeldt. Extensor-coding. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 151–164, 2018. doi:10.1145/3188745.3188902.

14 J. Chen, J. Kneis, S. Lu, D. Mölle, S. Richter, P. Rossmanith, S. Sze, and F. Zhang. Randomized Divide-and-Conquer: Improved Path, Matching, and Packing Algorithms. *SIAM Journal on Computing*, 38(6):2526–2547, 2009.

15 Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms Are a Good Basis for Counting Small Subgraphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 210–223, New York, NY, USA, 2017. ACM. doi:10.1145/3055399.3055502.

16 Radu Curticapean and Dániel Marx. Complexity of Counting Subgraphs: Only the Boundedness of the Vertex-Cover Number Counts. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 130–139, 2014. doi:10.1109/FOCS.2014.22.

17 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

18 Banu Dost, Tomer Shlomi, Nitin Gupta, Eytan Ruppin, Vineet Bafna, and Roded Sharan. QNet: A Tool for Querying Protein Interaction Networks. *Journal of Computational Biology*, 15(7):913–925, 2008. doi:10.1089/cmb.2007.0172.

19 Jörg Flum and Martin Grohe. The Parameterized Complexity of Counting Problems. *SIAM J. Comput.*, 33(4):892–922, 2004.

20 Fedor V. Fomin, Petteri Kaski, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Parameterized Single-Exponential Time Polynomial Space Algorithm for Steiner Tree. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 494–505, 2015. doi:10.1007/978-3-662-47672-7_40.

21 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient Computation of Representative Families with Applications in Parameterized and Exact Algorithms. *J. ACM*, 63(4):29:1–29:60, 2016. doi:10.1145/2886094.

22 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2018.

23 Gregory Z. Gutin, Felix Reidl, Magnus Wahlström, and Meirav Zehavi. Designing deterministic polynomial-space algorithms by color-coding multivariate polynomials. *J. Comput. Syst. Sci.*, 95:69–85, 2018. doi:10.1016/j.jcss.2018.01.004.

24 Falk Hüffner, Sebastian Wernicke, and Thomas Zichner. Algorithm Engineering for Color-Coding with Applications to Signaling Pathway Detection. *Algorithmica*, 52(2):114–132, 2008. doi:10.1007/s00453-007-9008-7.

25 Stasys Jukna. *Extremal Combinatorics: With Applications in Computer Science*. Springer Publishing Company, Incorporated, 1st edition, 2010.

26 Ioannis Koutis. Faster Algebraic Algorithms for Path and Packing Problems. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, pages 575–586, 2008. doi:10.1007/978-3-540-70575-8_47.

27 Ioannis Koutis and Ryan Williams. Algebraic fingerprints for faster algorithms. *Commun. ACM*, 59(1):98–105, 2016. doi:10.1145/2742544.

28 Ioannis Koutis and Ryan Williams. LIMITS and applications of group algebras for parameterized problems. *ACM Trans. Algorithms*, 12(3):31:1–31:18, 2016. doi:10.1145/2885499.

29 Daniel Lokshtanov, Matthias Mnich, and Saket Saurabh. Planar k-Path in Subexponential Time and Polynomial Space. In *Graph-Theoretic Concepts in Computer Science - 37th International Workshop, WG 2011, Teplá Monastery, Czech Republic, June 21-24, 2011. Revised Papers*, pages 262–270, 2011. doi:10.1007/978-3-642-25870-1_24.

30 Daniel Lokshtanov and Jesper Nederlof. Saving space by algebraization. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 321–330, 2010. doi:10.1145/1806689.1806735.

31 R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network Motifs: Simple Building Blocks of Complex Networks. *Science*, 298(5594):824–827, 2002. doi:10.1126/science.298.5594.824.

32 Michael Mitzenmacher and Eli Upfal. *Probability and computing - randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

33 Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. Splitters and Near-Optimal Derandomization. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, 23-25 October 1995*, pages 182–191, 1995. doi:10.1109/SFCS.1995.492475.

34 Jacob Scott, Trey Ideker, Richard M. Karp, and Roded Sharan. Efficient Algorithms for Detecting Signaling Pathways in Protein Interaction Networks. *Journal of Computational Biology*, 13(2):133–144, 2006. doi:10.1089/cmb.2006.13.133.

35 Hadas Shachnai and Meirav Zehavi. Representative families: A unified tradeoff-based approach. *J. Comput. Syst. Sci.*, 82(3):488–502, 2016. doi:10.1016/j.jcss.2015.11.008.

36 Roded Sharan and Trey Ideker. Modeling cellular machinery through biological network comparison. Nat. Biotechnol. 24, 427-433. *Nature biotechnology*, 24:427–33, May 2006.

37 Tomer Shlomi, Daniel Segal, Eytan Ruppin, and Roded Sharan. QPath: a method for querying pathways in a protein-protein interaction network. *BMC Bioinformatics*, 7:199, 2006. doi:10.1186/1471-2105-7-199.

38 Ryan Williams. Finding paths of length k in $O^*(2^k)$ time. *Inf. Process. Lett.*, 109(6):315–318, 2009. doi:10.1016/j.ipl.2008.11.004.

39 Virginia Vassilevska Williams and Ryan Williams. Finding, Minimizing, and Counting Weighted Subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013. doi:10.1137/09076619X.

40 Meirav Zehavi. Mixing Color Coding-Related Techniques. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 1037–1049, 2015. doi:10.1007/978-3-662-48350-3_86.

Computing Permanents and Counting Hamiltonian Cycles by Listing Dissimilar Vectors

Andreas Björklund
Department of Computer Science, Lund University, Sweden

Ryan Williams
Department of Electrical Engineering and Computer Science & CSAIL, MIT, Cambridge, MA, USA

Abstract

We show that the permanent of an $n \times n$ matrix over any finite ring of $r \leq n$ elements can be computed with a deterministic $2^{n-\Omega(\frac{n}{r})}$ time algorithm. This improves on a Las Vegas algorithm running in expected $2^{n-\Omega(n/(r \log r))}$ time, implicit in [Björklund, Husfeldt, and Lyckberg, IPL 2017]. For the permanent over the integers of a 0/1-matrix with exactly d ones per row and column, we provide a deterministic $2^{n-\Omega(\frac{n}{d^{3/4}})}$ time algorithm. This improves on a $2^{n-\Omega(\frac{n}{d})}$ time algorithm in [Cygan and Pilipczuk ICALP 2013]. We also show that the number of Hamiltonian cycles in an n-vertex directed graph of average degree δ can be computed by a deterministic $2^{n-\Omega(\frac{n}{\delta})}$ time algorithm. This improves on a Las Vegas algorithm running in expected $2^{n-\Omega(\frac{n}{\mathrm{poly}(\delta)})}$ time in [Björklund, Kaski, and Koutis, ICALP 2017].

A key tool in our approach is a reduction from computing the permanent to listing pairs of *dissimilar* vectors from two sets of vectors, i.e., vectors over a finite set that differ in each coordinate, building on an observation of [Bax and Franklin, Algorithmica 2002]. We propose algorithms that can be used both to derandomise the construction of Bax and Franklin, and efficiently list dissimilar pairs using several algorithmic tools. We also give a simple randomised algorithm resulting in Monte Carlo algorithms within the same time bounds.

Our new fast algorithms for listing dissimilar vector pairs from two sets of vectors are inspired by recent algorithms for detecting and counting orthogonal vectors by [Abboud, Williams, and Yu, SODA 2015] and [Chan and Williams, SODA 2016].

2012 ACM Subject Classification Mathematics of computing → Combinatorial algorithms

Keywords and phrases permanent, Hamiltonian cycle, orthogonal vectors

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.25

Category Track A: Algorithms, Complexity and Games

Funding *Andreas Björklund*: The Swedish Research Council grant VR 2016-03855 "Algebraic Graph Algorithms".
Ryan Williams: The U.S. National Science Foundation grant CCF-1741615 "CAREER: Common Links in Algorithms and Complexity".

1 Introduction

In recent years, the apparent impossibility of finding a pair of orthogonal vectors among two sets of N Boolean vectors in $N^{2-\varepsilon}$ time (the OV problem) has been used to derive conditional hardness results for many problems (e.g., [2] to name just one). However, when the dimensionality of the vectors is at most $d \log(N)$ for a constant d, N^c-time algorithms do exist for some $c < 2$, and these algorithms for OV have been used to derive faster algorithms for other problems; a prominent example is counting satisfying assignments to sparse CNF formulas [11]. In this paper we consider a natural generalisation of the OV problem, design algorithms for it, and apply those algorithms to derive faster deterministic algorithms for two other notoriously hard, well-known problems:

1. Computing matrix permanents over finite rings and regular 0/1 matrices over the integers.
2. Counting Hamiltonian cycles in sparse directed graphs.

The natural generalisation of OV is a problem we call **listing dissimilar vector pairs**: *Given N vectors over a finite set of size r, list all pairs of vectors that differ in every coordinate.* This problem can easily be reduced to listing orthogonal vector pairs (see Proposition 9). We also propose some tailored algorithms for listing dissimilar pairs for ease of understanding. (and in the hopes that our new algorithmic ideas will lead to interesting future work). As in the literature on orthogonal vector detection and counting [1, 11], we show how fast rectangular matrix multiplication can be used to list dissimilar vector pairs; we apply a fast algorithm for *counting* (not listing) orthogonal vectors ([11]) as a black box to derandomise our application algorithms.

A key difference in our work is that our applications to counting problems require us to list *all* dissimilar vector pairs, rather than merely finding one. The task of listing OV pairs also arises in the fastest known online algorithm for Boolean matrix-vector multiplication [15].

1.1 Faster Algorithm for Ring Permanents

The permanent of a square matrix $M = \{m_{i,j}\} \in K^{n\times n}$ over a ring K is defined as

$$\operatorname{per}(M) = \sum_{\sigma\in S_n} \prod_{i=1}^{n} m_{i,\sigma(i)}, \tag{1}$$

where S_n is the symmetric group on n elements. The problem of computing permanents is known to be #P-complete, even for sparse binary (i.e., 0/1) matrices over the integers [22], and hence we only expect exponential-time algorithms for the problem in general.

An inclusion–exclusion formula by Herbert Ryser from 1963 [18] states that

$$\operatorname{per}(M) = \sum_{X\subseteq[n]} (-1)^{n-|X|} \prod_{i=1}^{n} \left(\sum_{j\in X} m_{i,j} \right). \tag{2}$$

By enumerating the subsets X in a Gray code order (i.e., an enumeration that lists each subset exactly once by only adding or removing one element at a time) one can compute each term $\prod_{i=1}^{n} \left(\sum_{j\in X} m_{i,j}\right)$ in only $O(n)$ operations per subset, leading to an algorithm using $O(n2^n)$ additions and multiplications. To date, this is still the fastest method known for computing the permanent over general rings, and it is a major open question whether there is a $O((2-\epsilon)^n)$ time algorithm for some constant $\epsilon > 0$, even for the special case of binary matrices over the integers. For other special cases, like sparse matrices or finite rings, such algorithms *do* exist, and even for the computation of binary matrices over the integers, *somewhat* faster algorithms than Ryser's are known; see the Related Work section.

In this paper, we provide a faster algorithm for permanents over finite rings.

▶ **Theorem 1.** *There is a deterministic algorithm that computes the permanent of a matrix $M \in K^{n\times n}$ over any finite ring K on $r \le n$ elements in $2^{n-\Omega(n/r)}$ time.*

The previous best bound is a $2^{n-\Omega(n/(r\log r))}$ expected time algorithm implicit in [6].[1] Note that our result is much more than a "log-shaving" of the running time in the classical sense, as we are improving the *exponent* of the running time; moreover, our new algorithm is deterministic as opposed to the previous one.

[1] The randomised algorithms in that paper are stated for computations modulo a fixed prime and their first powers, but the algorithms can be adapted for finite rings on r elements.

We also provide a faster algorithm for d-regular 0/1 matrices with exactly d ones in each row and column.

▶ **Theorem 2.** *There is a deterministic algorithm that computes the permanent of a d-regular matrix $M \in \{0,1\}^{n\times n}$ over the integers in $2^{n-\Omega(n/d^{3/4})}$ time.*

This improves on the $2^{n-\Omega(n/d)}$ time bound of an algorithm for average d ones per row by Cygan and Pilipczuk [13].

1.2 Hamiltonian cycles

A Hamiltonian cycle in a directed graph is a simple cycle through all vertices. Counting Hamiltonian cycles is #P-complete, even in planar graphs of degree at most three [21, 16]. We provide a faster algorithm for sparse graphs.

▶ **Theorem 3.** *There is a deterministic algorithm that counts the number of Hamiltonian cycles in an n-vertex directed graph of average degree δ in $2^{n-\Omega(n/\delta)}$ time.*

The previously fastest counting algorithm (sensitive to the average degree) is a Las Vegas algorithm running in expected $2^{n-\Omega(n/\operatorname{poly}(\delta))}$ time outlined in [8]. The polynomial in the exponent is at least $\delta^4 \log \delta$.[2]

1.3 Related Work

The fastest known algorithms for the $n \times n$ matrix permanent over the integers, and counting Hamiltonian cycles in an n-vertex directed graph, are those of Björklund, Kaski, and Williams [9] which run in $2^{n-\Omega\left(\sqrt{n/\log\log n}\right)}$ time.

For permanents over the integers of binary matrices with d ones per row on average, Servedio and Wan [19] showed how to compute the permanent in $2^{n-\Omega(n/\exp(d))}$ time and polynomial space. Using exponential space, Izumi and Wadayama [14] derived a $2^{n-\Omega(n/(d\log d))}$ time algorithm. Cygan and Pilipczuk [13] gave a $2^{n-\Omega(n/d)}$ time algorithm that works over any ring, where d denotes the *average* number of non-zero entries per row. Björklund, Husfeldt, and Lyckberg [6] showed that the integer permanent can be computed modulo $p^{(1-\lambda)n/p}$ in $c_{p,\lambda}^n$ time, for $c_{p,\lambda} < 2$ depending only on the fixed prime p and $\lambda > 0$.

The fastest known algorithm for detecting the Hamiltonian cycles in an undirected graph is the $O(1.657^n)$ time Monte Carlo algorithm of Björklund [4]. For directed graphs, no detection algorithm running in $O((2-\epsilon)^n)$ time for any $\epsilon > 0$ is known, although for bipartite directed graphs, Hamiltonicity can be decided in $O(1.733^n)$ time [8]. Intriguingly, computing the *parity* of the number of Hamiltonian cycles can be done in $O(1.619^n)$ time [5], and they can be counted modulo certain integers with all prime factors at most p, in $2^{n-\Omega(n/p)}$ time [8].

There are also $6^{\operatorname{pw}(G)}\operatorname{poly}(n)$ time and $15^{\operatorname{tw}(G)}\operatorname{poly}(n)$ time algorithms parameterised by the pathwidth and treewidth of the underlying graph [10].

1.4 Our techniques and contributions

For binary 0/1 matrices over the integers, Bax and Franklin [3] gave a $2^{n-\Omega(n^{1/3}/\log n)}$ expected time Las Vegas algorithm for the $n \times n$ matrix permanent. Their algorithm is slower than the state-of-the-art algorithm of [9] (based on entirely different techniques), but still uses very interesting ingredients whose potential has probably not yet been fully utilised.

[2] See Theorem 8 in Appendix B of [7] for details.

Our algorithms in this paper stem from two ideas in Bax and Franklin's paper:

1. Perturb the input in a way that does not affect the answer, but "zeroes out" all but a tiny fraction of terms in an 2^n-sized formula for computing the answer (e.g., for permanents, we will zero-out most terms in Ryser's formula (2)). Hence to evaluate the sum, it is sufficient to list only the non-zero terms, offering an approach to a faster algorithm.
2. Divide the columns of the matrix in two halves of about $n/2$ columns each, construct vectors representing the terms restricted to the halves, and find ways to "combine" pairs of vectors corresponding to non-zero terms in the exponential sum.

By design, the vector pairs corresponding to non-zero terms will be those differing in each coordinate, which we call *dissimilar pairs*. Recently, Björklund, Kaski, and Koutis, translated this idea to the problem of counting Hamiltonian cycles [8]. In this paper, we present two new methods of listing dissimilar pairs that are more efficient than the simple tabulation-based schemes used in [3] and [8]. Our first listing algorithm is deterministic:

▶ **Theorem 4.** *Given two lists $\mathcal{X}$ and $\mathcal{Y}$ of N vectors in $[r]^d$ with $d < \log_3(N)/20$, and an integer $s > N$, the first s dissimilar pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ can be listed in $\tilde{O}(N\sqrt{s})$ time.*[3]

The algorithm applies fast rectangular matrix multiplication. In particular, we use:

▶ **Theorem 5** (Coppersmith, 1984 [12])**.** *Over any finite field $\mathbb{F}$, the number of arithmetic operations needed to multiply an $N \times N^\alpha$ sized matrix with an $N^\alpha \times N$ sized matrix, for $\alpha < 0.17$, is $N^2 \cdot \operatorname{poly}\log(N)$.*

Our second listing algorithm is randomised, and based on hashing. It is arguably much more implementable, as it does not rely on fast matrix multiplication.

▶ **Theorem 6.** *Given two lists $\mathcal{X}$ and $\mathcal{Y}$ of N vectors in $[r]^d$, the set of dissimilar pairs $S \subseteq \mathcal{X} \times \mathcal{Y}$ can be listed in $\tilde{O}(|S| + 2^d \cdot N)$ time with probability of success $1 - o(1)$.*

We also prove a stronger upper bound on the number of non-zero terms needed to analyse in the special case of d-regular 0/1 matrices. The result can be seen as a sparsity parameterisation of Bax and Franklin's algorithm [3].

Outline. We describe our two dissimilarity listing algorithms in Sections 2 and 3. In Section 4 we review the (randomised) reductions from the permanent and Hamiltonian cycle counting to listing dissimilar vector pairs. Finally, in Section 5 we show how the randomised reductions can be derandomised by applying an algorithm for counting OV pairs, and using the method of conditional expectation.

2 Listing dissimilar vectors with fast matrix multiplication

We first describe a deterministic algorithm for listing dissimilar pairs based on fast rectangular matrix multiplication. Given two sets $\mathcal{X}, \mathcal{Y} \subseteq [r]^d$ with $|\mathcal{X}| = |\mathcal{Y}| = N$ and a positive integer s, we want to output a set of the (lexicographically) first s pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ that are *dissimilar*, i.e., for all $i = 1, \ldots, d$, $x_i \neq y_i$.

▶ **Reminder of Theorem 4.** *Given lists $\mathcal{X}$ and $\mathcal{Y}$ of N vectors in $[r]^d$ with $d < \log_3(N)/20$, and $N < s \leq N^2$, the first s dissimilar pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ can be listed in $\tilde{O}(N\sqrt{s})$ time.*

[3] Note that $s \leq N^2$ implies $s \leq N\sqrt{s}$, so the running time is at least s. Also note that for $s = 2^{n-\delta n}$ and $N = 2^{n/2}$ (our applications of interest), $N \cdot s^{1/2} \leq 2^{n-\delta n/2}$.

Proof. The idea of the algorithm is to partition both $\mathcal{X}$ and $\mathcal{Y}$ in $m = s^{1/4}\sqrt{N}$ pieces, each of size at most N/m, as $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \cdots \cup \mathcal{X}_m$ and $\mathcal{Y} = \mathcal{Y}_1 \cup \mathcal{Y}_2 \cup \cdots \cup \mathcal{Y}_m$. First, the algorithm locates all pairs $(i, j) \in [m] \times [m]$ such that $\mathcal{X}_i \times \mathcal{Y}_j$ contains a dissimilar vector pair. This is achieved for all pairs simultaneously through several rectangular matrix products. Then the algorithm "brute-forces" all pairs in those $\mathcal{X}_i \times \mathcal{Y}_j$ containing dissimilar pairs.

Given a pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$, consider the following polynomial defined over $\mathbb{Z}$:

$$p(x, y) = \prod_{i=1}^{d} (x_i - y_i)^2. \tag{3}$$

Note that $p(x, y) > 0$ if (x, y) is a dissimilar vector pair, otherwise $p(x, y) = 0$.

We can write $p(x, y)$ as a sum of 3^d products:

$$p(x, y) = \sum_{z \in \{0,1,2\}^d} (-2)^{\text{ones}(z)} \left(\prod_{j=1}^{d} x_j^{z_j} \right) \left(\prod_{k=1}^{d} y_k^{2-z_k} \right), \tag{4}$$

where $\text{ones}(z)$ counts the number of coordinates i in z such that $z_i = 1$.

Let $c(i, j)$ be the sum of $p(x, y)$ over all pairs $(x, y) \in \mathcal{X}_i \times \mathcal{Y}_j$, and note $c(i, j) > 0$ if and only if some pair in $\mathcal{X}_i \times \mathcal{Y}_j$ is dissimilar. We wish to compute whether $c(i, j) > 0$ for all i, j. To this end, we construct an $m \times 3^d$ integer matrix $M_{\mathcal{X}}$ representing $\mathcal{X}$, with row i representing $\mathcal{X}_i$, and columns representing different terms in (4), labeled by the corresponding z-vector. Formally, the entry at row i and column $z \in \{0, 1, 2\}^d$ is

$$M_{\mathcal{X},i,z} = \sum_{x \in \mathcal{X}_i} (-2)^{\text{ones}(z)} \prod_{j=1}^{d} x_j^{z_j}. \tag{5}$$

Similarly, we construct an $m \times 3^d$ integer matrix $M_{\mathcal{Y}}$ representing $\mathcal{Y}$:

$$M_{\mathcal{Y},i,z} = \sum_{y \in \mathcal{Y}_i} \prod_{k=1}^{d} y_k^{2-z_k}. \tag{6}$$

We consider $P = M_{\mathcal{X}} \cdot M_{\mathcal{Y}}{}^{\mathrm{T}}$, and observe that $P_{i,j} = c(i, j)$ for all $i, j = 1, \ldots, m$. All entries in the result are $\text{poly}(\log N, \log r)$-bit non-negative integers. To see if $c(i, j) > 0$, we compute P modulo the first $\text{poly}(\log N, \log r)$ primes using Theorem 5. If any of the products has $P_{i,j} \neq 0$, we know that $c(i, j) > 0$ and mark (i, j) as containing dissimilar pairs.

Next, we loop over all marked entries (i, j) of the matrix, and test every $(x, y) \in \mathcal{X}_i \times \mathcal{Y}_j$ for dissimilarity by brute force in lexicographical order. As soon as s dissimilar pairs have been listed, the algorithm terminates.

Noting that the dimensions of the matrices obey the condition for Coppersmith's algorithm (Theorem 5), i.e., $3^d < m^{0.17}$, the running time is

$$(3^d N + m^2 + s(N/m)^2)\,\text{poly}(\log N, \log r) = N\sqrt{s}\,\text{poly}(\log N, \log r).$$

Here, the first two summands come from building the matrices and computing the product P, and the last summand arises from the worst case of the brute-force listing part, when every $\mathcal{X}_i \times \mathcal{Y}_j$ with $c(i, j) > 0$ contains only one dissimilar pair. This concludes the proof. ◀

3 Listing dissimilar vectors by hashing

In this section, we give an alternative listing method that avoids fast rectangular matrix multiplication. However, it is randomised, and only provides Monte Carlo algorithms running in the same time as the deterministic algorithms of Theorem 1, 2, and 3.

▶ **Reminder of Theorem 6.** *Given two lists $\mathcal{X}$ and $\mathcal{Y}$ of N vectors in $[r]^d$, the set of dissimilar pairs $S \subseteq \mathcal{X} \times \mathcal{Y}$ can be listed in $\tilde{O}(|S| + 2^d \cdot N)$ time with probability of success $1 - o(1)$.*

Proof. Let $\mathcal{H}$ be the family of hash functions $h : [r] \to \{0, 1\}$. Pick $t := 3 \cdot 2^d \log(N)$ vectors of d hash functions $h_j = (h_{j,1}, h_{j,2}, \ldots, h_{j,d}) \in \mathcal{H}^d$ for $j = 1, 2, \ldots, t$.

Consider a pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$. We say (x, y) *passes the hash* j if

$$\forall i \in 1, \ldots, d,\ h_{j,i}(x_i) \neq h_{j,i}(y_i). \tag{7}$$

Note that a similar vector pair never passes any j. Let $\varphi_{j,(x,y)}$ be the indicator for the event that a dissimilar pair (x, y) passes j, then

$$\mathbb{E}[\varphi_{j,(x,y)}] = \Pr[(x, y) \text{ pass } j] = \frac{1}{2^d}. \tag{8}$$

Form the sum

$$X = \sum_{j=1}^{t} \sum_{x,y \in S} \varphi_{j,(x,y)}. \tag{9}$$

The quantity X is the number of the vector pairs in S that survive some j, counted with multiplicity when they pass several hashes. Applying linearity of expectation to (8) and (9),

$$\mathbb{E}[X] = \frac{t|S|}{2^d} = 3|S| \log(N). \tag{10}$$

Applying Markov's inequality to (10),

$$\Pr\left[X > 10\mathbb{E}[X]\right] \leq \frac{1}{10}. \tag{11}$$

Also, the probability that a particular $(x, y) \in S$ does not pass any j, is

$$\prod_{j=1}^{2^d 3 \log N} \Pr[(x, y) \text{ does not pass } j] = \left(1 - \frac{1}{2^d}\right)^{3 \cdot 2^d \log(N)} < \exp(-3 \log N). \tag{12}$$

By a union bound, the probability that some dissimilar pair does not pass any j is at most

$$N^2 \exp(-3 \log N) < \frac{1}{N}. \tag{13}$$

Suppose we list all pairs (x, y) that pass some j. From (11) and (13), we see that with probability at least $\frac{9}{10} - \frac{1}{N}$, we will list all dissimilar pairs (possibly with repetition), and we do not list more than $30 \log N$ times the number of dissimilar pairs.

Now we describe how to list these pairs. Iterate over $j \in [t]$. For each j, iterate over $y = (y_1, \ldots, y_d) \in \mathcal{Y}$, compute its hash vector $h_j(y) := (h_{j,1}(y_1), \ldots, h_{j,d}(y_d)) \in \{0, 1\}^d$, and build lists $\ell_j(v)$ for all relevant vectors $v \in \{0, 1\}^d$, where $\ell_j(v) := \{y \in \mathcal{Y} \mid h_j(y) = v\}$. Note that $\sum_{v \in \{0,1\}^d} |\ell_j(v)| = N$. Next, iterate over $x = (x_1, \ldots, x_d) \in \mathcal{X}$, and output the pair (x, y') for each $y' \in \ell_j(\overline{h_j(x)})$, where $\overline{h_j(x)} := (1 - h_{j,1}(x_1), \ldots, 1 - h_{j,d}(x_d))$.

Observe the running time is $\tilde{O}(tN + X) \leq \tilde{O}(N \cdot 2^d + |S|)$. ◀

4 Reductions to listing dissimilar vectors

Here we outline reductions from our two application problems to the problem of listing dissimilar vectors, roughly following Bax and Franklin [3] for the permanent, and Björklund et al. [8] for Hamiltonian cycles. We also provide bounds on the number of dissimilar pairs that are needed for the analysis of the resulting algorithms. In particular, in Section 4.2 we provide a novel stronger bound on the number of dissimilar pairs than was previously known, for the reduction from 0/1 matrix permanents with d ones in each row and column.

4.1 Reduction for the permanent

In the following, let K be a ring on r elements denoted by $e_1, \ldots, e_r$.

Bax and Franklin [3] observed the following simple but intriguing fact. For any $M \in K^{n\times n}$, let 0_n be the $1 \times n$ row vector of all zeros, and let $q \in K^{n\times 1}$ be any column vector. Form the $(n+1) \times (n+1)$ matrix

$$M_q = \begin{bmatrix} M & q \\ 0_n & 1 \end{bmatrix}. \tag{14}$$

Then we have

$$\mathrm{per}(M_q) = \mathrm{per}(M). \tag{15}$$

The equation (15) follows because every summand in the permanent (1) must include the 1 on the last row of M_q, and hence cannot include any elements of the vector q. The usefulness of this simple fact can be seen from inspecting Ryser's formula (2), partitioned in the form

$$\mathrm{per}(M_q) = \sum_{X\subseteq[n]} (-1)^{n-|X|} f(M_q, X),\ f(M_q, X) = \prod_{i=1}^{n} g_i(M_q, X),\ g_i(M_q, X) = q_i + \sum_{j\in X} M_{i,j}. \tag{16}$$

Note that $f(M_q, X) = 0$ if and only if $g_i(M_q, X) = 0$ for some i. So in order to compute $\mathrm{per}(M)$, it is enough to list those subsets $X \subseteq [n]$ such that $g_i(M_q, X) \neq 0$ for all i, and accumulate their contributions. We call such X's *contributing terms*. Furthermore, we say a subset $X \subseteq [n]$ is *k-weakly contributing* if $g_i(M_q, X)$ is non-zero for all $i \leq k$.

In particular, by choosing q uniformly at random, we can easily compute the expected number of k-weakly contributing terms, as the events $g_i(M_q, X) \neq 0$ for $i = 1, \ldots, k$ are mutually independent (they depend on different q_i). Let Y be the random variable equal to the number of k-weakly contributing terms under a random $q \in K^{n\times 1}$, i.e., $Y = \sum_X Y_X$ where Y_X is the indicator of whether X is k-weakly contributing. By linearity of expectation,

$$\mathbb{E}(Y) = \sum_{X\subseteq[n]} \mathbb{E}(Y_X) = 2^n \left(1 - \frac{1}{r}\right)^k < 2^n \exp(-k/r). \tag{17}$$

Thus, if we could efficiently list the k-weakly contributing terms for some $k \geq \Omega(n)$, we would have a Las Vegas algorithm running in expected $2^{n-\Omega(n/r)}$ time. In Theorem 1, we claim a *deterministic* algorithm, in which case we cannot choose q at random. We will address this issue later in Section 5. For now, we concentrate on finding the contributing terms for a fixed q. We describe next how k-weakly contributing terms can be viewed as *dissimilar vector pairs* from two sets of short vectors.

4.1.1 Contributing terms as dissimilar vectors

Here we show how to reduce the problem of listing contributing terms to the problem of listing dissimilar vectors, implying a randomised version of Theorem 1. Assume WLOG that our matrix M is $n \times n$ and n is even; recall its entries are elements from the ring $K = \{e_1, \ldots, e_r\}$. We begin by partitioning the columns in two halves $L = \{1, \ldots, n/2\}$ and $R = \{n/2+1, \ldots, n\}$. Let $\mathcal{L}$ be the power set of L, and $\mathcal{R}$ the power set of R. Define a map $\phi : K \to [r]$ by $\phi(e_i) := i$.

For every $A \in \mathcal{L}$, construct a vector $v^A \in [r]^k$ for $i \in \{1, \ldots, k\}$ as

$$v_i^A = \phi\left(\sum_{k \in A} M_{i,k}\right), \text{ for all } i = 1, \ldots, k. \tag{18}$$

Similarly, but asymmetrically, for every $B \in \mathcal{R}$ we construct the vector $v^B \in [r]^k$ as

$$v_i^B = \phi\left(-q_i - \sum_{k \in B} M_{i,k}\right), \text{ for all } i = 1, \ldots, k. \tag{19}$$

Recall two vectors u and v are dissimilar if they differ in each coordinate. One can easily verify for all $A \in \mathcal{L}$ and $B \in \mathcal{R}$, the two vectors v^A and v^B are dissimilar if and only if the subset $X = A \cup B$ is k-weakly contributing. Note that $\mathcal{L}$ and $\mathcal{R}$ describe $N = 2^{n/2}$ vectors each of dimension k. For $k \geq \Omega(n)$, the number of k-weakly contributing terms in (17) is (expected to be) at most $2^{n-cn/r}$ for a constant $c > 0$. In that case, the number of dissimilar vector pairs s in our instance of $O(2^{n/2})$ vectors is at most $2^{n-cn/r}$.

Given an algorithm that efficiently lists dissimilar vector pairs, we can then efficiently list all $2^{n-cn/r}$ of the k-weakly contributing terms in the modified Ryser's formula (16). Given the list of contributing terms, we can then compute the permanent via (16), in time $2^{n-cn/r} \cdot \text{poly}(n)$. Using the listing algorithm of Theorem 4 with $s = 2^{n-cn/r}$, we obtain an algorithm with running time $2^{n-\Omega(\frac{n}{r})}$ as claimed.

4.2 The permanent algorithm for regular matrices

We now turn to giving a Las Vegas version of Theorem 2, showing that for d-regular matrices the permanent can be computed in $2^{n-\Omega(n/d^{3/4})}$ time. Later in Section 5, we will outline how to derandomise the algorithm.

Let $M \in \{0,1\}^{n \times n}$ have exactly d ones in every row and column. As in the case of permanents over finite rings, we reduce to dissimilar pair listing and apply the algorithm of Theorem 4 to locate k-weakly contributing terms for the permanent of the perturbed matrix M_q of (14). However, here we will pick the vector q from a different distribution, and we may also permute the rows of M to select a suitable subset of rows to use in the reduction to dissimilar pair listing. Our choices drastically reduce the number of contributing terms.

Index the rows and columns of M by $[n]$. For $i = 1, \ldots, n$, the ith row of M naturally corresponds to a d-size subset R_i of $[n]$ indicating which columns have a 1 in the row.

We first consider the case $d \geq n^{4/5}$. Here our algorithm will work as in Bax and Franklin [3] (whose analysis works for $d \in \Omega(n)$, for which the slightly better bound $2^{n-\Omega(n^{1/3})}$ can be argued). The probability for a fixed row $i \in [n]$, that the sieved row sum $|X \cap R_i|$ is deviating significantly from its expectation, is small:

$$\Pr_X\left[\left||X \cap R_i| - \frac{d}{2}\right| > d^{3/4}\right] \leq \exp(-\Omega(\sqrt{d})), \tag{20}$$

as seen by a standard Chernoff bound. Note that $\frac{n}{d^{3/4}} \leq n^{2/5} \leq \sqrt{d}$ for $d \geq n^{4/5}$. By a union bound, the probability that some row has intersection with X outside of the $2d^{3/4}$-length interval in (20) is at most $n \cdot \exp(-\Omega(\sqrt{d})) \leq \exp(-\Omega(n/d^{3/4}))$. Hence there are at most $2^{n-\Omega(n/d^{3/4})}$ such deviating subsets $X \subseteq [n]$.

Therefore, if we run our previous permanent algorithm but with the vector q taking random values in the range $[\frac{d}{2} - d^{3/4}, \frac{d}{2} + d^{3/4}]$, we see that the probability that any X within the $2d^{3/4}$-span for every row, is k-weakly contributing, is at most

$$\Pr_q[X \text{ passes}] \leq \left(1 - \frac{1}{2d^{3/4}}\right)^k. \tag{21}$$

Hence for $k \geq \Omega(n)$, the expected number of non-zero terms is only $2^{n-\Omega(n/d^{3/4})}$, so we only need to list that many dissimilar vector pairs in our reduction.

Now consider the more interesting case of $d < n^{4/5}$. First, we preprocess the matrix via a process we call P, which puts the row sets representing M into a number of ordered lists.

> **Process P:** Start with an empty list L_1, and put the first row set R_1 in L_1. While there is another row set R_i, not yet assigned to a list, that has at most $d^{3/4}$ elements in common with the union of the row sets in L_1, insert R_i into L_1. This operation is repeated until there either are no more row sets that meet this criterion, or L_1 contains $n/(2d^{5/4})$ row sets. In the latter case we say that the list is *full*. We consider L_1 *finished*, put it aside, and start building a new, initially empty, list L_2. We insert row sets into L_2 in the same way, continue with a third list L_3 once L_2 is finished, and so on. The process P terminates when every row set has been assigned to some list.

After we have constructed the lists $L_1, L_2, \ldots$, we reorder the rows and columns of M according to the insertion order of the row sets during process P. This permutation does not change the value of the permanent. The following lemma ensures that the first $n/2$ rows of M, after the reordering step, can be partitioned into full lists.

▶ **Lemma 7.** *The first $d^{5/4}$ lists produced by process P are full, i.e., each list contains $n/(2d^{5/4})$ row sets.*

Proof. Assume there are at least $n/2 + n/(2d^{5/4})$ row sets left to place when we start populating the list L_j. After we have put t sets into a list L_j, their union covers no more than td elements. Each element of $[n]$ is covered d times in total, since M is d-regular. Thus there are at most $td^{2-3/4}$ row sets left that has an overlap of more than $d^{3/4}$ with the union.

Therefore, as long as $\frac{n}{2} + n/(2d^{5/4}) - t > td^{2-3/4}$, i.e., $t < \frac{n}{2(d^{2-3/4})}$, there is still a row set that can be put into L_j. Hence we can put at least $\frac{n}{2d^{5/4}}$ row sets into L_j, making it full. We can repeat this for $j = 1, 2, \ldots, d^{5/4}$, as there are still $n/2 + n/(2d^{5/4})$ row sets left when we start to populate $L_{d^{5/4}}$. ◀

Next we need a Chernoff-like concentration bound on the sum of variables with bounded dependence resulting from a list L_j.

▶ **Lemma 8.** *Let L be an ordered list of d-subsets $S_1, ..., S_m$ of $[n]$ such that for all i, $|S_i \cap (\cup_{j=1}^{i-1} S_j)| < d^{3/4}$. Pick $X \subseteq [n]$ uniformly at random. Let Z_i be the indicator variable for the event $\frac{d}{2} - 3d^{3/4} \leq |S_i \cap X| \leq \frac{d}{2} + 3d^{3/4}$. Then*

$$\Pr_X\left[\sum_{i=1}^{m} Z_i < \frac{m}{2}\right] \leq \exp(-\Omega(d^{1/2}m)). \tag{22}$$

Proof. Let $U = [\ell] = \cup_{i=1}^{m} S_i$ and $X_1, \ldots, X_\ell$ be indicator variables such that $X_i = 1 \Leftrightarrow i \in X$. Hence the X_i's are mutually independent with $E[X_i] = 1/2$. Construct the sets $T_i = S_i \setminus (\cup_{l=1}^{i-1} S_l)$ for $i = 1, \ldots, m$. Note that $|T_i| \geq d - d^{3/4}$ for all i, and that T_i and T_j are disjoint for $i \neq j$. Letting $Y_i = \sum_{j \in T_i} X_j$, we have

$$\Pr_X \left[\left| Y_i - \frac{d}{2} \right| > 2d^{3/4} \right] \leq \exp(-\Omega(d^{1/2})), \tag{23}$$

as seen by a standard Chernoff bound. Note that if $|Y_i - d/2| \leq 2d^{3/4}$, then also $Z_i = 1$, as there are only at most $d^{3/4}$ other elements of X that are in $S_i \setminus T_i$. Further note that the events $|Y_i - d/2| > 2d^{3/4}$ for different i are mutually independent as they depend on different mutually independent X_j's. Hence the probability that $\sum_i Z_i < m/2$ is at most $m/2 \cdot \exp(-\Omega(d^{1/2}))^{m/2} \binom{m}{m/2} \leq \exp(-\Omega(d^{1/2} m))$. ◀

We can now argue as in the dense case (where d is large). We first bound the number of X's such that for some list L_i, more than half of the row sets in the list have a deviating sieve row sum $\left| |R_j \cap X| - \frac{d}{2} \right| > 3d^{3/4}$. From Lemma 8, replacing $m = n/(2d^{5/4})$, and a union bound over all lists, we know that this happens for at most $2^{n - \Omega(n/d^{3/4})}$ X's. Thus we may restrict our analysis to the X's that are within the $6d^{3/4}$-length interval for at least half of the rows in the first $d^{5/4}$ lists (amounting to the first $n/2$ rows of the matrix, after the pre-processing reordering). Running our algorithm from before, but with the vector q taking random values in the range $[\frac{d}{2} - 3d^{3/4}, \frac{d}{2} + 3d^{3/4}]$, we see that each of these remaining well-behaved X's are k-weakly contributing with probability at most

$$\Pr_q[X \text{ passes}] \leq \left(1 - \frac{1}{6d^{3/4}}\right)^{k/2}. \tag{24}$$

Hence, for $k = \frac{n}{c}$ for some $c > 2$, we have in expectation $2^{n - \Omega(n/d^{3/4})}$ non-zero terms.

4.3 Reduction for counting Hamiltonian cycles

In the following let $G' = (V', A')$ be the directed input graph on $n = |V'|$ vertices in which we want to count Hamiltonian cycles. Let d_v for $v \in V'$ be the out-degree of the vertex v, and let $\delta = |A'|/n$ be the average out-degree of G'. It will be convenient to work on a slightly modified graph $G = (V, E)$ constructed from G' as follows: pick an arbitrary vertex $s' \in V'$, and obtain G from G' by replacing s' with two new vertices s and t (i.e., $V = V' \setminus \{s'\} \cup \{s, t\}$), where s retains all outgoing arcs from s', i.e. $(s', u) \in A' \Leftrightarrow (s, u) \in A$, and t retains all incoming arcs to s', i.e. $(u, s') \in A' \Leftrightarrow (u, t) \in A$. Note that the Hamiltonian paths from s to t in G are in one-to-one correspondence with the Hamiltonian cycles in G', and that the average degree of G is not larger than that of G'. In the following, we consider the problem of counting the s-t Hamiltonian paths on the modified $n + 1$ vertex graph G.

4.3.1 Random columns and contributing terms

Björklund, Kaski, and Koutis [8] observed that the number of Hamiltonian cycles in a directed graph can be evaluated as an inclusion–exclusion summation over a determinant of a polynomial matrix representing the graph. We will use their construction, and restate the main ideas here. The Laplacian of the graph G, is a $(n + 1) \times (n + 1)$ polynomial matrix

with rows and columns indexed by the vertices V, in the variables x_v for $v \in V$:

$$L(G)_{i,j} = \begin{cases} \sum_{(u,v)\in A} x_u & \text{if } i = j = v \\ -x_u & \text{if } i = u, j = v, (u,v) \in A \\ 0 & \text{otherwise.} \end{cases} \tag{25}$$

The Laplacian *punctured at the vertex* $s \in V$, is the matrix $L(G)_s$ obtained by omitting row and column s from $L(G)$. By Tutte's directed version of the Matrix-Tree theorem of Kirchhoff [20], we know that $\det(L(G)_s)$ is a polynomial where each term corresponds to a directed spanning out-branching rooted at s. Denote by $\mathrm{hp}(G)_{s,t}$ the number of Hamiltonian paths starting in s and ending in t. By the principle of inclusion–exclusion, we have[4]

$$\mathrm{hp}(G)_{s,t} = \sum_{\substack{x:(V\setminus\{t\})\to\{0,1\} \\ x_t=1}} (-1)^{n-1-|x|} \det\left(L(G)_s(x)\right). \tag{26}$$

The summation is over all 2^{n-1} assignments x with the restriction that $x_t = 1$. Next, we mimic the Bax and Franklin idea for computing permanents by perturbing the matrix (of (15)): we shall parametrise the Laplacian matrices so that in expectation, many summands in the above formula are zeroed-out. To this end we introduce fresh random variables $q_v \in \{0, 1, \ldots, d_v\}$ for $v \in V$, and define the *q-perturbed Laplacian* of G as

$$L_q(G)_{i,j} = \begin{cases} \sum_{(u,v)\in A} x_u - q_v & \text{if } i = j = v \\ -x_u & \text{if } i = u, j = v, (u,v) \in A \\ 0 & \text{otherwise.} \end{cases} \tag{27}$$

The extra q_v variables may be thought of as weighted arcs originating from t: these arcs cannot be used by any of the Hamiltonian paths from s to t. The point of our perturbation is that irrespective of q, we can still compute the number of Hamiltonian paths:

$$\mathrm{hp}(G')_{s',t'} = \sum_{\substack{x:(V\setminus\{t\})\to\{0,1\} \\ x_t=1}} (-1)^{n-1-|x|} \det\left(L_q(G)_s(x)\right). \tag{28}$$

However, in expectation, many assignments x may yield $\det\left(L_q(G)_s(x)\right) = 0$, particularly when a row in $L_q(G)_s(x)$ is all-zeroes. Observe that a row in $L_q(G)_s(x)$ is all-zero if and only if for some $v \in V \setminus \{s\}$ we have $x_v = 0$ and $\sum_{(u,v)\in A} x_u = q_v$. Let ε_v be a Boolean variable that is true if and only if $x_v = 0$ and $\sum_{(u,v)\in A} x_u = q_v$; i.e., $L_q(G)_s(x)$ is all-zero in the row indexed by v. In analogy with the case of the permanent, we say that an assignment x is *contributing* if ε_v is false for all vertices $v \in V \setminus \{s\}$, and x is *k-weakly contributing* if ε_v is false for the first k vertices. As in the case of the permanent, it is sufficient to list the k-weakly contributing terms for some k to compute the number of Hamiltonian paths in G (and hence the Hamiltonian cycles in G') through the formula of (28).

We can easily compute the expected number of contributing terms for a $q \in \{0, 1, \ldots, d_v\}^n$ picked uniformly at random, since the events ε_v are mutually independent (they depend on different q-values). The probability that $L_q(G)_s(x)$ is all-zero in the row v, with $x_v = 0$, is

$$\Pr_q[\varepsilon_v] = \frac{1}{(d_v + 1)}. \tag{29}$$

We will assign $k := cn$ for some $c < 1/2$, and assume WLOG that the vertices are sorted by increasing out-degree. This means by an averaging argument that for all $i \leq k$, $d_i < 2\delta$.

[4] See [8] for a proof of (26).

Let Z_x be the indicator variable that the assignment x is k-weakly contributing, under a randomly chosen q. Let Z be the random variable equal to the number of k-weakly contributing terms under a random q, i.e., $Z = \sum_x Z_x$. By linearity of expectation, and Jensen's inequality for concave functions,

$$\mathbb{E}(Z) = \sum_{X \subseteq V \setminus \{s\}} \prod_{v \in X \cap [k]} \left(1 - \frac{1}{d_v + 1}\right) \leq 2^n \left(1 - \frac{1}{2\delta + 1}\right)^k \leq 2^n \cdot \exp(-\Omega(k/\delta)). \quad (30)$$

Thus, the expected number of k-weak contributors behaves similarly as in previous cases.

4.3.2 Contributing terms as dissimilar vectors

We now turn to describing how the contributing terms can be encoded as dissimilar vectors. Assume WLOG that n is even, and identify $V \setminus \{s\}$ with the set $[n]$. Partition the vertices into $L = \{1, \ldots, n/2\}$ and $R = \{n/2 + 1, \ldots, n\}$, letting $\mathcal{L}$ and $\mathcal{R}$ be the power sets of L and R, respectively. For every $C \in \mathcal{L}$, we construct the vector $v^C \in [2\delta + 1]^k$ for $i = 1, \ldots, k$ as

$$v_i^C = \begin{cases} \sum_{(i,w) \in (A \cap C)} 1 & \text{if } i \notin C, \\ \star & \text{otherwise.} \end{cases} \quad (31)$$

where $\star$ is an extra symbol encoded as 2δ. Similarly, but asymmetrically, noting in particular that $i \notin R$ because $k < n/2$, for every $D \in \mathcal{R}$ we construct the k-length vector v^D as

$$v_i^D = q_i - \sum_{(i,w) \in (A \cap D)} 1, \text{ for all } i = 1, \ldots, k. \quad (32)$$

It is readily verified that for $C \in \mathcal{L}$ and $D \in \mathcal{R}$, the two vectors v^C and v^D are dissimilar if and only if the first k columns of $L_q(G)(x')$ are non-zero, where the Boolean vector x' is defined as $x'_v := 1 \iff v \in C \cup D$. Hence, the k-weakly contributing assignments x are precisely those corresponding to dissimilar pairs (v^C, v^D).

Note that $\mathcal{L}$ and $\mathcal{R}$ each contain $N = 2^{n/2}$ vectors of dimension k. Using a sufficiently fast algorithm for listing dissimilar vector pairs, we can enumerate all k-weakly contributing terms in (28) in $N^{2-\Omega(1/\delta)}$ time. Once we have a list of all the $2^n \cdot \exp(-\Omega(k/\delta))$ contributing terms, the number of Hamiltonian cycles can be computed with (28) in time $2^n \cdot \exp(-\Omega(k/\delta))$ as well. From (30), we know that the upper bound s on the number of dissimilar pairs can be set to $2^{n - cn/\delta}$, for some positive $c > 0$. Applying the dissimilar pair listing algorithm of Theorem 4, we arrive at the running time $2^{n - \Omega(\frac{n}{\delta})}$. This concludes the algorithm.

5 Derandomisation

Our deterministic algorithm for dissimilar pair listing from Section 2 can be used to list k-weakly contributing terms efficiently in all our desired applications (Theorems 1, 2, and 3), provided that the number of contributing terms is not much more than their expected number would be on a random vector q.

To make all of our application algorithms fully deterministic, we must provide a deterministic procedure for setting the vector q so that the number of resulting terms is bounded by the expectation. This can be done by using other known algorithmic tools, along with the well-known method of conditional expectation (see e.g., [17]).

It turns out that a fast deterministic algorithm for counting dissimilar pairs will suffice. This can be obtained by reducing our problem to counting orthogonal pairs:

▶ **Proposition 9.** *Counting the dissimilar pairs on two sets of N vectors in $[r]^d$ can be reduced in $O(Nrd)$ time to counting orthogonal pairs on two sets of N vectors in $\{0,1\}^{rd}$.*

Proof. Let $e_1, \ldots, e_r \in \{0,1\}^r$ be the standard basis vectors, where e_i is 1 in the ith component and is 0 everywhere else. Map every vector $x \in [r]^d$ to the Boolean vector

$$\rho(x) = [e_{x_1}\ e_{x_2}\ \cdots\ e_{x_d}].$$

That is, $x' \in \{0,1\}^{rd}$ is obtained by *concatenating* the d standard basis vectors corresponding to the entries of x. Note $x, y \in [r]^d$ are dissimilar $\iff$ $\rho(x)$ and $\rho(y)$ are orthogonal. ◀

Applying the proposition, we can count dissimilar pairs by counting OV pairs:

▶ **Theorem 10** (Chan and Williams [11])**.** *For every $c \leq 2^{o(\sqrt{\log N})}$, there is a deterministic algorithm that for two sets of N vectors from $\{0,1\}^{c \log N}$, counts the orthogonal vector pairs in $N^{2-1/O(\log c)}$ time.*

An immediate corollary is that, for $d \leq \log(N)$, we can deterministically count all dissimilar pairs over N vectors in $[r]^d$ in only $N^{2-1/O(\log r)}$ time. We can use this counting algorithm to *deterministically* search for a vector q that has at most as many k-weakly contributing terms as the expected number for a random q. We describe the procedure generically, as it is essentially the same for all three applications in the paper:

> Iterate over $j = 1, \ldots, k$. Suppose we have determined the first $j-1$ components of our vector q, and we wish to determine the jth component, q_j. Let us inductively suppose that there are at most $2^n \cdot \prod_{i=1}^{j-1}(1 - 1/C_i)$ contributing terms remaining (where the C_i depend on the application), and let there be C_j possible values for q_j. Construct and compute C_j distinct instances of dissimilar pair counting with j-dimensional vectors, corresponding to the C_j different values for q_j. Finally, set q_j to be the value which minimises the number of dissimilar pairs obtained.

Since we always choose q_j to minimise the number of dissimilar pairs, and we know a random setting of q_j reduces the number by a $(1 - 1/C_j)$ fraction in expectation (by our analyses in previous sections), our k-dimensional vector q produces a number of k-weakly contributing terms which is at most the expectation. Finally, note that this initial procedure for selecting the vector q is much faster than the overall running time in our applications.

This concludes our derandomisation, and the proofs of Theorems 1, 2, and 3.

References

1. Amir Abboud, Ryan Williams, and Huacheng Yu. More Applications of the Polynomial Method to Algorithm Design. In *Proceedings of the Twenty-sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 218–230, Philadelphia, PA, USA, 2015. Society for Industrial and Applied Mathematics.
2. Arturs Backurs and Piotr Indyk. Edit Distance Cannot Be Computed in Strongly Subquadratic Time (Unless SETH is False). In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 51–58, New York, NY, USA, 2015. ACM.
3. Bax and Franklin. A Permanent Algorithm with $\exp[\Omega(N^{1/3}/2 \ln N)]$ Expected Speedup for 0-1 Matrices. *Algorithmica*, 32(1):157–162, January 2002.
4. Andreas Björklund. Determinant Sums for Undirected Hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014. doi:10.1137/110839229.

5 Andreas Björklund and Thore Husfeldt. The Parity of Directed Hamiltonian Cycles. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 727–735, 2013. doi:10.1109/FOCS.2013.83.

6 Andreas Björklund, Thore Husfeldt, and Isak Lyckberg. Computing the permanent modulo a prime power. *Inf. Process. Lett.*, 125:20–25, 2017. doi:10.1016/j.ipl.2017.04.015.

7 Andreas Björklund, Petteri Kaski, and Ioannis Koutis. Directed Hamiltonicity and Out-Branchings via Generalized Laplacians. *CoRR*, abs/1607.04002, 2016. arXiv:1607.04002.

8 Andreas Björklund, Petteri Kaski, and Ioannis Koutis. Directed Hamiltonicity and Out-Branchings via Generalized Laplacians. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 91:1–91:14, 2017. doi:10.4230/LIPIcs.ICALP.2017.91.

9 Andreas Björklund, Petteri Kaski, and Ryan Williams. Generalized Kakeya sets for polynomial evaluation and faster computation of fermionants. *Algorithmica*, September 2018.

10 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic Single Exponential Time Algorithms for Connectivity Problems Parameterized by Treewidth. *Inf. Comput.*, 243(C):86–111, August 2015.

11 Timothy M. Chan and Ryan Williams. Deterministic APSP, Orthogonal Vectors, and More: Quickly Derandomizing Razborov-smolensky. In *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 1246–1255, Philadelphia, PA, USA, 2016. Society for Industrial and Applied Mathematics.

12 Don Coppersmith. Rapid Multiplication of Rectangular Matrices. *SIAM J. Comput.*, 11(3):467–471, 1982. doi:10.1137/0211037.

13 Marek Cygan and Marcin Pilipczuk. Faster Exponential-time Algorithms in Graphs of Bounded Average Degree. *Inf. Comput.*, 243(C):75–85, August 2015.

14 Taisuke Izumi and Tadashi Wadayama. A New Direction for Counting Perfect Matchings. In *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, FOCS '12, pages 591–598, Washington, DC, USA, 2012. IEEE Computer Society.

15 Kasper Green Larsen and Ryan Williams. Faster Online Matrix-vector Multiplication. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 2182–2189, Philadelphia, PA, USA, 2017. Society for Industrial and Applied Mathematics.

16 Maciej Liśkiewicz, Mitsunori Ogihara, and Seinosuke Toda. The complexity of counting self-avoiding walks in subgraphs of two-dimensional grids and hypercubes. *Theoretical Computer Science*, 304(1):129–156, 2003.

17 Prabhakar Raghavan. Probabilistic Construction of Deterministic Algorithms: Approximating Packing Integer Programs. *J. Comput. Syst. Sci.*, 37(2):130–143, October 1988.

18 Herbert John Ryser. *Combinatorial Mathematics*. Mathematical Association of America, 1963.

19 Rocco A. Servedio and Andrew Wan. Computing sparse permanents faster. *Information Processing Letters*, 96(3):89–92, 2005.

20 W. T. Tutte. The dissection of equilateral triangles into equilateral triangles. *Mathematical Proceedings of the Cambridge Philosophical Society*, 44(4):463–482, 1948.

21 Leslie G. Valiant. The Complexity of Enumeration and Reliability Problems. *SIAM J. Comput.*, 8:410–421, 1979.

22 L.G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

Solving Systems of Polynomial Equations over GF(2) by a Parity-Counting Self-Reduction

Andreas Björklund
Department of Computer Science, Lund University, Sweden

Petteri Kaski
Department of Computer Science, Aalto University, Finland

Ryan Williams
Department of Electrical Engineering and Computer Science & CSAIL, MIT, Cambridge, MA, USA

Abstract

We consider the problem of finding solutions to systems of polynomial equations over a finite field. Lokshtanov et al. [SODA'17] recently obtained the first worst-case algorithms that beat exhaustive search for this problem. In particular for degree-d equations modulo two in n variables, they gave an $O^*\big(2^{(1-1/(5d))n}\big)$ time algorithm, and for the special case $d = 2$ they gave an $O^*\big(2^{0.876n}\big)$ time algorithm.

We modify their approach in a way that improves these running times to $O^*\big(2^{(1-1/(2.7d))n}\big)$ and $O^*\big(2^{0.804n}\big)$, respectively. In particular, our latter bound – that holds for all systems of quadratic equations modulo 2 – comes close to the $O^*\big(2^{0.792n}\big)$ expected time bound of an algorithm empirically found to hold for *random* equation systems in Bardet et al. [*J. Complexity*, 2013]. Our improvement involves three observations:

1. The Valiant-Vazirani lemma can be used to reduce the solution-finding problem to that of counting solutions modulo 2.
2. The monomials in the probabilistic polynomials used in this solution-counting modulo 2 have a special form that we exploit to obtain better bounds on their number than in Lokshtanov et al. [SODA'17].
3. The problem of solution-counting modulo 2 can be "embedded" in a smaller instance of the original problem, which enables us to apply the algorithm as a subroutine to itself.

2012 ACM Subject Classification Mathematics of computing → Combinatorial algorithms

Keywords and phrases equation systems, polynomial method

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.26

Category Track A: Algorithms, Complexity and Games

Funding *Andreas Björklund*: The Swedish Research Council grant VR 2016-03855 "Algebraic Graph Algorithms".
Petteri Kaski: The European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement 338077 "Theory and Practice of Advanced Search and Enumeration".
Ryan Williams: The U.S. National Science Foundation under grants CCF-1741638 and CCF-1741615.

1 Introduction

We study the problem of finding a simultaneous root to a system of m polynomials

$$P_1(x) = 0\,, \quad P_2(x) = 0\,, \quad \ldots\,, \quad P_m(x) = 0 \tag{1}$$

over n variables $x = (x_1, x_2, \ldots, x_n)$. The computational tractability of this problem is known to dramatically depend on the domain of the variables and polynomial coefficients. Over the integers, the problem is undecidable, by Matiyasevich's celebrated solution of

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 26; pp. 26:1–26:13

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

Hilbert's tenth problem on the algorithmic decidability of Diophantine equations [10]. Over an algebraically closed field, the problem reduces via Hilbert's Nullstellensatz to deciding whether 1 belongs to the ideal generated by the polynomials, which for polynomials with rational coefficients can be decided in exponential space by computing a reduced Gröbner basis for the ideal [3, 4, 11]. Over the integers *modulo two* – our object of study in this paper – the problem is NP-complete even when the polynomials are severely constrained. Indeed, systems of polynomial equations modulo two enable the compact modeling of a versatile range of tasks. For example, one can easily represent k-CNFSAT formulas on n Boolean variables by expressing each clause in the formula as a degree-k polynomial [5]. A similar reduction from NAE3SAT proves that the problem remains NP-hard even in the case of quadratic equations modulo two. As the fastest known worst-case algorithms for k-CNF satisfiability (for example, see Moser and Scheder [13]) run in time $2^{n-\Omega(n/k)}$, it is a difficult challenge to design faster-than-$2^{n-\Omega(n/k)}$-time algorithms for solving systems of degree-k polynomial equations. Still it is interesting to ask precisely how much savings over the brute-force $O^*\big(2^n\big)$-time solution one can obtain, particularly in the case of quadratic equations, since the postulated hardness of solving quadratic systems modulo two forms the basis of several proposed cryptographic primitives, such as HFE proposed by Patarin [15] and UOV proposed by Kipnis, Patarin, and Goubin [8].

An $O^*\big(2^{0.792n}\big)$ expected-time algorithm for a system of $m = n$ quadratic polynomials over n variables modulo two was proposed by Bardet, Faugère, Salvy, and Spaenlehauer [1]. However, their algorithm only works for systems satisfying certain algebraic assumptions, and these assumptions were only experimentally verified to hold for the vast majority of such systems. Still, further refinements of the method makes it practical even for small systems [6], and the algorithmic ideas underlies the to date fastest known implementation we are aware of [14].

Such a result highlights the potential vulnerability of cryptographic primitives whose security is based on the postulated hardness of solving random systems. However, from a theoretical viewpoint it is preferable to have rigorous proofs and algorithms that work efficiently on all inputs. The algorithm of Bardet et al. [1] is based on finding proofs of non-solvability, a so-called effective Nullstellensatz of finding low-degree polynomials H_i such that

$$\sum_i H_i(x')P_i(x', r) = 1,$$

for each specialisation r of the polynomials, i.e. after replacing a fixed subset of the variables to a restriction r. The algorithm then continues to look for solutions only in those specialisations r for which no proof was found. The search for proofs is formulated as a linear equation system whose dimensions depend on the bound of the degree in the proofs. The argument made in their paper is that for most equation systems with sufficiently more equations than variables, "small-degree" polynomials can be used in the proofs. Getting rigorous bounds on the degree appears to be a difficult problem, and in the worst case some systems will likely require large-degree proofs.

Lokshtanov, Paturi, Tamaki, Williams, and Yu [9] took a radically different approach and presented an $O^*\big(2^{0.876n}\big)$-time algorithm for quadratic equation systems modulo two that uses no algebraic assumptions at all and works for all quadratic systems. They also gave a general $O^*\big(2^{n-n/(5d)}\big)$ time algorithm for systems of equations of degree bounded by d.

Their approach uses the so-called "polynomial method": the entire system of equations is randomly replaced by a single "probabilistic" polynomial that has a "small" exponential number of terms, and is consistent with the system on exponentially many assignments. This polynomial is then evaluated quickly on many assignments using an FFT or fast matrix multiplication, gaining an advantage over exhaustive search.

We present a new algorithm that largely follows the approach of Lokshtanov et al. [9] but offers a few simplifications to their scheme. Most prominently, we will consider the problem of computing the parity of the number of solutions, rather than the decision problem directly. Whereas Lokshtanov et al. [9] apply a random parity sieve on monomials of a subset of the variables to implement a decision-to-parity reduction within the algorithm, our main observation is that this can be done on the system itself. That is, rather than explicitly being tailored into the algorithm, we can reduce decision to the parity problem by adding random affine equations to the system. This is based on the well-known theorem of Valiant and Vazirani [18] in complexity theory to isolate solutions to Boolean satisfiability by adding random equations. Our analysis in Section 2.5 is borrowed from theirs and is included here solely for the sake of completeness.

One immediate effect of our alternative parity-counting approach is that it reduces the need for random bits from exponential in n to merely polynomial in n. A more interesting gain is that our approach leads to faster algorithms, via two further observations. Our algorithm for quadratic systems runs in $O^*\big(2^{0.804n}\big)$ time, and for degree-d systems we provide a $O^*\big(2^{n-n/(2.7d)}\big)$ time algorithm. Our running time for quadratic equation systems in particular comes much closer to the $O^*\big(2^{0.792n}\big)$ running time of Bardet et al. [1]. To get a quantitive feeling of our incremental result, note:

1. We can solve quadratic systems modulo two with 9% more variables in about the same time as the algorithm of [9].
2. Our algorithm for degree-3 systems is faster than the one for degree-2 systems in [9].

Our first observation is that the seemingly more difficult problem of computing the parity of solutions apparently makes it easier to identify the structure of monomials in the probabilistic polynomials used in the polynomial method. Making use of this structure leads to better bounds on their number and (indirectly) on the total running time, and is the source of most of our improvement. Our second observation is that the parity-summation part in the method is identical to the original problem, leading to a self-reduction: we can use our algorithm as a subroutine to itself, again leading to a faster algorithm.

We present our algorithm for computing the parity of the number of solutions to a polynomial equation system in Section 3. We shall highlight the differences to the original decision algorithm by Lokshtanov et al. [9] as we go along. We begin by some preliminaries in the subsequent section.

2 Preliminaries

Here we review some notation and well-known facts. Let $\mathbb{F}_2$ denote the field of two elements; that is, integer arithmetic modulo two. For a non-negative integer n, we write $[n]$ for the set $\{1, 2, \ldots, n\}$. For a finite set D, we write 2^D for the power set of D, $\binom{D}{k}$ for the set of all k-element subsets of D, and $\binom{D}{\downarrow k} = \bigcup_{j=0}^{k} \binom{D}{j}$ for the set of all at-most-k-element subsets of D. Accordingly, we write $\binom{n}{\downarrow k} = \sum_{j=0}^{k} \binom{n}{j}$. For a function $f(n)$, the notation $O^*\big(f(n)\big)$ suppresses factors polynomial in n.

2.1 Yates's algorithm

Let us write I_ℓ for an $\ell \times \ell$ identity matrix. For an $s \times t$ matrix A and a non-negative integer n, the n^{th} Kronecker power of A factors into the sequence of matrices

$$A^{\otimes n} = \prod_{j=1}^{n} \left(I_s^{\otimes j-1} \otimes A \otimes I_t^{\otimes n-j} \right). \tag{2}$$

Each matrix $I_s^{\otimes j-1} \otimes A \otimes I_t^{\otimes n-j}$ is sparse with at most $s^{j-1} \cdot st \cdot t^{n-j} = s^j t^{n-j+1}$ nonzero entries. Thus, using sparse matrix–vector multiplication along the sequence (2), we may multiply the matrix $A^{\otimes n}$ with a given vector in at most

$$2\sum_{j=1}^{n} s^j t^{n-j+1} = \begin{cases} 2ns^{n+1} & \text{if } s = t, \\ \frac{2st(s^n - t^n)}{s-t} & \text{if } s \neq t \end{cases} \tag{3}$$

operations on scalars. This algorithm is known as Yates's algorithm [19].

2.2 The fast zeta transform for the subset lattice

The matrix $\zeta = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$ is invertible over any ring, with inverse $\zeta^{-1} = \left[\begin{smallmatrix} 1 & 0 \\ -1 & 1 \end{smallmatrix}\right]$. In particular, in the field $\mathbb{F}_2$ of two elements, we have $\zeta = \zeta^{-1}$. Let x and y be vectors whose components are indexed by the subsets in $2^{[n]}$. Then, the matrix–vector multiplication $y = \zeta^{\otimes n} x$ implements the linear map $x \mapsto y$ defined for all $B \subseteq [n]$ by $y_B = \sum_{A \subseteq B} x_A$. This map is the *zeta transform* for the lattice $(2^{[n]}, \subseteq)$. By (3) in Section 2.1, Yates's algorithm can be used to implement the zeta transform in $O(2^n n)$ operations. This algorithm is known as the *fast zeta transform*. The inverse transform is called the Möbius transform. In characteristic 2, these transforms coincide. The zeta transform remains invertible when the relevant vectors and matrices are restricted from $2^{[n]}$ to $\binom{[n]}{\downarrow d}$. The corresponding restriction of Yates's algorithm runs in $O\left(\binom{n}{\downarrow d} n\right)$ operations. See [2, 7] and the references therein for more on fast zeta transforms.

2.3 Polynomials modulo two: the monomial basis and the evaluation basis

Observe $x^2 = x$ holds for all $x \in \mathbb{F}_2$. Thus, WLOG, an n-variate polynomial $f = f(x_1, x_2, \ldots, x_n)$ in the polynomial ring $\mathbb{F}_2[x_1, x_2, \ldots, x_n]$ consists of only *multilinear* monomials indicated by a function $M_f : 2^{[n]} \to \mathbb{F}_2$ with

$$f = \sum_{Y \subseteq [n]} M_f(Y) \prod_{j \in Y} x_j \,.$$

Intuitively, $M_f(S)$ gives the coefficient of $\prod_{i \in S} x_i$ in the (unique) multilinear polynomial representing f. In particular, $\mathbb{F}_2[x_1, x_2, \ldots, x_n]$ is a 2^n-dimensional vector space over $\mathbb{F}_2$, and the function M_f, viewed as a vector with entries indexed by $2^{[n]}$, represents f in the *monomial basis*. We say that f has *degree* at most d if M_f vanishes outside $\binom{[n]}{\downarrow d}$. The monomial basis is the algebraic normal form of the Boolean function.

Associate each vector $x \in \mathbb{F}_2^n$ with the subset $X = \{i \in [n] : x_i = 1\} \subseteq [n]$. Define the *evaluation map* $E_f : 2^{[n]} \to \mathbb{F}_2$ for all $X \subseteq [n]$ by the rule $E_f(X) = f(x)$, where x is the vector corresponding to X. In what follows we often find it convenient to abuse notation slightly and write simply $f(X)$ in place of $E_f(X)$. Viewing the function E_f as a vector with entries indexed by $2^{[n]}$, we say that E_f represents f in the *evaluation basis*.

The monomial basis and the evaluation basis are related by the zeta transform. That is, for all $f \in \mathbb{F}_2[x]$, we have

$$E_f = \zeta^{\otimes n} M_f \,. \tag{4}$$

Indeed, for all $Z \subseteq [n]$ we have

$$E_f(Z) = f(Z) = \sum_{Y \subseteq Z} M_f(Y) \,.$$

By properties of the zeta transform, the basis-change identity (4) holds also when restricted from $2^{[n]}$ to $\binom{[n]}{\downarrow d}$; that is, when restricted from arbitrary polynomials to polynomials of degree at most d. Fast zeta transforms enable fast basis changes between the monomial basis and the evaluation basis as needed.

2.4 From finding to decision, from decision to parity-counting

The task of finding a solution to a given system of polynomial equations reduces to the task of deciding whether a given system has at least one solution. Indeed, assuming the system has a solution, we may try both values 0 and 1 to a selected variable, and focus on one assignment that indicates that the system after the substitution of the value has a solution. Thus, finding a solution takes at most $2n$ queries to a decision algorithm.

The task of deciding whether a given system of polynomial equations has a solution reduces to computing the parity of the number of such solutions by randomized isolation techniques. One elegant isolation technique is Valiant–Vazirani [18] affine hashing, which inserts $O(n)$ random linear equations into the system, without increasing the number of variables. For completeness, we recall affine hashing in Section 2.5. Thus, from here on, we consider the problem of counting the parity of solutions to a system of polynomial equations.

2.5 Valiant–Vazirani affine hashing

For completeness, this section recalls Valiant–Vazirani [18] affine hashing for isolating a unique solution (if any) by introducing a collection of random linear equations into to the system of polynomial equations. In particular, affine hashing does not increase the number of variables or the degree of the system, only the number of equations increases.

Let $\mathcal{S} \subseteq \{0,1\}^n$ be the set of solutions the system of polynomial equations. If $\mathcal{S}$ is empty there is nothing to isolate, so let us assume that $\mathcal{S}$ is nonempty in what follows. Let $k = 0, 1, \ldots, n$ be the unique integer such that $2^k \leq |\mathcal{S}| < 2^{k+1}$.

Draw independent uniform random values $\alpha_{ij} \in \{0,1\}$ for $i = 1, 2, \ldots, k+2$ and $j = 1, 2, \ldots, n$. For each $i = 1, 2 \ldots, k+2$, draw an independent uniform random value $\beta_i \in \{0,1\}$ and introduce the linear equation

$$\sum_{j=1}^{n} \alpha_{ij} x_j = \beta_i \tag{5}$$

into the system of polynomial equations.

Let us say that a solution $x \in \mathcal{S}$ *survives* if it satisfies every introduced equation (5). Let us write S_x for the event that x survives, and U_x for the event that x is the unique solution in $\mathcal{S}$ that survives. We want to control the probability

$$\Pr(U_x) = \Pr\bigl(S_x \cap \textstyle\bigcap_{y \in \mathcal{S} \setminus \{x\}} \bar{S}_y\bigr) \,.$$

By the union bound, we have

$$\Pr\big(S_x \cap \bigcap_{y\in\mathcal{S}\setminus\{x\}} \bar{S_y}\big) \geq \Pr(S_x) - \sum_{y\in\mathcal{S}\setminus\{x\}} \Pr\big(S_x \cap S_y\big)\,.$$

By independence of the events S_x and S_y for all $x, y \in \mathcal{S}$ with $x \neq y$, we have

$$\Pr(S_x) - \sum_{y\in\mathcal{S}\setminus\{x\}} \Pr\big(S_x \cap S_y\big) = \Pr(S_x)\left(1 - \sum_{y\in\mathcal{S}\setminus\{x\}} \Pr(S_y)\right).$$

By mutual independence of the $k+2$ equations, we have

$$\Pr(S_x) = \frac{1}{2^{k+2}}\,.$$

Hence,

$$\Pr(U_x) \geq \frac{1}{2^{k+2}}\left(1 - \frac{2^{k+1}}{2^{k+2}}\right) = \frac{1}{2^{k+3}}\,.$$

By mutual exclusiveness of the events U_x, we have

$$\Pr\big(\bigcup_{x\in\mathcal{S}} U_x\big) = \sum_{x\in\mathcal{S}} \Pr(U_x) \geq 2^k \frac{1}{2^{k+3}} = \frac{1}{8}\,.$$

From $(1-\frac{1}{8})^r \leq \exp(-\frac{r}{8}) \leq \epsilon$ we observe that $r = \lceil \ln \epsilon^{-1} \rceil$ independent repetitions will isolate a unique solution in $\mathcal{S}$ with probability at least $1-\epsilon$. Furthermore, we do not know the value of k, but we can exhaustively try out all the values $k = 0, 1, \ldots, n$ with $\epsilon = \frac{1}{n}$ so that a solution, if one exists, will be isolated and hence witnessed as odd parity in the solution space with high probability in total $O(n \log n)$ repetitions of the parity-counting algorithm.

3 A randomized reduction from parity-counting to itself

This section presents our technical contribution. All arithmetic in this section is over $\mathbb{F}_2$. Our task is to determine the parity of the number of solutions $x \in \{0,1\}^n$ to a given system of degree-d polynomial equations

$$P_1(x) = 0, \quad P_2(x) = 0, \quad \ldots, \quad P_m(x) = 0\,. \tag{6}$$

We present a randomized self-reduction that reduces (6) to multiple similar systems of degree at most d but over $\ell = \lambda n$ variables for a constant $0 < \lambda < 1$. Optimizing λ and applying the reduction recursively yields our main result.

3.1 Parity-counting as summation over the domain

We start with the elementary observation that determining the parity of the number of solutions to (6) amounts to computing the sum

$$I_F = \sum_{x\in\{0,1\}^n} F(x) \tag{7}$$

of the polynomial function

$$F(x) = (1 + P_1(x))(1 + P_2(x)) \cdots (1 + P_m(x))\,. \tag{8}$$

Indeed, for $x \in \{0,1\}^n$ we have $F(x) = 1$ if and only if x is a solution to (6), and otherwise $F(x) = 0$. For comparison, Lokshtanov et al. [9] used

$$J_F = \bigvee_{x \in \{0,1\}^{n-n'}} \left(\sum_{y \in \{0,1\}^{n'}} s_y \cdot F(x,y) \right) \tag{9}$$

with s_y independently and uniformly sampled scalars from $\{0,1\}$, with the observation that J_F is one with probability at least $1/2$ if the original system has a solution, and is always zero if the original system has no solutions. In the following, we show how to obtain a faster algorithm by dealing with I_F instead.

We do not know how to quickly evaluate I_F directly, but we will take an indirect and randomized approach to perform the summation.

3.2 Approximate the summand F by a low-degree probabilistic polynomial

The main difficulty in directly working with the polynomial F of (8) is its degree, which could be dm in general. As in Lokshtanov et al. [9], we construct a *probabilistic* polynomial $\tilde{F}$ with the property that for $0 < \epsilon \leq 1$ and for all $x \in \{0,1\}^n$ we have

$$\Pr_{f \in \tilde{F}} (F(x) = f(x)) \geq 1 - \epsilon \,. \tag{10}$$

We use the following construction generally credited to Razborov [16] and Smolensky [17]. For $i = 1, 2, \ldots, \lceil \log_2 \epsilon^{-1} \rceil$ and $j = 1, 2, \ldots, m$, draw an independent uniform random value $\rho_{ij} \in \{0,1\}$, and construct the polynomials

$$R_i(x) = \sum_{j=1}^{m} \rho_{ij} \cdot P_j(x) \,. \tag{11}$$

Let us now study the polynomial

$$f(x) = (1 + R_1(x))(1 + R_2(x)) \cdots (1 + R_{\lceil \log_2 \epsilon^{-1} \rceil}(x)) \,. \tag{12}$$

We easily observe that (10) holds. Furthermore, since each of the polynomials R_i has degree at most d, the degree of f is at most $d\lceil \log_2 \epsilon^{-1} \rceil$, rather than the degree $\Omega(dm)$ of F.

3.3 Sum the parts of multiple independent approximations

Suppose we replace the summands $F(x)$ in the computation of $I_F = \sum_x F(x)$ with $f(x)$'s from (12). By doing so we have reduced the degrees of the summands, but we also introduced a difficulty in the process: the summands $f(x)$ may introduce errors in the computation of I_F. We resolve this issue by drawing a sample of $s = O(n)$ independent Razborov–Smolensky approximations $f_1, f_2, \ldots, f_s \in \tilde{F}$, and then sum each of these, *in parts*.

Let us define precisely what we mean. Suppose our summand is g. Let us view g as the set function $g : 2^{[n]} \to \{0,1\}$ defined over the set of subsets of $[n]$. Let $A, B \subseteq [n]$ be disjoint with $A \cup B = [n]$. Think of A and B as a partition of n variables (indexed by $[n]$) into two parts; a subset $X \subseteq A$ will be construed as a 0-1 assignment to the variables in A, and a subset $Z \subseteq B$ will be construed as a 0-1 assignment to the variables in B. We will compute (7) as

$$I_F = \sum_{Z \in 2^B} \sum_{X \in 2^A} F(X \cup Z).$$

This is similar to Lokshtanov et al. [9] with $A = [n']$ in (9), except we take the actual sum modulo 2 over 2^B assignments, instead of a disjunction over the assignments.

For every $Z \subseteq B$ (construed as a 0-1 assignment to the variables in B), define the function

$$g|_A^{Z\to B} : 2^A \to \{0,1\}$$

for all $X \subseteq A$ by

$$g|_A^{Z\to B}(X) = g(X \cup Z)\,.$$

That is, $g|_A^{Z\to B}$ is the *part of* g obtained from fixing the variables in B to the 0-1 assignment g by Z. Each part of g is a polynomial over the variables in A.

Summing g in parts now amounts to computing the sums of all parts

$$I_{g|_A^{Z\to B}} = \sum_{X\subseteq A} g(X \cup Z) \quad \text{for each} \quad Z \subseteq B\,.$$

Once we have the sums of all parts, we can easily compute the overall sum:

$$I_g = \sum_{X\subseteq[n]} g(X) = \sum_{Z\subseteq B} I_{g|_A^{Z\to B}}\,.$$

However, we will *not* sum up the parts' sums directly. Indeed, we are summing potentially erroneous approximations of the true summands, and obtaining the (full) sum of a potentially erroneous approximation is not what we want. What we want, with high probability, is the sum of the true summands.

3.4 Correct the sum of each part by "scoreboarding"

Recall that we proposed to work with a sample of $s \le O(n)$ independent polynomials $f_1, f_2, \ldots, f_s \in \tilde{F}$ that approximate the true summand F. Suppose we have summed each approximation, in parts, to obtain the summand of each part of each approximation. That is, for each $Z \subseteq B$ we have the *scoreboard* of s sums

$$I_{f_1|_A^{Z\to B}} = \sum_{X\subseteq A} f_1(X\cup Z), \quad I_{f_2|_A^{Z\to B}} = \sum_{X\subseteq A} f_2(X\cup Z), \quad \ldots, \quad I_{f_s|_A^{Z\to B}} = \sum_{X\subseteq A} f_s(X\cup Z).$$

Each of these s sums is $\{0,1\}$-valued. Assuming s is odd, we take the unique majority value across the scoreboard and set, for every $Z \subseteq B$,

$$\tilde{I}_Z = \text{Majority}\big(I_{f_1|_A^{Z\to B}}, I_{f_2|_A^{Z\to B}}, \ldots, I_{f_s|_A^{Z\to B}}\big) = \begin{cases} 1 & \text{if } \sum_{j=1}^s I_{f_j|_A^{Z\to B}} > \frac{s}{2}\,, \\ 0 & \text{if } \sum_{j=1}^s I_{f_j|_A^{Z\to B}} < \frac{s}{2}\,. \end{cases} \tag{13}$$

Consider now the true summand F and a sum of its part $I_{F|_A^{Z\to B}}$. We can control the probability of error $\Pr[\tilde{I}_Z \neq I_{F|_A^{Z\to B}}]$ as follows. First, set $\epsilon = 2^{-(|A|+2)}$ and use the union bound with (10) to conclude that, for all $Z \subseteq B$ and $j = 1, 2, \ldots, s$, we have

$$\Pr\big[I_{F|_A^{Z\to B}} = I_{f_j|_A^{Z\to B}}\big] \ge 1 - 2^{|A|}\cdot\epsilon \ge \frac{3}{4}\,. \tag{14}$$

Consequently, the approximate summands f_j are bounded in degree by at most

$$\Delta = \lceil \log_2 \epsilon^{-1} \rceil d = (|A| + 2)d\,. \tag{15}$$

Second, recalling that $f_1, f_2, \ldots, f_s$ are independent, we can use a standard Chernoff bound to control the error from scoreboarding. When T is a sum of independent identically distributed random variables, for all $0 \leq \delta \leq 1$ it holds that [12]

$$\Pr\big[T \leq (1-\delta)\mathbb{E}[T]\big] \leq \exp\big(-\tfrac{\delta^2 \mathbb{E}[T]}{2}\big)\,. \tag{16}$$

Take $T_Z = \sum_{j=1}^{s} I_{f_j|_A^{Z\to B}}$. From (14) we have

$$\begin{aligned} \mathbb{E}[T_Z \,|\, I_{F|_A^{Z\to B}} = 1] &\geq \tfrac{3}{4}s\,, \\ \mathbb{E}[s - T_Z \,|\, I_{F|_A^{Z\to B}} = 0] &\geq \tfrac{3}{4}s\,, \end{aligned} \tag{17}$$

Recalling that we assume s to be odd, from (16) and (17) with $\delta = 1/3$ we thus have

$$\begin{aligned} \Pr\big[T_Z > \tfrac{s}{2} \,\big|\, I_{F|_A^{Z\to B}} = 1\big] &\geq 1 - \exp\big(-\tfrac{s}{24}\big)\,, \\ \Pr\big[s - T_Z > \tfrac{s}{2} \,\big|\, I_{F|_A^{Z\to B}} = 0\big] &\geq 1 - \exp\big(-\tfrac{s}{24}\big)\,. \end{aligned} \tag{18}$$

Set $s = 48n + 1$ and use (18) in (13) to conclude that

$$\Pr\big[\tilde{I}_Z = I_{F|_A^{Z\to B}}\big] \geq 1 - 2^{-2n}\,.$$

Taking a union bound over all $Z \subseteq B$, we have

$$\Pr\big[I_F = \textstyle\sum_{Z\subseteq B} \tilde{I}_Z\big] \geq 1 - 2^{-n}\,.$$

That is, error-correction by scoreboarding enables us to recover the sum $I_F = \sum_x F(x)$ with only exponentially small error probability. All that now remains is to sum in parts.

3.5 Summing the parts of a low-degree polynomial

As before, we view the summand as a set function $f : 2^{[n]} \to \{0,1\}$ and assume that the underlying polynomial representing f has degree at most Δ. For each $Z \subseteq B$, we want to produce the sum $I_{f|_A^{Z\to B}} = \sum_{X\subseteq A} f(X \cup Z)$.

Our strategy for summation will rely on the fact that these sums are linearly dependent, and fast basis changes via fast zeta transforms will enable fast summation of parts.

To witness the linear dependence, let us put to use the fact that f is low-degree. Let $M_f : 2^{[n]} \to \{0,1\}$ be the representation of f in the monomial basis (M_f maps monomials to coefficients). For all $W \subseteq [n]$, we thus have

$$f(W) = \sum_{U\subseteq W} M_f(U)\,.$$

Now recall that f has degree at most Δ if and only if M_f vanishes on subsets of size greater than Δ. That is, the representation in the monomial basis is sparse; we seek to express our sums in this basis.

Toward this end, let us study summing a part from the perspective of the coefficients M_f rather than f. For each $Z \subseteq B$, we have

$$I_{f|_A^{Z\to B}} = \sum_{X\subseteq A} f(X \cup Z) = \sum_{X\subseteq A} \sum_{U\subseteq X\cup Z} M_f(U) = \sum_{Y\subseteq Z} M_f(A \cup Y)\,. \tag{19}$$

The last equality in (19) follows because *every monomial (viewed as a subset) not containing A will cancel modulo two*, because it contributes to the sum an even number of times. This

property of contributing monomials being known to contain A is the key difference from the approach of Lokshtanov et al. [9]; the property is lost if we take a disjunction (as in (9)) instead of a sum modulo 2 (as in (7)). It will yield a smaller upper bound on the number of monomials.

Let us now get some corollaries of (19). First, since only monomials that contain A contribute to the sums $I_{f|_A^{Z\to B}}$, knowledge of only these monomials is sufficient information to compute the sum $I_{f|_A^{Z\to B}}$ for each $Z \subseteq B$. Second, we know that M_f vanishes on all subsets of size greater than Δ. Since each monomial must contain A, we only have to consider subsets from B of size at most $\delta = \Delta - |A|$, compared to $\delta = \Delta$ used in Lokshtanov et al. [9].

These two corollaries yield the following three-step strategy for summing the parts:

(i) Compute $I_{f|_A^{B\to Z}} = \sum_{X\subseteq A} f(X \cup Z)$ for each set $Z \in \binom{B}{\downarrow\delta}$.
By (19), we thus have $\binom{|B|}{\downarrow\delta}$ equations for the $\binom{|B|}{\downarrow\delta}$ unknowns $M_f(A \cup Y)$, for $Y \in \binom{B}{\downarrow\delta}$.

(ii) Solve the equations for the values $M_f(A \cup Y)$ for $Y \in \binom{B}{\downarrow\delta}$ (the coefficients of f as a polynomial).

(iii) Use the solved values to produce the sum $I_{f|_A^{B\to Z}}$ for each $Z \subseteq B$.

Observe that the only actual summations are made in (i). The step (iii) produces the sums in batch from the values obtained in (ii). To solve the equations in (ii), we use the fast zeta transform over the sets in $\binom{B}{\downarrow\delta}$. For the step (iii), use the fast zeta transform over 2^B with the knowledge that $M_f(A \cup Y)$ vanishes in 2^B outside $Y \in \binom{B}{\downarrow\delta}$.

3.6 Summing a part reduces back to parity-counting

Let us now define in detail what it means to sum a part in step (i). First, let us parameterise the partition A, B of $[n]$. For $1 \le \ell \le n$, set

$$|A| = \ell \qquad \text{and} \qquad |B| = n - |A| = n - \ell\,.$$

By (15), we have

$$\lceil \log_2 \epsilon^{-1} \rceil = |A| + 2 = \ell + 2.$$

Thus our polynomials have degree $\Delta \le (\ell + 2)d$, and

$$\delta = \Delta - |A| = (|A| + 2)d - |A| = (d - 1)\ell + 2d\,.$$

Recall that the given input consists of the polynomials $P_1, P_2, \ldots, P_m$ in (6). Using the polynomials $P_1, P_2, \ldots, P_m$, the algorithm draws s samples, where each sample is an independent collection of Razborov–Smolensky polynomials $R_1, R_2, \ldots, R_{\ell+2}$ constructed using (11). (Recall the R_i's are simply random linear combinations of the P_j's.) Each collection forms one of the approximate summands $f_j = \prod_i (1 + R_i)$ of (12). However, in our algorithm these approximate summands f_j are never constructed in explicit form: in Lokshtanov et al. [9] they are constructed explicitly, which leads to a worse running time.

Rather, we observe that we can access a part $f_j|_A^{Z\to B}$ for $Z \in \binom{B}{\downarrow\delta}$ by making the substitution $Z \to B$ directly into the Razborov–Smolensky polynomials $R_1, R_2, \ldots, R_{\ell+2}$ that define f_j, without constructing f_j itself. In particular, after the substitution, the polynomials have variables $x_A = (x_j : j \in A)$. In notation, we construct by the substitution $Z \to B$ the polynomials

$$Q_1(x_A) = R_1|_A^{Z\to B}(x_A)\,, \quad Q_2(x_A) = R_2|_A^{Z\to B}(x_A)\,, \quad \ldots, \quad Q_{\ell+2}(x_A) = R_{\ell+2}|_A^{Z\to B}(x_A)\,.$$

Since the R_i's are just linear combinations of polynomials of degree at most d, these polynomials also have degree at most d, and are over $\ell = |A|$ variables. Thus, computing the sum $I_{f|_A^{Z\to B}}$ of the part $f|_A^{Z\to B}$ is exactly the task of summing over $x_A \in \{0,1\}^{|A|}$ the polynomial

$$f|_A^{Z\to B}(x_A) = (1 + Q_1(x_A))(1 + Q_2(x_A)) \cdots (1 + Q_{\ell+2}(x_A)) \,.$$

Recalling (8), this is exactly the task of determining the parity of the number of solutions $x_A \in \{0,1\}^{|A|}$ to the system of polynomial equations

$$Q_1(x_A) = 0 \,, \quad Q_2(x_A) = 0 \,, \quad \ldots, \quad Q_{\ell+2}(x_A) = 0 \,.$$

This completes our randomized reduction from parity-counting to itself: we have reduced parity counting for a system of degree-d polynomials in n variables to $\binom{|B|}{\downarrow\delta} = \binom{n-\ell}{\downarrow(d-1)\ell+2d}$ calls to parity counting a system of degree-d polynomials in ℓ variables.

To compare, in Lokshtanov et al. [9], such a self-reduction seems not to be possible when one uses (9) instead of (7). A full $O^*(2^{|A|})$ time summation was used in their paper.

3.7 Running time analysis

Let us now analyze the running time as a function of the number of variables n and the reduction parameter ℓ with $1 \le \ell \le n$. Let us write $T(n,m)$ for an upper bound for the worst-case running time when the input consists of at most m polynomials of degree at most d in at most n variables. Similarly, let us write $S(n,m)$ for an upper bound for the worst-case space complexity.

Let us now recall the structure of the self-reduction, then analyze its recursive application. The reduction first builds the $s = 48n + 1$ approximate summands (via the constituent polynomials $R_1, R_2, \ldots, R_{\ell+2}$) and then works to complete $2^{|B|} = 2^{n-\ell}$ scoreboards, each recording the sum (over the integers to enable majority-voting) of summation of s parts. The summation of parts proceeds across the scoreboards, one entire approximate summand at a time using steps (i), (ii), and (iii). In step (i), we recursively perform summations for $\binom{|B|}{\downarrow\delta} = \binom{n-\ell}{\downarrow(d-1)\ell+2d}$ parts, each via the constituent polynomials $Q_1, Q_2, \ldots, Q_{\ell+2}$ of degree at most d over ℓ variables. In step (ii), we run the fast zeta transform over $\binom{B}{\downarrow\delta}$ to recover the monomials of the summand for all $A \cup Y$ with $Y \in \binom{B}{\downarrow\delta}$. In step (iii), we run the fast zeta transform over 2^B to recover all sums of parts for one approximate summand. This procedure is repeated for each of the approximate summands, updating the scoreboard as we go. Once the scoreboards are complete, the algorithm takes the majority vote in each scoreboard, and returns the parity of the majority votes.

The space complexity of the reduction can be upper-bounded via the recursive scoreboards and the representation of the polynomials in the monomial basis, with

$$S(n,m) \le S(\ell, \ell+2) + O\big(2^{n-\ell} \log s + m\tbinom{n}{\downarrow d}\big) \,. \tag{20}$$

Indeed, the zeta transforms at each level of recursion require only space $O(2^{n-\ell})$.

The time complexity of the reduction can be upper-bounded via the brute-force base case $T(n,m) \le O(2^n nm\binom{n}{\downarrow d})$ and the recurrence

$$T(n,m) \le s\tbinom{n-\ell}{\downarrow(d-1)\ell+2d} T(\ell,\ell+2) + O\big(s\big(\tbinom{n-\ell}{\downarrow(d-1)\ell+2d}\big)\big(n+(\ell+2)m\tbinom{n}{\downarrow d}\big)+2^{n-\ell}(n-\ell)\big)\big) \,. \tag{21}$$

The first term accounts for the parity self-reduction; the second term accounts for the fast zeta transforms.

Let us now assume that $d = 2, 3, \ldots$ is a fixed constant. We will run the reduction (21) recursively for $D = D(d)$ levels, and then switch to the brute-force base case. The parameter ℓ at each level is set by means of a constant $\lambda = \lambda(d)$ with $0 < \lambda < \frac{1}{2d-1}$ so that $\ell = \lfloor \lambda n \rfloor$ where n is the number of variables in the input to the level. Let

$$H(\rho) = -\rho \log_2 \rho - (1-\rho)\log_2(1-\rho)$$

be the binary entropy function and recall that we have $\binom{k}{\lfloor u \rfloor} \leq 2^{kH(u/k)}$ for all $1 \leq u \leq \frac{k}{2}$.

Since $\ell \leq \frac{n}{2d-1}$, the sums of binomial coefficients in (21) can be upper-bounded as follows:

$$\tbinom{n-\ell}{\downarrow(d-1)\ell+2d} \leq n^{2d+1}\tbinom{n-\ell}{(d-1)\ell} \leq n^{2d+2}2^{n(1-\lambda)H\left(\frac{(d-1)\lambda}{1-\lambda}\right)}.$$

Assuming that we run the recursion for $D = D(d)$ levels and then use brute force, we observe that there exists a constant $C = C(d) > 0$ such that we have

$$T(n,m) = O(mn^C(1 + 2^{\tau(1)n}))\,,$$

where $\tau(\lambda^k)$ is a parameter defined for $k = 0, 1, \ldots, D-1$ by

$$\tau(\lambda^k) = \lambda^k \max\bigl((1-\lambda)H\bigl(\tfrac{(d-1)\lambda}{1-\lambda}\bigr) + \tau(\lambda^{k+1}),\, 1-\lambda\bigr) \tag{22}$$

and

$$\tau(\lambda^D) = \lambda^D\,. \tag{23}$$

Recalling that $1 + \lambda + \lambda^2 + \ldots = \frac{1}{1-\lambda}$, and choosing a large enough D, we have that

$$\tau(1) \leq 1-\lambda \quad \text{whenever} \quad H\bigl(\tfrac{(d-1)\lambda}{1-\lambda}\bigr) < 1-\lambda\,.$$

Thus, for any $d \geq 2$ we can select $\lambda = 1/(2.7d)$ to obtain the running time $O^*\bigl(2^{(1-1/(2.7d))n}\bigr)$. For $d = 2$, we can select $\lambda = 0.196774680497$ to obtain the running time $O^*\bigl(2^{0.803225n}\bigr)$.

References

1. Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Pierre-Jean Spaenlehauer. On the complexity of solving quadratic Boolean systems. *J. Complexity*, 29(1):53–75, 2013. doi:10.1016/j.jco.2012.07.001.
2. Andreas Björklund, Thore Husfeldt, Petteri Kaski, Mikko Koivisto, Jesper Nederlof, and Pekka Parviainen. Fast zeta transforms for lattices with few irreducibles. *ACM Trans. Algorithms*, 12(1):Art. 4, 19, 2016.
3. Bruno Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenrings nach einem nulldimensionalen Polynomideal.* PhD thesis, Department of Mathematics, University of Innsbruck, 1965.
4. David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms.* Undergraduate Texts in Mathematics. Springer, Cham, fourth edition, 2015. doi:10.1007/978-3-319-16721-3.
5. Aviezri S. Fraenkel and Yaacov Yesha. Complexity of problems in games, graphs and algebraic equations. *Discrete Applied Mathematics*, 1(1-2):15–30, 1979. doi:10.1016/0166-218X(79)90012-X.
6. Antoine Joux and Vanessa Vitse. A crossbred algorithm for solving Boolean polynomial systems. Cryptology ePrint Archive, Report 2017/372, 2017. URL: https://eprint.iacr.org/2017/372.
7. Petteri Kaski, Jukka Kohonen, and Thomas Westerbäck. Fast Möbius inversion in semimodular lattices and ER-labelable posets. *Electron. J. Combin.*, 23(3):Paper 3.26, 13, 2016.

8 Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced Oil and Vinegar Signature Schemes. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 206–222. Springer, 1999. doi:10.1007/3-540-48910-X_15.

9 Daniel Lokshtanov, Ramamohan Paturi, Suguru Tamaki, R. Ryan Williams, and Huacheng Yu. Beating Brute Force for Systems of Polynomial Equations over Finite Fields. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2190–2202. SIAM, 2017. doi:10.1137/1.9781611974782.143.

10 Yuri V. Matiyasevich. *Hilbert's Tenth Problem*. Foundations of Computing Series. MIT Press, Cambridge, MA, 1993.

11 Ernst W. Mayr. Some complexity results for polynomial ideals. *J. Complexity*, 13(3):303–325, 1997. doi:10.1006/jcom.1997.0447.

12 Michael Mitzenmacher and Eli Upfal. *Probability and Computing*. Cambridge University Press, Cambridge, second edition, 2017.

13 Robin A. Moser and Dominik Scheder. A full derandomization of Schöning's k-SAT algorithm. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 245–252. ACM, 2011. doi:10.1145/1993636.1993670.

14 Ruben Niederhagen, Kai-Chun Ning, and Bo-Yin Yang. Implementing Joux-Vitse's Crossbred Algorithm for Solving MQ Systems over GF(2) on GPUs. Cryptology ePrint Archive, Report 2017/1181, 2017. URL: https://eprint.iacr.org/2017/1181.

15 Jacques Patarin. Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 1996. doi:10.1007/3-540-68339-9_4.

16 A. A. Razborov. Lower bounds on the dimension of schemes of bounded depth in a complete basis containing the logical addition function. *Mat. Zametki*, 41(4):598–607, 623, 1987.

17 Roman Smolensky. Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 77–82. ACM, 1987. doi:10.1145/28395.28404.

18 Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theor. Comput. Sci.*, 47(3):85–93, 1986. doi:10.1016/0304-3975(86)90135-0.

19 F. Yates. *The Design and Analysis of Factorial Experiments*. Imperial Bureau of Soil Science, Harpenden, 1937.

Quantum SDP Solvers: Large Speed-Ups, Optimality, and Applications to Quantum Learning

Fernando G. S. L. Brandão
Institute of Quantum Information and Matter, California Institute of Technology, USA
fgslbrandao@gmail.com

Amir Kalev
Joint Center for Quantum Information and Computer Science, University of Maryland, USA
amirk@umd.edu

Tongyang Li
Joint Center for Quantum Information and Computer Science, University of Maryland, USA
tongyang@cs.umd.edu

Cedric Yen-Yu Lin
Joint Center for Quantum Information and Computer Science, University of Maryland, USA
cedriclin37@gmail.com

Krysta M. Svore
Station Q, Quantum Architectures and Computation Group, Microsoft Research, USA
ksvore@microsoft.com

Xiaodi Wu
Joint Center for Quantum Information and Computer Science, University of Maryland, USA
xwu@cs.umd.edu

Abstract

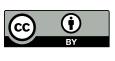
We give two new quantum algorithms for solving semidefinite programs (SDPs) providing quantum speed-ups. We consider SDP instances with m constraint matrices, each of dimension n, rank at most r, and sparsity s. The first algorithm assumes an input model where one is given access to an oracle to the entries of the matrices at unit cost. We show that it has run time $\tilde{O}(s^2(\sqrt{m}\epsilon^{-10} + \sqrt{n}\epsilon^{-12}))$, with ϵ the error of the solution. This gives an optimal dependence in terms of m, n and quadratic improvement over previous quantum algorithms (when $m \approx n$). The second algorithm assumes a fully quantum input model in which the input matrices are given as quantum states. We show that its run time is $\tilde{O}(\sqrt{m} + \mathrm{poly}(r)) \cdot \mathrm{poly}(\log m, \log n, B, \epsilon^{-1})$, with B an upper bound on the trace-norm of all input matrices. In particular the complexity depends only polylogarithmically in n and polynomially in r.

We apply the second SDP solver to learn a good description of a quantum state with respect to a set of measurements: Given m measurements and a supply of copies of an unknown state ρ with rank at most r, we show we can find in time $\sqrt{m} \cdot \mathrm{poly}(\log m, \log n, r, \epsilon^{-1})$ a description of the state as a quantum circuit preparing a density matrix which has the same expectation values as ρ on the m measurements, up to error ϵ. The density matrix obtained is an approximation to the maximum entropy state consistent with the measurement data considered in Jaynes' principle from statistical mechanics.

As in previous work, we obtain our algorithm by "quantizing" classical SDP solvers based on the matrix multiplicative weight update method. One of our main technical contributions is a quantum Gibbs state sampler for low-rank Hamiltonians, given quantum states encoding these Hamiltonians, with a poly-logarithmic dependence on its dimension, which is based on ideas developed in quantum principal component analysis. We also develop a "fast" quantum OR lemma with a quadratic improvement in gate complexity over the construction of Harrow et al. [14]. We believe both techniques might be of independent interest.

2012 ACM Subject Classification Theory of computation → Semidefinite programming; Theory of computation → Quantum query complexity; Theory of computation → Convex optimization

Keywords and phrases quantum algorithms, semidefinite program, convex optimization



46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 27; pp. 27:1–27:14

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

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.27

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1710.02581`.

Acknowledgements We thank Scott Aaronson, Joran van Apeldoorn, András Gilyén, Cupjin Huang, and anonymous reviewers for helpful discussions. We are also grateful to Joran van Apeldoorn and András Gilyén for sharing a working draft of [4] with us. FB was supported by NSF. CYL and AK were supported by the Department of Defense. TL was supported by NSF CCF-1526380. XW was supported by NSF grants CCF-1755800 and CCF-1816695, and also by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Quantum Algorithms Teams program.

1 Introduction

Motivation. Semidefinite programming has been a central topic in the study of mathematical optimization, theoretical computer science, and operations research in the last decades. It has become an important tool for designing efficient optimization and approximation algorithms. The power of semidefinite programs (SDPs) lies in their generality (that extends the better-known linear programs (LPs)) and the fact that they admit polynomial-time solvers.

It is natural to ask whether quantum computers can have advantage in solving this important optimization problem. In Ref. [9], Brandão and Svore provided an affirmative answer, giving a quantum algorithm with worst-case running time $\tilde{O}(\sqrt{mn}s^2(R\tilde{R}/\varepsilon)^{32})$ [1], where n and s are the dimension and row sparsity of the input matrices, respectively, m the number of constraints, ϵ the accuracy of the solution, and $R, \tilde{R}$ upper bounds on the norm of the optimal primal and dual solutions. This is a *polynomial* speed-up in m and n comparing to the two state-of-the-art classical SDP-solvers [20, 7] (with complexity $\tilde{O}(m(m^2 + n^\omega + mns)\operatorname{poly}\log(R/\epsilon))$ [20], where ω is the exponent of matrix multiplication, and $\tilde{O}(mns(R\tilde{R}/\varepsilon)^4 + ns\left(R\tilde{R}/\varepsilon\right)^7)$ [7]), and beating the classical lower bound of $\Omega(m+n)$ [9]. The follow-up work by van Apeldoorn et al. [5] improved the running time giving a quantum SDP solver with complexity $\tilde{O}(\sqrt{mn}s^2(R\tilde{R}/\epsilon)^8)$. In terms of limitations, Ref. [9] proved a quantum lower bound $\Omega(\sqrt{m}+\sqrt{n})$ when $R, \tilde{R}, s, \epsilon$ are constants; stronger lower bounds can be proven if R and/or $\tilde{R}$ scale with m and n [5]. We note all these results are shown in an input model in which there is an oracle for the entry of each of the input matrices (see Oracle 1.1 below for a formal definition).

In this paper, we investigate quantum algorithms for SDPs (i.e., quantum SDP solvers) further in the following two perspectives: (1) the best dependence of parameters, especially the dimension n and the number of constraints m; (2) whether there is any reasonable alternative input model for quantum SDP solvers and what is its associated complexity. To that end, let us first formulate the precise SDP instance in our discussion.

The SDP approximate feasibility problem. We will work with the SDP approximate feasibility problem formulated as follows: Given an $\epsilon > 0$, m real numbers $a_1, \ldots, a_m \in \mathbb{R}$, and Hermitian $n \times n$ matrices $A_1, \ldots, A_m$ where $-I \preceq A_i \preceq I, \forall\, j \in [m]$, define the convex

[1] $\tilde{O}$ hides factors that are polynomial in $\log m$ and $\log n$.

region $\mathcal{S}_\epsilon$ as all X such that

$$\begin{aligned} \mathrm{Tr}(A_i X) &\leq a_i + \epsilon \quad \forall\, i \in [m]; \\ X &\succeq 0;\ \mathrm{Tr}[X] = 1. \end{aligned} \tag{1.1}$$

For approximate feasibility testing, it is required that either (1) If $\mathcal{S}_0 = \emptyset$, output fail; or (2) If $\mathcal{S}_\epsilon \neq \emptyset$, output an $X \in \mathcal{S}_\epsilon$. Throughout the paper, we denote by n the the dimension of the matrices, m the number of constraints, and ϵ the (additive) error of the solution. For Hermitian matrices A and B, we denote $A \preceq B$ if $B - A$ is positive semidefinite, and $A \succeq B$ if $A - B$ is positive semidefinite. We denote I_n to be the $n \times n$ identity matrix.

There are a few reasons that guarantee our choice of approximate SDP feasibility problem do not lose generality: (1) first, it is a routine[2] to reduce general optimization SDP problems to the feasibility problem; (2) second, for general feasible solution $X \succeq 0$ with width bound $\mathrm{Tr}(X) \leq R$, there is a procedure[3] to derive an equivalent SDP feasibility instance with variable $\hat{X}$ s.t. $\mathrm{Tr}(\hat{X}) = 1$. Note, however, the change of ϵ to ϵ/R in this conversion. Also note one can use an approximate feasibility solver to find a strictly feasible solution, by changing ϵ to $\epsilon/R\tilde{R}$ (see Lemma 18 of Ref. [9]). The benefit of our choice of (1.1) is its simplicity in presentation, which provides a better intuition behind our techniques and an easy adoption of our SDP solver in learning quantum states. In contrast to Ref. [5], we do not need to formulate the dual program of Eq. (1.1) since our techniques do not rely on it. We will elaborate more on these points in Section 1.4.

1.1 Quantum SDP solvers with optimal dependence on m and n

Existing quantum SDP solvers [9, 5] have close-to-optimal dependence on some key parameters but poor dependence on others. Seeking optimal parameter dependence has been an important problem in the development of classical SDP solvers and has inspired many new techniques. It is thus well motivated to investigate the optimal parameter dependence in the quantum setting. Our first contribution is the construction of a quantum SDP solver with the optimal dependence on m and n in the (plain) input model as used by [9, 5], given as follows:

▶ **Oracle 1.1** (Plain model for A_j)**.** *A quantum oracle, denoted $\mathcal{P}_A$, such that given the indices $j \in [m]$, $k \in [n]$ and $l \in [s]$, computes a bit string representation of the l-th non-zero element of the k-th row of A_j, i.e. the oracle performs the following map:*

$$|j, k, l, z\rangle \to |j, k, l, z \oplus (A_j)_{k f_{jk}(l)}\rangle, \tag{1.2}$$

with $f_{jk} : [r] \to [N]$ a function (parametrized by the matrix index j and the row index k) which given $l \in [s]$ computes the column index of the l-th nonzero entry.

Before we move on to our main result, we will define two primitives which will appear in our quantum SDP solvers. Our main result will also be written in terms of the cost for each primitive.

---

[2] To see why this is the case, for any general SDP problem, one can guess a candidate value (e.g., c_0) for the objective function (e.g., $\mathrm{Tr}(CX)$ and assume one wants to maximize $\mathrm{Tr}(CX)$) and convert it into a constraint (e.g., $\mathrm{Tr}(CX) \geq c_0$). Hence one ends up with a feasibility problem and the candidate value c_0 can then be found via binary search with $O(\log(1/\epsilon))$ overhead when $\mathrm{Tr}(CX) \in [-1, 1]$.

[3] The procedure goes as follows: (a) scale down every constraint by a factor R and let $X' = X/R$ (thus $\mathrm{Tr}(X') \leq 1$) (b) let $\hat{X} = \mathrm{diag}\{X, w\}$ be a block-diagonal matrix with X in the upper-left corner and a scaler w in the bottom-right corner. It is easy to see that $\mathrm{Tr}(\hat{X}) = 1 \iff \mathrm{Tr}(X) \leq 1$.

▶ **Definition 1** (trace estimation). *Assume that we have an s-sparse $n \times n$ Hermitian matrix H with $\|H\| \leq \Gamma$ and a density matrix ρ. Then we define $\mathcal{S}_{\mathrm{Tr}}(s, \Gamma, \epsilon)$ and $\mathcal{T}_{\mathrm{Tr}}(s, \Gamma, \epsilon)$ as the number of copies of ρ and the time complexity (in terms of oracle call and number of gates) of using the plain model (Oracle 1.1) for H, respectively, such that one can compute $\mathrm{Tr}[H\rho]$ with additive error ϵ with success probability at least $2/3$.*

▶ **Definition 2** (Gibbs sampling). *Assume that we have an s-sparse $n \times n$ Hermitian matrix H with $\|H\| \leq \Gamma$. Then we define $\mathcal{T}_{Gibbs}(s, \Gamma, \epsilon)$ as the complexity of preparing the Gibbs state $\frac{e^{-H}}{\mathrm{Tr}[e^{-H}]}$ with additive error ϵ using the plain model (Oracle 1.1) for H.*

Our main result is as follows.

▶ **Theorem 3.** *In the plain input model (Oracle 1.1), for any $0 < \epsilon < 1$, there is a quantum SDP solver for the feasibility problem (1.1) using $\frac{s}{\epsilon^4}\tilde{O}\big(\mathcal{S}_{\mathrm{Tr}}\big(\frac{s}{\epsilon^2}, \frac{1}{\epsilon}, \epsilon\big)\mathcal{T}_{Gibbs}\big(\frac{s}{\epsilon^2}, \frac{1}{\epsilon}, \epsilon\big) + \sqrt{m}\mathcal{T}_{\mathrm{Tr}}\big(\frac{s}{\epsilon^2}, \frac{1}{\epsilon}, \epsilon\big)\big)$ quantum gates and queries to Oracle 1.1, where s is the sparsity of $A_j, j \in [m]$.*

When combined with specific instantiation of these primitives (i.e., in our case, we directly make use of results on $\mathcal{S}_{\mathrm{Tr}}(s, \Gamma, \epsilon)$ and $\mathcal{T}_{\mathrm{Tr}}(s, \Gamma, \epsilon)$ from Ref. [9], and results on $\mathcal{T}_{\mathrm{Gibbs}}(s, \Gamma, \epsilon)$ from Ref. [24]), we end up with the following concrete parameters:

▶ **Corollary 4.** *In the plain input model (Oracle 1.1), for any $0 < \epsilon < 1$, there is a quantum SDP solver for the feasibility problem (1.1) using $\tilde{O}(s^2(\frac{\sqrt{m}}{\epsilon^{10}} + \frac{\sqrt{n}}{\epsilon^{12}}))$ quantum gates and queries to Oracle 1.1, where s is the sparsity of $A_j, j \in [m]$.*

Comparing to prior art, our main contribution is to decouple the dependence on m and n, which used to be $O(\sqrt{mn})$ and now becomes $O(\sqrt{m} + \sqrt{n})$. Note that the $(\sqrt{m} + \sqrt{n})$ dependence is optimal due to the quantum lower bound proven in Ref. [9].

▶ Remark 1.5. Even though our result achieves the optimal dependence on m and n, it is nontrivial to obtain quantum speed-ups by directly applying our quantum SDP solvers to SDP instances from classical combinatorial problems. The major obstacle is the poly-dependence on $1/\epsilon$, whereas, for interesting SDP instances such as Max-Cut, $1/\epsilon$ is linear in n. In fact, the general framework of the classical Arora-Kale SDP solver also suffers from the poly-dependence on $1/\epsilon$ and cannot be applied directly either. Instead, one needs to specialize the design of SDP solvers for each instance to achieve better time complexity.

Extending this idea to quantum seems challenging. One difficulty is that known classical approaches require explicit information of intermediate states, which requires $\Omega(n)$ time and space even to store. It is not clear how one can directly adapt classical approaches on intermediate states when stored as amplitudes in quantum states, which is the case for our current SDP solvers. It seems to us that a resolution of the problem might require an independent tool beyond the scope of this paper. We view this as an important direction for future work.

However, our quantum SDP solvers are sufficient for instances with mild $1/\epsilon$, which are natural in the context of quantum information, such as learnability of the quantum state problem (elaborated in Section 1.5) as well as examples in [4]. For those cases, we do establish a quantum speed-up as any classical algorithm needs at least linear time in n and/or m.

1.2 Quantum SDP solvers with quantum inputs

Given the optimality of the algorithm presented before (in terms of m and n), a natural question is to ask about the existence of alternative input models, *which can be justified for specific applications, and at the same time allows more efficient quantum SDP solvers.* This

is certainly a challenging question, but we can get inspiration from the application of SDPs in quantum complexity theory (e.g., Refs. [16, 12]) and quantum information (e.g., Refs. [1, 2]). In these settings, input matrices of SDP instances, with dimension 2^ℓ, are typically quantum states and/or measurements generated by poly(ℓ)-size circuits on ℓ qubits. For the sake of these applications, it might be reasonable to equip quantum SDP solvers with the ability to leverage these circuit information, rather than merely allowing access to the entries of the input matrices.

In this paper, we propose a *truly* quantum input model in which we can construct quantum SDP solvers with running time only *poly-logarithmic* in the dimension. We note that such proposal was mentioned in an earlier version of Ref. [9], whose precise mathematical form and construction of quantum SDP solvers were unfortunately incorrect, and later removed. Note that since we consider a non-standard input model in this section, our results are incomparable to those in the plain input model. We argue for the relevance of our quantum input model, by considering an applications of the framework to the problem of learning quantum states in Section 1.5.

Quantum input model. Consider a specific setting in which we are given decompositions of each A_j: $A_j = A_j^+ - A_j^-$, where $A_j^+, A_j^- \succeq 0$. (For instance, a natural choice is to let A_j^+ (resp. A_j^-) be the positive (resp. negative) part of A.)

▶ **Oracle 1.2** (Oracle for traces of A_j)**.** *A quantum oracle (unitary), denoted O_{Tr} (and its inverse $O_{\mathrm{Tr}}^\dagger$), such that for any $j \in [m]$,*

$$O_{\mathrm{Tr}}|j\rangle|0\rangle|0\rangle = |j\rangle|\operatorname{Tr}[A_j^+]\rangle|\operatorname{Tr}[A_j^-]\rangle, \tag{1.3}$$

where the real values $\operatorname{Tr}[A_j^+]$ and $\operatorname{Tr}[A_j^-]$ are encoded into their binary representations.

▶ **Oracle 1.3** (Oracle for preparing A_j)**.** *A quantum oracle (unitary), denoted O (and its inverse $O^\dagger$), which acts on $\mathbb{C}^m \otimes (\mathbb{C}^n \otimes \mathbb{C}^n) \otimes (\mathbb{C}^n \otimes \mathbb{C}^n)$ such that for any $j \in [m]$,*

$$O|j\rangle|0\rangle|0\rangle = |j\rangle|\psi_j^+\rangle|\psi_j^-\rangle, \tag{1.4}$$

where $|\psi_j^+\rangle, |\psi_j^-\rangle \in \mathbb{C}^n \otimes \mathbb{C}^n$ are any purifications of $\frac{A_j^+}{\operatorname{Tr}[A_j^+]}, \frac{A_j^-}{\operatorname{Tr}[A_j^-]}$, respectively.

▶ **Oracle 1.4** (Oracle for a_j)**.** *A quantum oracle (unitary), denoted O_a (and its inverse $O_a^\dagger$), such that for any $j \in [m]$,*

$$O_a|j\rangle|0\rangle = |j\rangle|a_j\rangle, \tag{1.5}$$

where the real value a_j is encoded into its binary representation.

Throughout the paper, let us assume that A_j has rank at most r for all $j \in [m]$ and $\operatorname{Tr}[A_j^+] + \operatorname{Tr}[A_j^-] \leq B$. The parameter B is therefore an upper bound to the trace-norm of all input matrices which we assume is given as an input of the problem. Similar to the plain input model, we will define the same two primitives and their associated costs in the quantum input model.

▶ **Definition 6** (trace estimation)**.** *We define $\mathcal{S}_{\mathrm{Tr}}(B, \epsilon)$ and $\mathcal{T}_{\mathrm{Tr}}(B, \epsilon)$ as the sample complexity of a state $\rho \in \mathbb{C}^{n\times n}$ and the gate complexity of using the quantum input oracles (Oracle 1.2, Oracle 1.3, Oracle 1.4), respectively, for the fastest quantum algorithm that distinguishes with success probability at least $1 - O(1/m)$ whether for a fixed $j \in [m]$, $\operatorname{Tr}(A_j\rho) > a_j + \epsilon$ or $\operatorname{Tr}(A_j\rho) \leq a_j$.*

▶ **Definition 7** (Gibbs sampling). *Assume that $K = K^+ - K^-$, where $K^\pm = \sum_{j\in S} c_j A_j^\pm$, $c_j > 0$, $S \subseteq [m]$ and $|S| \leq \Phi$, and that K^+, K^- have rank at most r_K. Moreover, assume that $\mathrm{Tr}(K^+) + \mathrm{Tr}(K^-) \leq B_K$ for some B_K. Then we define $\mathcal{T}_{Gibbs}(r_K, \Phi, B_K, \epsilon)$ as the gate complexity of preparing the Gibbs state $\rho_G = \exp(-K)/\mathrm{Tr}(\exp(-K))$ to ϵ precision in trace distance using Oracle 1.2, Oracle 1.3, and Oracle 1.4.*

Our main result in the quantum input model is as follows.

▶ **Theorem 8.** *For any $\epsilon > 0$, there is a quantum algorithm for the approximate feasibility of the SDP using at most $\frac{1}{\epsilon^2}\tilde{O}\big(\mathcal{S}_{\mathrm{Tr}}(B,\epsilon)\mathcal{T}_{Gibbs}\big(\frac{r}{\epsilon^2}, \frac{1}{\epsilon^2}, \frac{B}{\epsilon}, \epsilon\big) + \sqrt{m}\mathcal{T}_{\mathrm{Tr}}(B,\epsilon)\big)$ quantum gates and queries to Oracle 1.2, Oracle 1.3, and Oracle 1.4.*

In contrast to the plain model setting, the quantum input model is a completely new setting so that we have to construct these two primitive by ourselves. In particular, we give a construction of trace estimation with $\mathcal{S}_{\mathrm{Tr}}(B,\epsilon) = \mathcal{T}_{\mathrm{Tr}}(B,\epsilon) = O(B^2 \log m/\epsilon^2)$ and a construction of Gibbs sampling $\mathcal{T}_{\mathrm{Gibbs}}(r_K, \Phi, B_K, \epsilon) = O(\Phi \cdot \mathrm{poly}(\log n, r_K, B_K, \epsilon^{-1}))$[4]. As a result,

▶ **Corollary 9.** *For any $\epsilon > 0$, there is a quantum algorithm for the feasibility of the SDP using at most $(\sqrt{m} + \mathrm{poly}(r)) \cdot \mathrm{poly}(\log m, \log n, B, \epsilon^{-1})$ quantum gates and queries to Oracle 1.2, Oracle 1.3, and Oracle 1.4.*

We also show the square-root dependence on m is also optimal by establishing the following result:

▶ **Theorem 10** (lower bound on Corollary 9). *There exists an SDP feasibility testing problem such that $B, r, \epsilon = \Theta(1)$, and solving the problem requires $\Omega(\sqrt{m})$ calls to Oracle 1.2, Oracle 1.3, and Oracle 1.4.*

Comparison between the plain model and the quantum input model. In the quantum input model (Oracle 1.2, Oracle 1.3, and Oracle 1.4), our quantum SDP solver has a *poly-logarithmic* dependence on n (but polynomial in r) and a *square-root* dependence on m, while in the plain input model (Oracle 1.1), the dependence on n needs to be $\Omega(\sqrt{n})$ [9]. It is also worth mentioning that our quantum SDP solver in Corollary 9 does *not* assume the *sparsity* of A_i's, which are crucial for the quantum SDP solvers with the plain model (such as Corollary 4 and Refs. [9, 5]). This is because the quantum input models provide an alternative way to address the technical difficulty that was resolved by the sparsity condition (namely efficient algorithms for Hamiltonian evolution associated with the input matrices of the SDP).

Comparison between quantum and classical input models. The poly-logarithmic dependence on n in Corollary 9 is intriguing and suggests that quantum computers might offer exponential speed-ups for some SDP instances. However one has to be cautious as the input model we consider is inherently quantum, so it is incomparable to classical SDP solvers. As suggested to us by Aram Harrow (personal communication), we could consider a classical setting in which we get as input all inner products between all eigenvectors of the input matrices. Then in that case one could solve the problem classically in time $\mathrm{poly}(r, m, 1/\epsilon)$ (essentially using Jaynes' principle which will be discussed in Section 1.5 to reduce the problem

[4] The construction details are given in Lemma 10 and 12 in the full version of our paper [8].

to a SDP of dimension poly(r)). We have not formalized this approach, and there seems to be some technical problems doing so when the input matrices have close-by eigenvalues. However Harrow's observation shows the importance of justifying the input model in terms of natural applications to argue for the relevance of the run time obtained. We present one application of it in Section 1.5; more applications are given in Ref. [4].

Furthermore, several quantum-inspired classical algorithms were recently proposed originated from Tang [25]. Such classical algorithms assume the following sampling access:

▶ **Definition 11** (Sampling access). *Let $A \in \mathbb{C}^{n\times n}$ be a matrix. We say that we have the* sampling access *to A if we can*

1. *sample a row index $i \in [n]$ of A where the probability of row i being chosen is $\frac{\|A_{i\cdot}\|^2}{\|A\|_F^2}$, and*[5]
2. *for all $i \in [n]$, sample an index $j \in [n]$ where the probability of j being chosen is $\frac{|A_{ij}|^2}{\|A_{i\cdot}\|^2}$*

with time and query complexity $O(\mathrm{poly}(\log n))$ for each sampling.

In particular, we notice that Ref. [10] recently gave a classical SDP solver for (1.1) with complexity $O(m\cdot\mathrm{poly}(\log n, r, \epsilon^{-1}))$, given the above sampling access to $A_1, \ldots, A_m$. We point out that this result is incomparable to Corollary 9 because the sampling access (Definition 11) and our quantum state model (Oracle 1.2, Oracle 1.3, and Oracle 1.4) are incomparable. Nevertheless, it reminds us that under various input models, the speedup of quantum SDP solvers (compared to their classical counterparts) can also vary.

1.3 Related works on quantum SDP solvers

Previous quantum SDP solvers [9, 5] focus on the plain input model. A major contribution of ours is to improve the dependence $O(\sqrt{mn})$ to $O(\sqrt{m}+\sqrt{n})$ (ignoring dependence on other parameters) which is optimal given the lower bound $\Omega(\sqrt{m}+\sqrt{n})$ in [9]. To that end, we have also made a few technical contributions, including bringing in a new SDP solving framework and a fast version of quantum OR lemma, which will be elaborated in Section 1.4.

The quantum input model was briefly mentioned in an earlier version of [9]. The construction of quantum SDP solvers under the quantum input model therein was unfortunately incorrect. We provide the first rigorous mathematical formulation of the quantum input model and its justification in the context of learning quantum states (see Section 1.5). We also provide a construction of quantum SDP solvers in this model with a rigorous analysis. Moreover, we construct the first Gibbs state sampler with quantum inputs.

Subsequent to a previous version of this paper, an independent interesting result by van Apeldoorn and Gilyén [4] has improved the complexity of trace-estimation and Gibbs sampling. After a personal communication [26] introducing our fast version of the quantum OR lemma, the authors of Ref. [4] observed independently that the application of the quantum OR lemma [14] can be applied to decouple the dependence of m and n. As a result, Ref. [4] improved the complexity of Corollary 4 to $\tilde{O}(s(\frac{\sqrt{m}}{\epsilon^4}+\frac{\sqrt{n}}{\epsilon^5}))$ in the quantum operator model, a stronger input model than the plain one proposed by Ref. [4]. Using novel techniques, it also has improved the complexity of Corollary 9 to $\tilde{O}(\frac{B\sqrt{m}}{\epsilon^4}+\frac{B^{3.5}}{\epsilon^{7.5}})$ in the quantum input model. Note there is no explicit dependence on the rank r, which is an important advance (though it can be argued that rank r is implicitly included in the parameter B).

[5] Here $\|A\|_F$ is the Frobenius norm of A and $\|A_{i\cdot}\|$ is the ℓ_2 norm of the i^{th} row of A.

1.4 Techniques

At a high level, and in similarity to Refs. [9, 5], our quantum SDP solver can be seen as a "quantized" version of classical SDP solvers based on the matrix multiplicative weight update (MMWU) method [6]. In particular, we will leverage quantum Gibbs samplers as the main source of quantum speed-ups. In Refs. [9, 5], quantum Gibbs samplers with quadratic speed-ups (e.g., [24, 11]) have been exploited to replace the classical Gibbs state calculation step in [6]. Because the number of iterations in MMWU is poly-logarithmic in terms of the input size, the use of quantum Gibbs samplers, together with a few other tricks, leads to the overall quadratic quantum speed-up.

However, there are a few key differences (our major technical contributions) which are essential for our improvements.

Zero-sum game approach for MMW. Our quantum SDP solvers do not follow the primal-dual approach in Arora-Kale's SDP solver [7] which is the classical counterpart of previous quantum SDP solvers [9, 5]. Instead, we follow a zero-sum game framework to solve SDP feasibility problems, which is also based on the MMWU method. This framework has appeared in the classical literature (e.g., [15]) and has already been used to in semidefinite programs of relevance in quantum complexity theory (e.g., [27, 12, 19]). Let us briefly describe how the zero-sum game framework works when solving the SDP feasibility problem (1.1).

Assume there are two players. Player 1 wants to provide a feasible $X \in \mathcal{S}_\epsilon$. Player 2, on the other side, wants to find any violation of any proposed X, which can be formulated as follows.

▶ **Oracle 1.5** (Search for violation). *Inputs a density matrix X, outputs an $i \in [m]$ such that $\mathrm{Tr}(A_i X) > a_i + \epsilon$. If no such i exists, output "FEASIBLE".*

If the original problem is feasible, there exists a feasible point X_0 (provided by Player 1) such that there is no violation of X_0 that can be found by Player 2 (i.e., Oracle 1.5). This actually refers to an *equilibrium* point of the zero-sum game, which can also be approximated by the matrix multiplicative weight update method [6].

We argue that there are a few advantages of adopting this framework. One prominent example is its simplicity, which perhaps provides more intuition than the primal-dual approach. Together with our choice of the approximate feasibility problem, our presentation is simple both conceptually and technically (indeed, the simplicity of this framework has led to the development of the fast quantum OR lemma, another main technical contribution of ours.) Another example is that the zero-sum game approach does not make use of the dual program of SDPs and thus there is no dependence on the size of any dual solution. The game approach also admits an intuitive application of our SDP solvers to learning quantum states Section 1.5, which coincides with the approach adopted by [19] in a similar context.

One might wonder whether the simplicity of this framework will restrict the efficiency of SDP solvers. As indicated by the independent work of van Apeldoorn and Gilyén [4] which has achieved the same complexity of quantum SDP solvers following both the primal-dual approach and the zero-sum approach, we conclude that it is not the case at least up to our current knowledge.

Fast quantum OR lemma. We now outlines what is the main idea to find a solution to Oracle 1.5 efficiently. Roughly speaking, the idea behind previous quantum SDP solvers [9, 5] when applied to this context was to generate a new copy of a quantum state X for each time one would query the expectation value of one of the input matrices on it. The cost of

generating X (i.e., Gibbs sampling) is $O(\sqrt{n})$ (ignoring the dependence on other parameters) and one can use a Grover-search-like approach to test for m constraints with $O(\sqrt{m})$ iterations. The resultant cost is then $O(\sqrt{mn})$. Our key observation is to leverage the quantum OR lemma [14] to detect a single violation with only a single copy of X.

At a high level, given a single copy of any state ρ and m projections $\Lambda_1, \ldots, \Lambda_m$, the quantum OR lemma describes a procedure to distinguish between the case that $\exists\, i \in [m]$ s.t. $\mathrm{Tr}[\rho\Lambda_i]$ is very large, or $\frac{1}{m}\sum_{i=1}^{m} \mathrm{Tr}[\rho\Lambda_i]$ is very small. It is not hard to see that with some gap-amplification step and a search-to-decision reduction, the above procedure will output a violation i^* if any. By using quantum OR lemma, one can already decouple the cost of generating X and the number of iterations in violation-detection.

Unfortunately, Ref. [14] has only been focusing on the use of a single copy of ρ, while its gate complexity is $O(m)$ for m projections. To optimize the gate complexity, we develop the following fast implementation of the quantum OR lemma with gate complexity $O(\sqrt{m})$, using ideas from the fast amplification technique in [22]. Overall, this leads to a complexity of $O(\sqrt{m} + \sqrt{n})$.

▶ **Lemma 12.** *Let $\Lambda_1, \ldots, \Lambda_m$ be projections, and fix parameters $0 < \varepsilon \leq 1/2$ and $\varphi, \xi > 0$. Let ρ be a state such that either $\exists\, j \in [m]$ $\mathrm{Tr}[\rho\Lambda_j] \geq 1 - \varepsilon$, or $\frac{1}{m}\sum_{j=1}^{m} \mathrm{Tr}[\rho\Lambda_j] \leq \varphi$. There is a test using one copy of ρ and $O(\xi^{-1}\sqrt{m}(p + \mathrm{poly}(\log m)))$ operations such that: in the former case, accepts with probability at least $(1-\varepsilon)^2/4 - \xi$; in the latter case, accepts with probability at most $3\varphi m + \xi$.*

The dependence on m is also tight, as one can easily embed Grover search into this problem.

Gibbs sampler with quantum inputs. To work with the quantum input model, as our main technical contribution, we construct the first quantum Gibbs sampler of low-rank Hamiltonians when given Oracles 1.2 and 1.3:

▶ **Theorem 13.** *Assume the $n \times n$ matrix $K = K^+ - K^-$ and K^+, K^- are PSD matrices with rank at most r_K and $\mathrm{Tr}[K^+] + \mathrm{Tr}[K^-] \leq B$. Given quantum oracles that prepare copies of $\rho^+ = K^+/\mathrm{Tr}(K^+)$, $\rho^- = K^-/\mathrm{Tr}(K^-)$ and estimates of $\mathrm{Tr}(K^+)$, $\mathrm{Tr}(K^-)$, there is a quantum Gibbs sampler that prepares the Gibbs state $\rho_G = \exp(-K)/\mathrm{Tr}(\exp(-K))$ to precision ϵ in trace distance, using $\mathrm{poly}(\log n, r_K, B, \epsilon^{-1})$ quantum gates.*

Our quantum Gibbs sampler has a poly-logarithmic dependence on n and polynomial dependence on the maximum rank of the input matrices, while in the plain input model the dependence of n is $\Theta(\sqrt{n})$ [24, 11]. Our construction deviates significantly from [24, 11]. Because of the existence of copies of ρ^+ and ρ^-, we rely on efficient Hamiltonian simulation techniques developed in quantum principle component analysis (PCA) [21] and its follow-up work in [18]. As a result, we can also get rid of the sparsity assumption which is crucial for evoking results about efficient Hamiltonian simulation into the Gibbs sampling used in [24, 11].

1.5 Application: Efficient learnability of quantum states

Problem description. Given many realizations of an experiment producing a quantum state with density matrix ρ, learning an approximate description of ρ is a fundamental task in quantum information and experimental physics. It refers to *quantum state tomography*, which has been widely used to identify quantum systems. However, to tomograph an ℓ-qubit state ρ (with dimension $n = 2^\ell$) , the optimal procedure [23, 13] requires n^2 number of copies of ρ, which is impractical already for relatively small ℓ.

An interesting alternative is to find a description of the unknown quantum state ρ which approximates $\mathrm{Tr}[\rho E_i]$ up to error ϵ for a specific collection of POVM elements $E_1, \ldots, E_m$, where $I \succeq E_i \succeq 0$ and $E_i \in \mathbb{C}^{n\times n}, \forall i \in [m]$. This is an old problem, dating back at least to the work of Jaynes on statistical mechanics in the 50ies. Jaynes' principle [17] (also known as the principle of maximum entropy) gives a general form for the solution of the problem above. It shows that there is always a state of the form

$$\frac{\exp\left(\sum_i \lambda_i E_i\right)}{\mathrm{Tr}\left(\exp\left(\sum_i \lambda_i E_i\right)\right)}, \tag{1.6}$$

which has the same expectation values on the E_i's as the original state ρ, where the λ_i's are real numbers. In words, there is always a Gibbs state with Hamiltonian given by a linear combination of the E_i's which gives the same expectation values as the state described by ρ. Therefore one can solve the learning problem by finding the right λ_i's (or finding a quantum circuit creating the state in Eq. (1.6)).

Applying quantum SDP solvers. By formulating the learning problem in terms of the SDP feasibility problem (with each A_i replaced by E_i) where one looks for a trace unit PSD σ matching the measurement statistics, i.e., $\mathrm{Tr}(\sigma E_i) \approx \mathrm{Tr}(\rho E_i), \forall i \in [m]$, we observe that our quantum SDP solvers actually provides a solution to the learning problem with associated speed-ups on m and n.

In fact, our algorithm also outputs each of the λ_i's (one can show that $\mathrm{poly}(\log(mn))/\varepsilon^2$ non-zero of them suffices for a solution with error ε), as well as a circuit description of the Gibbs state in Eq. (1.6) achieving the same expectation values as ρ up to error ε. (This is mainly because the similarity between the matrix multiplicative update method and Jaynes' principle.) In this sense our result can be seen as an *algorithmically* version of Jaynes' principle. We note that a similar idea was adopted by [19] in learning quantum states, although for a totally different purpose (namely proving lower bounds on the size of SDP approximations to constraint satisfaction problems).

It is worthwhile noting that our quantum SDP solvers when applied in this context will output a description of the state ρ in the form of Eq. (1.6) which has the same expectation values as ρ on measurements $E_1, \ldots, E_m$ up to error ϵ. This is slightly different from directly outputting estimates of $\mathrm{Tr}(E_i\rho)$ for each $i \in [m]$, which by itself will take $\Omega(m)$ time.

Relevance of quantum input model. More importantly, we argue that our quantum input model is *relevant* in this setting for low-rank measurements E_i's. Since all $E_i \succeq 0$ by definition, we can consider the following (slightly simplified version of) oracles:

Oracle 1.2 for traces of E_i: A unitary O_{Tr} such that for any $i \in [m]$, $O_{\mathrm{Tr}}|i\rangle|0\rangle = |i\rangle|\,\mathrm{Tr}[E_i]\rangle$.

Oracle 1.3 for preparing E_i: A unitary O such that for any $i \in [m]$, $O|i\rangle\langle i| \otimes |0\rangle\langle 0|O^\dagger = |i\rangle\langle i| \otimes |\psi_i\rangle\langle\psi_i|$, where $|\psi_i\rangle\langle\psi_i|$ is any purification of $E_i/\,\mathrm{Tr}[E_i]$.

We now show how one can implement this oracle in the case where each E_i is a low rank projector and we have an efficient (with $\mathrm{poly}\log(n)$ many gates) implementation of the measurement. Let the rank of E_i's bounded by r and suppose the measurement operators E_i's are of the form

$$E_i = V_i P_i V_i^\dagger \tag{1.7}$$

for polynomial (in $\log(n)$) time circuits V_i, and projectors P_i of the form

$$P_i := \sum_{i=1}^{r_i} |i\rangle\langle i| \tag{1.8}$$

with $|i\rangle$ the computational basis and $r_i \leq r$. Then for Oracle 1.2 we just need to output the r_i's. Oracle 1.3 can be implemented efficiently (in time $r \operatorname{poly}\log(n)$) by first creating a maximally entangled state between the subspace spanned by P_i and a purification and applying V_i to one half of it. In more detail, consider the following purification of $E_i/\operatorname{Tr}(E_i)$:

$$|\psi_i\rangle := \frac{1}{\sqrt{r_i}} \sum_{i=1}^{r_i} (V_i \otimes I)|i,i\rangle \tag{1.9}$$

This can be constructed first by preparing the state $\frac{1}{\sqrt{r_i}} \sum_{i=1}^{r_i} |i,i\rangle$ in time r_i and then applying $V_i \otimes I$ to it (which can be done in time $\operatorname{poly}\log(n)$).

Efficient learning for low rank measurements. By applying our SDP solver in the quantum input model, we obtain that

▶ **Theorem 14.** *For any $\epsilon > 0$, there is a quantum procedure that outputs a description of the state ρ in the form of Eq. (1.6) (namely the λ_i's parameters) using at most $\operatorname{poly}(\log m, \log n, r, \epsilon^{-1})$ copies of ρ and at most $\sqrt{m} \cdot \operatorname{poly}(\log m, \log n, r, \epsilon^{-1})$ quantum gates and queries to Oracle 1.2 and Oracle 1.3.*

Let us briefly sketch how our SDP solver applies to this setting. Note first that we do not aim to estimate $\operatorname{Tr}(E_i\rho)$ for each $i \in [m]$, which helps us circumvent the $\Omega(m)$ lower bound. What we really want is to generate a state $\tilde{\rho}$ such that $\operatorname{Tr}(E_i\tilde{\rho}) \approx \operatorname{Tr}(E_i\rho)$ for each i. Our SDP solver will maintain and update a description of $\tilde{\rho}$ per iteration. In each iteration, given copies of $\tilde{\rho}$ and the actual unknown state ρ, we want to know whether $\operatorname{Tr}(E_i\tilde{\rho}) \approx \operatorname{Tr}(E_i\rho)$ $\forall i \in [m]$ or there is at least a violation i^*. To that end, we design for each i a projection for the following procedure: (1) perform multiple independent SWAP tests between $E_i/\operatorname{Tr}[E_i]$ (from Oracle 1.3) and ρ, $\tilde{\rho}$ respectively; (2) accept when the statistics of both SWAP tests (one with ρ, the other with $\tilde{\rho}$) are close. Hence, one can apply our fast quantum OR lemma on these projections to find such i^* if it exists.

Note that both the sample complexity and the gate complexity of the above procedure have a poly-log dependence on n (i.e., the dimension of the quantum state to learn).

Shadow tomography problem. In a sequence of works [1, 2], Aaronson asked whether one can predict information about a dimension-n quantum state with poly-log(n) many copies. In Ref. [1], he showed that a linear number of copies is sufficient to predict the outcomes of "most" measurements according to some (arbitrary) distribution over a class of measurements. Very recently, in Ref. [2], he referred the following problem as the "shadow tomography" problem: for any n-dimensional state ρ and two-outcome measurements $E_1, \ldots, E_m$, estimate $\operatorname{Tr}[\rho E_i]$ up to error ϵ, $\forall i \in [m]$. He has further designed a quantum procedure for the shadow tomography problem with $\tilde{O}(\ell \cdot \log^4 m/\epsilon^5)$ [6] copies of ρ.

[6] Here $\tilde{O}$ hides factors that are polynomial in $\log\log m$, $\log\log n$, and $\log 1/\epsilon$.

Noting that the shadow tomography problem is essentially the same problem considered by Jaynes [17], one can apply Jaynes' principle and its algorithmic version we discussed before. Although this can be used to give a version of the result of Ref. [2], Aaronson obtained his result [2] through a different route, based on a post-selection argument. A drawback of this approach is that its gate complexity is high, scaling linearly in m and as $n^{O(\log\log n)}$ (for fixed error).

Our Theorem 14 can be applied here to improve the time complexity. It gives a quantum procedure with a *square-root* dependence on m and $n^{O(1)}$ dependence on n for arbitrary E_i's.

When we assume r is small, say $r = O(\mathrm{poly}\log n)$, the gate complexity of the entire procedure becomes $\tilde{O}(\sqrt{m}\,\mathrm{poly}\log(n))$. This gives a class of measurement (namely any set of low-rank measurements which can be efficiently implemented) for which the learning problem is efficient both in the number of samples and the computational complexity. This solves an open problem proposed in Ref. [1]

Although we have not worked out an explicit bound of the sample complexity of our procedure, the authors of [4] followed our approach with more sophisticated techniques and obtained a sample complexity of $\tilde{O}(\ell \cdot \log^4 m/\epsilon^4)$, improving on the bound from [2]. We also note that very recently, Aaronson et al. claimed the same sample complexity (i.e., $\tilde{O}(\ell \cdot \log^4 m/\epsilon^4)$) in [3].

1.6 Overview of detailed results and proofs

In the full version of our paper [8], we formulate the SDP feasibility problem and prove the correctness of the basic framework in Appendix A. Our implementation of the fast quantum OR lemma is given in Appendix B. We describe our main results the constructions of quantum SDP solvers in the plain input model and the quantum input model in Appendix C and Appendix D, respectively. The application to learning quantum states is illustrated in Appendix E. In Appendix F (with full details in Appendix G) we demonstrate how to sample from the Gibbs state of low-rank Hamiltonians.

1.7 Open questions

This work leaves several natural open questions for future work. For example:

- Are there more examples of interesting SDPs where our form of input is meaningful? We have shown the example of learning quantum states. Intuitively, we are looking for SDP instances where the constraints are much "simpler" than the solution space. Is there any such example in the context of big data and/or machine learning?
- Our work has identified one setting where Gibbs sampling has a poly-log dependence on the dimension? Is there any other setting for the same purpose?
- For any reasonable quantum input setting, what is the effect of potential noises on quantum inputs in practice?
- Can we improve further on other parameters (e.g., the dependence on m and $1/\epsilon$)? In particular, is it possible to improve the error dependence to $\mathrm{poly}\log(1/\epsilon)$? This probably implies that we have to consider a quantum version of the interior point method.
- Are there other classes of measurements for which the quantum learning problem can solved in a computationally efficient way beyond the low-rank measurements we consider in this work? We note that most measurements of interest are not low rank (e.g. local measurements) and therefore the practical applicability of the present result is limited.

References

1 Scott Aaronson. The learnability of quantum states. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 463(2088):3089–3114, 2007. arXiv:quant-ph/0608142.

2 Scott Aaronson. Shadow Tomography of Quantum States. In *Proceedings of the Fiftieth Annual ACM Symposium on Theory of Computing*. ACM, 2018. arXiv:1711.01053.

3 Scott Aaronson, Xinyi Chen, Elad Hazan, and Ashwin Nayak. Online Learning of Quantum states, 2018. arXiv:1802.09025.

4 Joran van Apeldoorn and András Gilyén. Improvements in Quantum SDP-solving with Applications, 2018. arXiv:1804.05058.

5 Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Quantum SDP-Solvers: Better upper and lower bounds. In *58th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 2017. arXiv:1705.01843.

6 Sanjeev Arora, Elad Hazan, and Satyen Kale. The Multiplicative Weights Update Method: a Meta-Algorithm and Applications. *Theory of Computing*, 8(6):121–164, 2012. doi:10.4086/toc.2012.v008a006.

7 Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, pages 227–236. ACM, 2007.

8 Fernando G. S. L. Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M. Svore, and Xiaodi Wu. Quantum SDP Solvers: Large Speed-ups, Optimality, and Applications to Quantum Learning, 2019. arXiv:1710.02581.

9 Fernando G. S. L. Brandão and Krysta Svore. Quantum speed-ups for semidefinite programming. In *58th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 2017. arXiv:1609.05537.

10 Nai-Hui Chia, Tongyang Li, Han-Hsuan Lin, and Chunhao Wang. Quantum-inspired classical sublinear-time algorithm for solving low-rank semidefinite programming via sampling approaches, 2019. arXiv:1901.03254.

11 Anirban Narayan Chowdhury and Rolando D. Somma. Quantum algorithms for Gibbs sampling and hitting-time estimation. *Quantum Information & Computation*, 17(LA-UR-16-21218), 2017. arXiv:1603.02940.

12 Gus Gutoski and Xiaodi Wu. Parallel approximation of min-max problems with applications to classical and quantum zero-sum games. In *Proceedings of the Twenty-seventh Annual IEEE Symposium on Computational Complexity (CCC)*, pages 21–31. IEEE, 2012.

13 Jeongwan Haah, Aram W. Harrow, Zhengfeng Ji, Xiaodi Wu, and Nengkun Yu. Sample-optimal Tomography of Quantum States. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing*, pages 913–925. ACM, 2016. arXiv:1508.01797.

14 Aram W. Harrow, Cedric Yen-Yu Lin, and Ashley Montanaro. Sequential measurements, disturbance and property testing. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1598–1611. SIAM, 2017. arXiv:1607.03236.

15 Elad Hazan. *Efficient algorithms for online convex optimization and their applications*. PhD thesis, Princeton University, 2006.

16 Rahul Jain, Zhengfeng Ji, Sarvagya Upadhyay, and John Watrous. QIP=PSPACE. *Journal of the ACM (JACM)*, 58(6):30, 2011. arXiv:0907.4737.

17 Edwin T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106(4):620, 1957.

18 Shelby Kimmel, Cedric Yen-Yu Lin, Guang Hao Low, Maris Ozols, and Theodore J. Yoder. Hamiltonian simulation with optimal sample complexity. *npj Quantum Information*, 3(1):13, 2017. arXiv:1608.00281.

19 James R. Lee, Prasad Raghavendra, and David Steurer. Lower Bounds on the Size of Semidefinite Programming Relaxations. In *Proceedings of the Forty-seventh Annual ACM*

Symposium on Theory of Computing, pages 567–576, New York, NY, USA, 2015. ACM. doi:10.1145/2746539.2746599.

20 Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *Proceedings of the Fifty-sixth Annual IEEE Symposium on Foundations of Computer Science*, pages 1049–1065. IEEE, 2015. arXiv:1508.04874.

21 Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631, 2014. arXiv:1307.0401.

22 Daniel Nagaj, Pawel Wocjan, and Yong Zhang. Fast amplification of QMA, 2009. arXiv:0904.1549.

23 Ryan O'Donnell and John Wright. Efficient Quantum Tomography. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing*, pages 899–912. ACM, 2016. arXiv:1508.01907.

24 David Poulin and Pawel Wocjan. Sampling from the thermal quantum Gibbs state and evaluating partition functions with a quantum computer. *Physical Review Letters*, 103(22):220502, 2009. arXiv:0905.2199.

25 Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing*. ACM, 2019. arXiv:1807.04271.

26 Ronald de Wolf. Personal communication, 2017.

27 Xiaodi Wu. Parallelized Solution to Semidefinite Programmings in Quantum Complexity Theory, 2010. arXiv:1009.2211.

A Simple Protocol for Verifiable Delegation of Quantum Computation in One Round

Alex B. Grilo
CWI, Amsterdam, The Netherlands
QuSoft, Amsterdam, The Netherlands
abgrilo@gmail.com

Abstract

The importance of being able to verify quantum computation delegated to remote servers increases with recent development of quantum technologies. In some of the proposed protocols for this task, a client delegates her quantum computation to non-communicating servers in multiple rounds of communication. In this work, we propose the first protocol where the client delegates her quantum computation to two servers in one-round of communication. Another advantage of our protocol is that it is conceptually simpler than previous protocols. The parameters of our protocol also make it possible to prove security even if the servers are allowed to communicate, but respecting the plausible assumption that information cannot be propagated faster than speed of light, making it the first relativistic protocol for quantum computation.

2012 ACM Subject Classification Hardware → Quantum communication and cryptography; Theory of computation → Quantum complexity theory

Keywords and phrases quantum computation, quantum cryptography, delegation of quantum computation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.28

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1711.09585`.

Funding *Alex B. Grilo*: Supported by ERC Consolidator Grant 615307-QPROGRESS.

Acknowledgements I thank Iordanis Kerenidis, Damián Pitalúa-García and Thomas Vidick for useful discussions and comments in early drafts of this manuscript. I also thank anonymous reviewers helped me improving the presentation of this work. Part of this work was done when I was member of IRIF, Université Paris Diderot, Paris, France, where I was supported by ERC QCC and French Programme d'Investissement d'Avenir RISQ P141580.

1 Introduction

With the recent progress in the development of quantum technologies, large-scale quantum computers may be available in a not-so-distant future. Their costs and infrastructure requirements make it impractical for them to be ubiquitous, however clients could send their quantum computation to be performed remotely by a quantum server in the cloud [9], broadening the use of quantum advantage to solve computational problems (see Ref. [24] for such examples). For the clients, it is a major concern whether the quantum servers are performing the correct computation and quantum speedup is really being experienced.

In order to solve this problem, we aim a protocol for verifiable delegation of quantum computation where the client exchanges messages with the server, and, at the end of the protocol, either the client holds the output of her computation, or she detects that the server is defective. Ideally, the client is a classical computer and an honest server only needs polynomial-time quantum computation to answer correctly. Also, one would aim for *blind* protocols, in which the server does not learn the circuit delegated by the client. We


notice that verification protocols could also be used for validating devices that claim to have quantum computational power, but in this work we focus on the point of view of delegation of computation.

There are efficient protocols that can perform this task if the model is relaxed, for instance giving limited quantum power and quantum communication to the client [14, 3, 8, 25, 26]. There are also protocols where the security of the protocol only holds against bounded malicious servers [20].In this work, we focus on a third line of protocols, where a classical client delegates her computation to non-communicating quantum servers. Although the servers are supposed to share and maintain entangled states, which is feasible in principle but technologically challenging, these protocols are "plug-and-play" in the sense that the client only needs classical communication with the quantum servers.

Following standard notation in these protocols, we start calling the client and servers by verifier and provers, respectively. The security of such protocols relies on the so called self-testing of non-local games. We consider games where a verifier interacts with non-communicating provers by exchanging one round of classical communication and, based on the correlation of the provers' answers, the verifier decides to accept or reject. The goal of the provers is to maximize the acceptance probability in the game and they can share a common strategy before the game starts. A game is non-local [6] whenever there exists a quantum strategy for the provers that achieves acceptance probability strictly higher than any classical strategy, allowing the verifier to certify that the provers share some entanglement, if the classical bound is surpassed. Self-testing [21] goes one step further, proving that if the correlation of the provers' answers is close to the optimal quantum value, their strategy is close to the honest one.

Reichardt, Unger and Vazirani [32] used the ideas of self-testing to propose a verifiable delegation scheme where the verifier interleaves questions of non-local games and instructions for the computation, and from the point of view of the provers, these two types of questions are indistinguishable. In this case, the correctness of the quantum computation is inherited by the guarantees achieved in self-testing. Follow-up works [22, 15, 17, 13, 27, 10] have used the same approach in order to propose more efficient protocols (see Table 1 for summary of the properties of the different protocols).

In this work, we present the first one-round protocol for verifiable delegation of quantum computation. We notice that our protocol is conceptually simple, in contrast with previous protocols that have a rather complicated structure. We expect that its main ideas can be generalized to other contexts as MIP* protocols for iterated non-deterministic exponential time and even in new protocols for delegation of quantum computation. We also remark that the parameters of the protocol allow us to replace the unjustified assumption that the provers do not communicate to a more plausible assumption that the communication cannot be faster than speed of light.

Technically, we achieve our protocol by showing a non-local game for Local Hamiltonian problem, where the verifier plays against two provers in one round of classical communication. In this game, honest provers perform polynomial time quantum computation on copies of the groundstate of the Hamiltonian. This non-local game is of independent interest since it was an open question if a one-round game for Local Hamiltonian problem could be achieved with only two efficient provers. This non-local game can be used as a delegation protocol through the circuit-to-Hamiltonian construction.

Table 1 Comparison between different protocols for verifiable delegation of quantum computation.

	Client	Provers	Rounds	Blind	Security
ABOEM [3]	Quantum	1	poly(g)	yes	information theoretical
FK [14]	Quantum	1	poly(g)	yes	information theoretical
RUV [32]	Classical	2	$\geq g^{8192}$	yes	information theoretical
McKague [22]	Classical	poly(n)	$\geq 2^{153}g^{22}$	yes	information theoretical
GKW [15]	Classical	2	$\geq g^{2048}$	yes	information theoretical
HPDF [17]	Classical	poly(n)	$\Theta(g^4 \log g)$	yes	information theoretical
FH [13]	Classical	5	2	no	information theoretical
NV [27, 28]	Classical	7	2	no	information theoretical
CGJV [10]	Classical	2	O(depth)	yes	information theoretical
CFJV [10]	Classical	2	2	no	information theoretical
Mahadev [20]	Classical	1	2	no	computational
This work	Classical	2	1	no	information theoretical

1.1 Our contributions

New Non-local game for Local Hamiltonians. The main technical result of this work is presenting one-round two-prover game for the Local Hamiltonian problem, where honest provers only need quantum polynomial time computation, copies of the groundstate of the Hamiltonian and shared EPR pairs. More concretely, we show how to construct a game $G(H)$ based on a XZ Local Hamiltonian[1] H acting on n qubits and the upper and lower bounds on the maximum acceptance probability in $G(H)$ are tightly related to the groundstate energy of H. Then, based on $G(H)$, we devise a game $\tilde{G}(H)$ such that if the groundstate energy of H is low, then the maximum acceptance probability in $\tilde{G}(H)$ is at least $\frac{1}{2} + \Delta$, while if the groundstate energy is high, the acceptance probability in the game is at most $\frac{1}{2} - \Delta$. We describe now the main ideas of $G(H)$.

The game is composed by two tests: the Pauli Braiding Test (PBT) [27], where the verifier checks if the provers share the expected state and perform the indicated Pauli measurements, and the Energy Test (ET), where the verifier estimates the groundstate energy of H.

The same structure was used in a different way in the non-local game for LH proposed by Natarajan and Vidick [27] (and implicitly in Ji [18]). In their game, 7 provers are expected to share the encoding of the groundstate of H under a quantum error correcting code. In ET, the provers estimate the groundstate energy by jointly performing the measurements on the state, while PBT checks if the provers share a correct encoding of some state and if they perform the indicated measurements. The provers receive questions consisting in a Pauli tensor product observable and they answer with the one-bit outcome of the measurement on their share of the state. The need of 7 provers comes from the fact that the verifier must test if the provers are committed to an encoded state and use it in all of their measurements. It is an open problem if the number of provers can be decreased in this setup.

In this work, we are able to reduce the number of provers to 2 by making them asymmetric. In ET, one of the provers holds the groundstate of H and teleports it to the second prover, who is responsible for measuring it. In our case, PBT checks if the provers share EPR pairs

[1] An XZ Local Hamiltonian is a Hamiltonian that can be decomposed in sum of polynomially many terms that are tensor products of Paulis σ_X, σ_Z and σ_I

and if the second prover's measurements are correct. We remark that no test is needed for the state, since the chosen measurement is not known by the first prover. We notice that the size of the answers in our protocol is polynomial in n, since the verifier needs the teleportation results for every qubit (in order to hide the measurement). We leave as an open problem if the size of the answers can be reduced, hopefully achieving constant-size answers as in [27].

We state now the key ideas to upper bound the maximum acceptance probability of $G(H)$. The behavior of the second prover in ET can be verified thanks to PBT, since the two tests are indistinguishable to him. On the other hand, the first prover can perfectly distinguish PBT and ET, but he has no information about the measurement being performed. We show that his optimal strategy is to teleport the groundstate of H, but in this case the acceptance probability is high iff the groundstate energy is low.

Protocol for verifiable delegation of quantum computation. The task of verifiable delegation of quantum computation can be easily reduced to estimating the groundenergy of local Hamiltonians through the circuit-to-Hamiltonian construction [13, 27], which has been called *post-hoc verification of quantum computation*. In this construction, a quantum circuit Q is reduced to an instance H_Q of LH, such that H_Q has low groundstate energy iff Q accepts with high probability. Our non-local game for H_Q can be seen as a delegation protocol, where the verifier interacts with two non-communicating entangled provers in one-round of classical communication.

When compared to previous protocols, our result has some very nice properties. First, differently to previous results, our protocol is very simple to state, which could make it easier to be extended to other settings. Secondly, using standard techniques in relativistic cryptography, we can replace the unjustified assumption that the two servers do not communicate by the No Superluminal Signaling (NSS) principle: the security of the protocol would only rely that the two servers cannot communicate faster than the speed of light.

The circuit-to-Hamiltonian construction also causes an overhead on the resources needed by honest provers. Namely, in our protocol the provers need $\tilde{O}(ng^2)$ EPR pairs for delegating the computation of a quantum circuit acting on n qubits and composed by g gates, while other protocols need only $\tilde{O}(g)$ EPR pairs [10]. We leave as an open problem finding more efficient two-provers one-round protocol for delegating quantum computation.

We also leave as an open question if it is possible to create a one-round and blind verifiable delegation scheme for quantum computation, or proving that this is improbable, in the lines of Ref. [1].

Non-local games for QMA. In Complexity Theory, the connection between the PCP theorem [5, 4, 12] and multi-prover games [7] has had a lot of fruitful consequences, such as tighter inapproximability results [31]. Our protocol directly implies a one-round two-prover game for QMA but with polynomial-size questions and answers. We wonder if it could be used to prove the game version of the quantum PCP theorem with two prover [28].

Organization

In Section 2, we give the necessary preliminaries, including the definition of the Pauli Braiding Test. Then, in Section 3 we present our non-local game for local Hamiltonian problem.

2 Preliminaries

We assume basic knowledge on Quantum Computation topics and we refer the readers that are not familiar with them to Ref. [29].

2.1 Notation

We denote $[n]$ as the set $\{1, ..., n\}$. For a finite set S, we denote $x \in_R S$ as x being an uniformly random element from S. We assume that all Hilbert spaces are finite-dimensional. For a Hilbert space $\mathcal{H}$ and a linear operator M on $\mathcal{H}$, we denote $\lambda_0(M)$ as its smallest eigenvalue and $\|M\|$ as its maximum singular value. An n-qubit binary observable O is a Hermitian matrix with eigenvalues ± 1. We denote $\mathrm{Obs}(\mathcal{H})$ as the set of binary observables on the Hilbert space $\mathcal{H}$.

We will use the letters X, Z and I to denote questions in multi-prover games, the letters in the sans-serif font X, Z and I to denote unitaries and σ_X, σ_Z and σ_I to denote observables such that

$$\mathsf{I} = \sigma_I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad \mathsf{X} = \sigma_X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \text{ and } \qquad \mathsf{Z} = \sigma_Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

2.2 Non-local games, Self-testing and the Pauli Braiding Test

We consider games where a verifier plays against two provers in the following way. The verifier sends questions to the provers according to a publicly known distribution and the provers answer back to the verifier. Based on the correlation of the answers, the verifier decides to accept or reject according to an acceptance rule that is also publicly known. The provers share a common strategy before the game starts in order to maximize the acceptance probability in the game, but they do not communicate afterwards.

For a game G, its classical value $\omega(G)$ is the maximum acceptance probability in the game if the provers share classical randomness, while the quantum value $\omega^*(G)$ is the maximum acceptance probability if they are allowed to follow a quantum strategy, i.e. share a quantum state and apply measurements on it. Non-local games (or Bell tests) [6] are such games where $\omega^*(G) > \omega(G)$ and they have played a major role in Quantum Information Theory, since they allow the verifier to certify that there exists some quantumness in the strategy of the provers, if the classical bound is surpassed.

Self-testing (also known as device-independent certification or rigidity theorems) of a non-local game G allows us to achieve stronger conclusions by showing that if the acceptance probability on G is close to $\omega^*(G)$, then the strategy of the provers is close to the ideal one, up to local isometries.

2.2.1 Magic Square game

The Magic Square or Mermin-Peres game [23, 30], is a two-prover non-local game where one of the provers is asked a row $r \in \{1, 2, 3\}$ and the second prover is asked with a column $c \in \{1, 2, 3\}$. The first and second prover answer with $a_1, a_2 \in \{\pm 1\}$ and $b_1, b_2 \in \{\pm 1\}$, respectively. By setting $a_3 = a_1 \oplus a_2$ and $b_3 = b_1 \oplus b_2$, the provers win the game if $a_c = b_r$.

If the provers follow a classical strategy, their maximum winning probability in this game is $\frac{8}{9}$, while we describe now a quantum strategy that makes them win with probability 1. The provers share two EPR pairs and, on question r (resp. c), the prover performs the measurements indicated in the first two columns (resp. rows) of row r (resp. column c) of the following table

IZ	ZI	ZZ
XI	IX	XX
XZ	ZX	YY

and answer with the outcomes of the measurements. The values a_3 and b_3 should correspond to the measurement of the EPR pairs according to the third column and row, respectively.

The self-testing theorem proved by Wu, Bancal and Scarani [33] states that if the provers win the Magic Square game with probability close to 1, they share two EPR pairs and the measurements performed are close to the honest Pauli measurements, up to local isometries.

▶ **Lemma 1.** *Suppose a strategy for the provers, using state $|\psi\rangle$ and observables W, succeeds with probability at least $1-\varepsilon$ in the Magic Square game. Then there exist isometries $V_D : \mathcal{H}_D \to (\mathbb{C}^2 \otimes \mathbb{C}^2)_{D'} \otimes \mathcal{H}_{\hat{D}}$, for $D \in \{A, B\}$ and a state $|\mathrm{AUX}\rangle_{\hat{A}\hat{B}} \in \mathcal{H}_{\hat{A}} \otimes \mathcal{H}_{\hat{B}}$ such that*

$$\left\|(V_A \otimes V_B)|\psi\rangle_{AB} - |\Phi_{00}\rangle^{\otimes 2}_{A'B'}|\mathrm{AUX}\rangle_{\hat{A}\hat{B}}\right\|^2 = O(\sqrt{\varepsilon}),$$

and for $W \in \{I, X, Z\}^2$,

$$\left\|\left(W - V_A^\dagger \sigma_W V_A\right) \otimes I_B|\psi\rangle\right\|^2 = O(\sqrt{\varepsilon}).$$

2.2.2 Pauli Braiding Test

The starting point of our work is the Pauli Braiding Test(PBT) [27], a non-local game that allows the verifier to certify that two provers share t EPR pairs and perform the indicated measurements, which consist of tensors of Pauli observables.

We define PBT in details later in this section and here we state the main properties that will be used in our Hamiltonian game. In PBT, each prover receives questions in the form $W \in \{X, Z, I\}^t$, and each one is answered with some $b \in \{-1, +1\}^t$. For $W \in \{X, Z\}^t$ and $a \in \{0, 1\}^t$, we have $W(a) \in \{X, Z, I\}^t$ where $W(a)_i = W_i$ if $a_i = 1$ and $W(a)_i = I$ otherwise.

In the honest strategy, the provers share t EPR pairs and measure them with respect to the observable $\sigma_W \stackrel{\text{def}}{=} \bigotimes_{i\in[t]} \sigma_{W_i}$ on question W. However, the provers could deviate and perform an arbitrary strategy, sharing an entangled state $|\psi\rangle_{AB} \in \mathcal{H}_\mathsf{A} \otimes \mathcal{H}_\mathsf{B}$ and performing projective measurements τ^A_W and τ^B_W for each possible question W. It was shown that if the provers pass PBT with probability $1-\varepsilon$, their strategy is, up to local isometries, $O(\sqrt{\varepsilon})$-close to sharing t EPR pairs and measuring σ_W on question W [27].

We describe now PBT. The test is divided in three different tests, which are performed with equal probability. The first one, the Consistency Test, checks if the measurement performed by both provers on question W are equivalent, i.e. $\tau^A_W \otimes I_B|\psi\rangle_{AB}$ is close to $I_A \otimes \tau^B_W|\psi\rangle_{AB}$. In the Linearity Test, the verifier checks if the measurement performed by the provers are linear, i.e. $\tau^A_{W(a)}\tau^A_{W(a')} \otimes I_B|\psi\rangle_{AB}$ is close to $\tau^A_{W(a+a')} \otimes I_B|\psi\rangle_{AB}$. Finally, in the Anti-commutation Test, the verifier checks if the provers' measurements follow commutation/anti-commutation rules consistent with the honest measurements, namely $\tau^A_{W(a)}\tau^A_{W'(a')} \otimes I_B|\psi\rangle_{AB}$ is close to $(-1)^{|\{W_i \neq W'_i \text{ and } a_i = a'_i = 1\}|}\tau^A_{W'(a')}\tau^A_{W(a)} \otimes I_B|\psi\rangle_{AB}$.

The Consistency Test and Linearity Test are very simple and are described in Figure 1. For the Anti-commutation Test, we can use non-local games that allow the verifier to check that the provers share a constant number of EPR pairs and perform Pauli measurements on them. In this work we use the Magic Square game since there is a perfect quantum strategy for it.

The verifier performs the following steps, with probability $\frac{1}{3}$ each:

(A) Consistency test
- **a.** The verifier picks $W \in_R \{X, Z\}^n$ and $a \in \{0,1\}^n$.
- **b.** The verifier sends $W(a)$ to both provers.
- **c.** The verifier accepts iff the provers' answers are equal.

(B) Linearity test
- **a.** The verifier picks $W \in_R \{X, Z\}^t$ and $a, a' \in_R \{0,1\}^t$.
- **b.** The verifier sends $(W(a), W(a'))$ to P_1 and $W' \in_R \{W(a), W(a')\}$ to P_2.
- **c.** The verifier receives $b, b' \in \{\pm 1\}^t$ from P_1 and $c \in \{\pm 1\}^t$ from P_2.
- **d.** The verifier accepts iff $b = c$ when $W' = W(a)$ or $b' = c$ when $W' = W(a')$.

(C) Anti-commutation test
- **a.** The verifier makes the provers play Magic Square games in parallel with the t EPR pairs (see Section 2.2.1).

Figure 1 Pauli Braiding Test.

▶ **Theorem 2** (Theorem 14 of [27]). *Suppose $|\psi\rangle \in \mathcal{H}_A \otimes \mathcal{H}_B$ and $W(a) \in \mathrm{Obs}(\mathcal{H}_A)$, for $W \in \{X, Z\}^t$ and $a \in \{0,1\}^t$, specify a strategy for the players that has success probability at least $1-\varepsilon$ in the Pauli Braiding Test. Then there exist isometries $V_D : \mathcal{H}_D \to ((\mathbb{C}^2)^{\otimes t})_{D'} \otimes \hat{\mathcal{H}}_{\hat{D}}$, for $D \in \{A, B\}$, such that*

$$\left\| (V_A \otimes V_B)|\psi\rangle_{AB} - |\Phi_{00}\rangle^{\otimes t}_{A'B'} |\mathrm{AUX}\rangle_{\hat{A}\hat{B}} \right\|^2 = O(\sqrt{\varepsilon}),$$

and on expectation over $W \in \{X, Z\}^t$,

$$\mathop{\mathrm{E}}_{a \in \{0,1\}^t} \left\| \left(W(a) - V_A^\dagger (\sigma_W(a) \otimes I) V_A\right) \otimes I_B |\psi\rangle \right\|^2 = O(\sqrt{\varepsilon}).$$

Moreover, if the provers share $|\Phi_{00}\rangle^{\otimes t}_{A'B'}$ and measure with the observables $\bigotimes \sigma_{W_i}$ on question W, they pass the test with probability 1.

2.3 Local Hamiltonian problem

The Local Hamiltonian problem can be seen as the quantum analog of MAX-SAT problem. An instance for this problem consists in m Hermitian matrices $H_1, \ldots, H_m$, where each H_i acts non-trivially on at most most k qubits. For some parameters $\alpha, \beta \in \mathbb{R}$, $\alpha < \beta$, the Local Hamiltonian problem asks if there is a global state such that its energy in respect of $H = \frac{1}{m} \sum_{i \in [m]} H_i$ is at most α or all states have energy at least β. This problem was first proved to be QMA-complete for $k = 5$ and $\beta - \alpha \geq \frac{1}{\mathrm{poly}(n)}$ [19]. In this work, we are particularly interested in the version of LH where all the terms are tensor products of σ_X, σ_Z and σ_I.

▶ **Definition 3** (XZ Local Hamiltonian). *The XZ k-Local Hamiltonian problem, for $k \in \mathbb{Z}^+$ and parameters $\alpha, \beta \in [0,1]$ with $\alpha < \beta$, is the following promise problem. Let n be the number of qubits of a quantum system. The input is a sequence of $m(n)$ values $\gamma_1, ..., \gamma_{m(n)} \in [-1, 1]$ and $m(n)$ Hamiltonians $H_1, \ldots, H_{m(n)}$ where m is a polynomial in n, and for each $i \in [m(n)]$, H_i is of the form $\bigotimes_{j \in n} \sigma_{W_j} \in \{\sigma_X, \sigma_Z, \sigma_I\}^{\otimes n}$ with $|\{j | j \in [n] \text{ and } \sigma_{W_j} \neq \sigma_I\}| \leq k$. For $H \stackrel{\mathrm{def}}{=} \frac{1}{m(n)} \sum_{j=1}^{m(n)} \gamma_j H_j$, one of the following two conditions hold.*

Yes. *There exists a state $|\psi\rangle \in \mathbb{C}^{2^n}$ such that $\langle\psi|H|\psi\rangle \leq \alpha(n)$*

No. *For all states $|\psi\rangle \in \mathbb{C}^{2^n}$ it holds that $\langle\psi|H|\psi\rangle \geq \beta(n)$.*

Whenever the value of n is clear from the context, we call $\alpha(n)$, $\beta(n)$ and $m(n)$ by α, β and m. The XZ k-LH problem has been also proved QMA-complete [11, 18].

▶ **Lemma 4** (Lemma 22 of [18], Lemma 22 of [11]). *There exist $\alpha, \beta \in [0,1]$ satisfying $\beta - \alpha \geq \frac{1}{poly(n)}$ such that XZ k-Local Hamiltonian is QMA-complete, for some constant k.*

It is an open question if k-LH is QMA-complete for $\beta - \alpha = O(1)$ while maintaining k constant [2]. However, it is possible to achieve this gap at the cost of increasing the locality of the Hamiltonian [27].

▶ **Lemma 5** (Lemma 26 of [27]). *Let H be an n-qubit Hamiltonian with minimum energy $\lambda_0(H) \geq 0$ and such that $\|H\| \leq 1$. Let $\alpha, \beta \geq \frac{1}{poly(n)}$ and $\alpha < \beta$ for all n. Let H' be the following Hamiltonian on $(\beta - \alpha)^{-1} n$ qubits*

$$H' = \sigma_I^{\otimes na} - (\sigma_I^{\otimes n} - (H - a^{-1}\sigma_I^{\otimes n}))^{\otimes a}, \text{ where } a = (\beta - \alpha)^{-1}.$$

It follows that if $\lambda_0(H) \leq \alpha$ then $\lambda_0(H') \leq \frac{1}{2}$, while if $\lambda_0(H) \geq \beta$ then $\lambda_0(H') \geq 1$. Moreover if H is a XZ Hamiltonian, so is H'.

Finally, we define now non-local games for Local Hamiltonian problems.

▶ **Definition 6** (Non-local games for Hamiltonians). *A non-local game for the Local Hamiltonian problem consists in a reduction from a Hamiltonian H acting on n qubits to a non-local game $G(H)$ where a verifier plays against r provers, and for some parameters $\alpha, \beta, c, s \in [0,1]$, for $\alpha < \beta$ and $c > s$, the following holds.*
Completeness. *If $\lambda_0(H) \leq \alpha$, then $\omega^*(G(H)) \geq c$*
Soundness. *If $\lambda_0(H) \geq \beta$, then $\omega^*(G(H)) \leq s$.*

3 One-round two-prover game for Local Hamiltonian

In this section, we define our non-local game for Local Hamiltonian problem, proving Theorem 9. We start with a XZ Hamiltonian $H = \frac{1}{m}\sum_{l \in m} \gamma_l H_l$ acting on n qubits and $\alpha, \beta \in [0,1]$ with $\alpha < \beta$. We propose then the Hamiltonian Test $G(H)$, a non-local game based on H, whose maximum acceptance probability upper and lower bounds are tightly related to $\lambda_0(H)$. Based on $G(H)$, we show how to construct another non-local game $\tilde{G}(H)$ for which there exists some universal constant $\Delta > 0$ such that if $\lambda_0(H) \leq \alpha$, then $\omega^*(\tilde{G}(H)) \geq \frac{1}{2} + \Delta$, whereas if $\lambda_0(H) \geq \beta$, then $\omega^*(\tilde{G}(H)) \leq \frac{1}{2} - \Delta$. The techniques used to devise $G(H)$ and $\tilde{G}(H)$ are based on Ref. [18, 27].

We describe now the Hamiltonian Test $G(H)$, which is composed by the Pauli Braiding Test (PBT) (see Section 2.2) and the Energy Test (ET), which allows the verifier to estimate $\lambda_0(H)$. The provers are expected to share t EPR pairs and the first prover holds a copy of the groundstate of H. In ET, the verifier picks $l \in_R [m]$, $W \in_R \{X, Z\}^t$ and $e \in_R \{0,1\}^t$, and chooses $\mathcal{T}_1, ..., \mathcal{T}_n \in [t]$ such that $W(e)_{\mathcal{T}_i}$ matches the i-th Pauli observable of H_l. By setting $t = O(n \log n)$, it is possible to choose such positions for a random $W(e)$ with overwhelming probability. The verifier sends $\mathcal{T}_1, ..., \mathcal{T}_n$ to the first prover, who is supposed to teleport the groundstate of H through the EPR pairs in these positions. As in PBT, the verifier sends $W(e)$ to the second prover, who is supposed to measure his EPR halves with the corresponding observables. The values of $\mathcal{T}_1, ..., \mathcal{T}_n$ were chosen in a way that the first prover teleports the groundstate of H in the exact positions of the measurement according to H_l.

The verifier performs each of the following steps with probability $1-p$ and p, respectively:
(A) Pauli Braiding Test
(B) Energy Test
- **a.** The verifier picks $W \in_R \{X,Z\}^t$, $e \in_R \{0,1\}^t$ and $l \in_R [m]$
- **b.** The verifier picks positions $\mathcal{T}_1$, ...$\mathcal{T}_n$ such that $H_l = \bigotimes \sigma_{W(e)_{\mathcal{T}_i}}$.
- **c.** The verifier sends $\mathcal{T}_1, ..., \mathcal{T}_n$ to the first prover and $W(e)$ to the second prover.
- **d.** The first prover answers with $a, b \in \{0,1\}^n$ and the second prover with $c \in \{+1,-1\}^t$.
- **e.** Let $d \in \{-1,+1\}^n$ such that $d_i = (-1)^{a_i} c_{\mathcal{T}_i}$ if $W_{\mathcal{T}_i} = X$ and $d_i = (-1)^{b_i} c_{\mathcal{T}_i}$ if $W_{\mathcal{T}_i} = Z$.
- **f.** If $\prod_{i\in[n]} d_i \neq sign(\gamma_l)$, the verifier accepts.
- **g.** Otherwise, the verifier rejects with probability $|\gamma_l|$.

Figure 2 Hamiltonian Test $G(H)$ for a XZ Hamiltonian H.

With the outcomes of the teleportation measurements, the verifier can correct the output of the measurement of the second prover and estimate $\lambda_0(H)$. The full description of the game is presented in Figure 2.

We state now two auxiliary lemmas with lower and upper bounds on the maximum acceptance probability on $G(H)$.

▶ **Lemma 7.** *Let $H = \sum_{l\in[m]} \gamma_l H_l$ be a XZ Hamiltonian, let $G(H)$ be the Hamiltonian-self test for H, described in Figure 2, and*

$$\omega_h(H) \stackrel{\text{def}}{=} 1 - p\left(\frac{1}{2m}\sum_{l\in[m]} |\gamma_l| - \frac{1}{2}\lambda_0(H)\right).$$

If the provers use the honest strategy in PBT*, the maximum acceptance probability in $G(H)$ is $\omega_h(H)$. Moreover, this probability is achieved if the first prover behaves honestly in* ET*.*

▶ **Lemma 8.** *Let H, $G(H)$ and $\omega_h(H)$ be defined as Lemma 7. For every $\eta > 0$, there is some value of $p = O(\sqrt{\eta})$ such that $\omega^*(G(H)) \leq \omega_h(H) + \eta$.*

We defer the proof of these lemmas to Section 3.1 and we concentrate now in proving our main theorem.

▶ **Theorem 9.** *There exists a universal constant Δ such that the following holds. Let $H = \sum_{l\in m} \gamma_l H_l$ be XZ k-Local Hamiltonian acting on n qubits with parameters $\alpha, \beta \in (0,1)$, for $\beta > \alpha$. There exists one-round two-prover non-local game such that*
- *if $\lambda_0(H) \leq \alpha$, then the verifier accepts with probability at least $\frac{1}{2} + \Delta$; and*
- *if $\lambda_0(H) \geq \beta$, then the verifier accepts with probability at most $\frac{1}{2} - \Delta$.*

Moreover, each message is $\tilde{O}(n(\beta-\alpha)^{-1})$-bit long.

Proof. Lemma 5 states that from H we can construct a Hamiltonian H' such that

$$\lambda_0(H) \leq \alpha \Rightarrow \lambda_0(H') \leq \frac{1}{2} \text{ and } \lambda_0(H) \geq \beta \Rightarrow \lambda_0(H') \geq 1,$$

and $H' = \sum_{l\in m'} \gamma'_l H'_l$ is an instance of XZ Local Hamiltonian problem.

We now bound the maximum acceptance probability of the Hamiltonian Test on H', relating it to the groundstate energy of H. From Lemma 7 it follows that

$$\lambda_0(H) \leq \alpha \Rightarrow \omega^*(G(H')) \geq 1 - p\left(\frac{1}{2m}\sum_{l\in[m]} |\gamma'_l| - \frac{1}{4}\right) \stackrel{\text{def}}{=} c,$$

while from Lemma 8, for any $\eta > 0$ and some $p \leq C\sqrt{\eta}$, we have that

$$\lambda_0(H) \geq \beta \Rightarrow \omega^*(G(H')) \leq 1 - p\left(\frac{1}{2m}\sum_{l\in[m]}|\gamma_l'| - \frac{1}{2}\right) + \eta = c - \frac{C\sqrt{\eta}}{4} + \eta.$$

By choosing η to be a constant such that $\eta' \stackrel{\text{def}}{=} \frac{C\sqrt{\eta}}{4} - \eta > 0$, it follows that

$$\lambda_0(H) \leq \alpha \Rightarrow \omega^*(G(H')) \geq c \text{ and } \lambda_0(H) \geq \beta \Rightarrow \omega^*(G(H')) \leq c - \eta'.$$

We describe now the game $\tilde{G}(H)$ that achieves the completeness and soundness properties stated in the theorem. In this game, the verifier accepts with probability $\frac{1}{2} - \frac{2c-\eta'}{4}$, rejects with probability $\frac{2c-\eta'}{4}$ or play $G(H')$ with probability $\frac{1}{2}$. Within this new game, if $\lambda_0(H) \leq \alpha$ then $\omega^*(\tilde{G}(H')) \geq \frac{1}{2} + \frac{\eta'}{4}$, whereas when $\lambda_0(H) \geq \beta$, we have that $\omega^*(\tilde{G}(H')) \leq \frac{1}{2} - \frac{\eta'}{4}$. ◀

▶ **Corollary 10.** *There exists a protocol for verifiable delegation of quantum computation where a classical client communicates with two entangled servers in one round of classical communication.*

Proof. The corollary holds from composing the circuit-to-Hamiltonian construction (see the full version [16] of the paper for more details) with our non-local game. ◀

▶ Remark 11. The parameters of our delegation protocol allow us to use standard arguments in relativistic cryptography to replace the assumption that the provers do not communicate by the assumption that they can only communicate at most as fast as the speed of light. See the full version [16] of this paper for more details on this matter.

3.1 Proof of Lemmas 7 and 8

We start by proving Lemma 7, showing an upper bound on the acceptance probability if the provers are honest in PBT.

Proof of Lemma 7. Since PBT and ET are indistinguishable to the second prover, he also follows the honest strategy in ET and the acceptance probability in $G(H)$ depends uniquely in the strategy of the first prover in ET.

Let $a, b \in \{0,1\}^n$ be the answers of the first prover in ET and τ be the reduced state held by the second prover on the positions $\mathcal{T}_1, ..., \mathcal{T}_n$ of his EPR halves, after the teleportation.

For a fixed H_l, the verifier rejects with probability

$$\frac{|\gamma_l| + \gamma_l \mathbb{E}\left[\prod_{i\in n} d_i\right]}{2}. \tag{1}$$

We notice that measuring a qubit $|\phi\rangle$ in the Z-basis with outcome $f \in \{\pm 1\}$ is equivalent of considering the outcome $(-1)^g f$ when measuring $X^g Z^h |\phi\rangle$ in the same basis. An analog argument follows also for the X-basis. Therefore, by fixing the answers of the first prover, instead of considering that the second prover measured τ in respect of H_l with outcome c, we consider that he measured $\rho = \mathsf{Z}^b\mathsf{X}^a\tau\mathsf{X}^a\mathsf{Z}^b$ with respect to H_l with outcome d. In this case, by taking $\prod_{i\in n} d_i$ as the outcome of the measurement of H_l on ρ, and averaging over all $l \in [m]$, it follows from Equation (1) that the verifier rejects in ET with probability

$$\frac{1}{m}\sum_{l\in[m]}\frac{|\gamma_l| + \gamma_l \mathrm{Tr}(\rho H_l)}{2} = \frac{1}{2m}\sum_{l\in[m]}|\gamma_l| + \frac{1}{2}\mathrm{Tr}(\rho H),$$

and this value is minimized when ρ is the groundstate of H. In this case the overall acceptance probability in $G(H)$ is at most

$$1 - p\left(\frac{1}{2m}\sum_{l\in[m]}|\gamma_l| - \frac{1}{2}\lambda_0(H)\right) = \omega_h(H).$$

Finally, this acceptance probability is achieved if the first prover teleports the groundstate $|\psi\rangle$ of H and report the honest outcomes from the teleportation, since $\tau = \mathsf{X}^a\mathsf{Z}^b|\psi\rangle\langle\psi|\mathsf{Z}^b\mathsf{X}^a$ and $\rho = |\psi\rangle\langle\psi|$. ◀

We use now the self-testing of PBT to certify the measurements of the second prover in ET. In this way, we can bound the acceptance probability in $G(H)$ with Lemma 7 and prove Lemma 8.

Proof of Lemma 8. Let S be the strategy of the provers, which results in acceptance probabilities $1-\varepsilon$ in PBT and $1 - \frac{1}{2m}\sum_{l\in[m]}|\gamma_l| - \frac{1}{2}\lambda_0(H) + \delta$ in ET, for some ε and δ.

By Lemma 2, their strategy in PBT is $O(\sqrt{\varepsilon})$-close to the honest strategy, up to the local isometries V_A and V_B. Let S_h be the strategy where the provers follow the honest strategy in PBT and, for ET, the first prover performs the same operations of S, but considering the isometry V_A from Theorem 2. Since the measurements performed by the provers in S and S_h are $O(\sqrt{\varepsilon})$-close to each other, considering the isometries, the distributions of the corresponding transcripts have statistical distance at most $O(\sqrt{\varepsilon})$. Therefore, the provers following strategy S_h are accepted in ET with probability at least

$$1 - \frac{1}{2m}\sum_{l\in[m]}|\gamma_l| - \frac{1}{2}\lambda_0(H) + \delta - O(\sqrt{\varepsilon}).$$

Since in S_h the provers perform the honest strategy in PBT, it follows from Lemma 7 that

$$1 - \frac{1}{2m}\sum_{l\in[m]}|\gamma_l| - \frac{1}{2}\lambda_0(H) + \delta - O(\sqrt{\varepsilon}) \leq 1 - \frac{1}{2m}\sum_{l\in[m]}|\gamma_l| - \frac{1}{2}\lambda_0(H),$$

which implies that $\delta \leq C\sqrt{\varepsilon}$, for some constant C.

The original strategy S leads to acceptance probability at most

$$(1-p)(1-\varepsilon) + p\left(1 - \frac{1}{2m}\sum_{l\in[m]}|\gamma_l| - \frac{\lambda_0(H)}{2} + C\sqrt{\varepsilon}\right) = \omega_h(H) - (1-p)\varepsilon + pC\sqrt{\varepsilon}.$$

For any η, we can pick $p = \min\left\{\frac{\sqrt{\eta}}{D}, 1\right\}$, for $D \geq 2C$, and it follows that

$$pC\sqrt{\varepsilon} - (1-p)\varepsilon \leq \frac{2C\sqrt{\eta}\sqrt{\varepsilon}}{D} - \varepsilon \leq \sqrt{\eta}\sqrt{\varepsilon} - \varepsilon \leq \eta$$

and therefore the maximum acceptance probability is at most $\omega_h(H) + \eta$. ◀

References

1 Scott Aaronson, Alexandru Cojocaru, Alexandru Gheorghiu, and Elham Kashefi. On the implausibility of classical client blind quantum computing. *arXiv preprint*, 2017. arXiv:1704.08482.

2 Dorit Aharonov, Itai Arad, and Thomas Vidick. Guest column: the quantum PCP conjecture. *SIGACT News*, 44(2):47–79, 2013. URL: http://dblp.uni-trier.de/db/journals/sigact/sigact44.html#AharonovAV13.

3 Dorit Aharonov, Michael Ben-Or, Elad Eban, and Urmila Mahadev. Interactive proofs for quantum computations. *arXiv preprint*, 2017. arXiv:1704.04487.

4 Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof Verification and the Hardness of Approximation Problems. *J. ACM*, 45(3):501–555, 1998. doi:10.1145/278298.278306.

5 Sanjeev Arora and S Safra. Probabilistic Checking of Proofs: A New Characterization of NP. *jacm*, 45(1):70–122, 1998. doi:10.1145/273865.273901.

6 John S Bell. On the Einstein-Podolsky-Rosen Paradox. *Physics*, 1:195–200, 1964.

7 Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover Interactive Proofs: How to Remove Intractability Assumptions. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, STOC '88, pages 113–131. ACM, 1988. doi:10.1145/62212.62223.

8 Anne Broadbent. How to Verify a Quantum Computation, 2018.

9 Davide Castelvecchi. IBM's quantum cloud computer goes commercial. *Nature News*, 543(7644), 2017.

10 Andrea Coladangelo, Alex Grilo, Stacey Jeffery, and Thomas Vidick. Verifier-on-a-Leash: new schemes for verifiable delegated quantum computation, with quasilinear resources. *arXiv preprint*, 2017. arXiv:1708.07359.

11 Toby S Cubitt and Ashley Montanaro. Complexity Classification of Local Hamiltonian Problems. In *Proceedings of the 55th IEEE Annual Symposium on Foundations of Computer Science (FOCS '14)*, pages 120–129, 2014.

12 Irit Dinur. The PCP Theorem by Gap Amplification. *jacm*, 54(3), 2007.

13 Joseph F. Fitzsimons, Michal Hajdušek, and Tomoyuki Morimae. Post hoc Verification of Quantum Computation. *Phys. Rev. Lett.*, 120:040501, January 2018.

14 Joseph F. Fitzsimons and Elham Kashefi. Unconditionally verifiable blind quantum computation. *Physical Review A*, 96(012303), 2012.

15 Alexandru Gheorghiu, Elham Kashefi, and Petros Wallden. Robustness and device independence of verifiable blind quantum computing. *New Journal of Physics*, 17, 2015.

16 Alex B. Grilo. Relativistic verifiable delegation of quantum computation. *arXiv preprint*, 2017. arXiv:1711.09585.

17 Michal Hajdušek, Carlos A Pérez-Delgado, and Joseph F. Fitzsimons. Device-independent verifiable blind quantum computation. *arXiv preprint*, 2015. arXiv:1502.02563.

18 Zhengfeng Ji. Classical verification of quantum proofs. In *Proceedings of the Forty-eighth Annual ACM SIGACT Symposium on Theory of Computing (STOC 2016)*, pages 885–898, 2016.

19 Alexei Kitaev, A Shen, and M N Vyalyi. *Classical and quantum computation*. Graduate studies in mathematics. American mathematical society, Providence (R.I.), 2002. URL: http://opac.inria.fr/record=b1100148.

20 Urmila Mahadev. Classical Verification of Quantum Computations. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 259–267, 2018.

21 Dominic Mayers and Andrew Yao. Self testing quantum apparatus. *Quantum Information & Computation*, 4:273–286, 2004.

22 Matthew McKague. Interactive Proofs for BQP via Self-Tested Graph States. *Theory of Computing*, 12(3):1–42, 2016.

23 David Mermin. Simple unified form for the major no-hidden-variables theorems. *Physical Review Letters*, 65:3373–3376, 1990.

24 Ashely Montanaro. Quantum algorithms: an overview. *npj Quantum Information*, 2(15023), 2016.

25 Tomoyuki Morimae. Verification for measurement-only blind quantum computing. *Physical Review A*, 89, 2014.

26 Tomoyuki Morimae and Joseph F. Fitzsimons. Post hoc verification with a single prover. *arXiv preprint*, 2016. `arXiv:arXiv:1603.06046`.

27 Anand Natarajan and Thomas Vidick. A quantum linearity test for robustly verifying entanglement. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2017)*, pages 1003–1015, 2017.

28 Anand Natarajan and Thomas Vidick. Low-Degree Testing for Quantum States, and a Quantum Entangled Games PCP for QMA. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018*, pages 731–742, 2018.

29 Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge Series on Information and the Natural Sciences. Cambridge University Press, 2000.

30 Asher Peres. Incompatible results of quantum measurements. *Physics Letters A*, 151(3):107–108, 1990. `doi:10.1016/0375-9601(90)90172-K`.

31 Ran Raz. A Parallel Repetition Theorem. *sicomp*, 27(3):763–803, 1998.

32 Ben W Reichardt, Falk Unger, and Umesh Vazirani. Classical Command of Quantum Systems. *Nature*, 496:456–460, 2013.

33 Xingyao Wu, Jean-Daniel Bancal, Matthew McKague, and Valerio Scarani. Device-independent parallel self-testing of two singlets. *Physical Review A*, 93:62121, 2016.

Dismantlability, Connectedness, and Mixing in Relational Structures

Raimundo Briceño
School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel
raimundo@alumni.ubc.ca

Andrei A. Bulatov
School of Computing Science, Simon Fraser University, Canada
abulatov@sfu.ca

Víctor Dalmau
Department of Information and Communication Technologies, Universitat Pompeu Fabra, Barcelona, Spain
victor.dalmau@upf.edu

Benoît Larose
LACIM, Université du Québec a Montréal, Montréal, Canada
blarose@lacim.ca

Abstract

The Constraint Satisfaction Problem (CSP) and its counting counterpart appears under different guises in many areas of mathematics, computer science, statistical physics, and elsewhere. Its structural and algorithmic properties have demonstrated to play a crucial role in many of those applications. For instance, topological properties of the solution set such as connectedness is related to the hardness of CSPs over random structures. In approximate counting and statistical physics, where CSPs emerge in the form of spin systems, mixing properties and the uniqueness of Gibbs measures have been heavily exploited for approximating partition functions or the free energy of spin systems. Additionally, in the decision CSPs, structural properties of the relational structures involved – like, for example, dismantlability – and their logical characterizations have been instrumental for determining the complexity and other properties of the problem.

In spite of the great diversity of those features, there are some eerie similarities between them. These were observed and made more precise in the case of graph homomorphisms by Brightwell and Winkler, who showed that the structural property of dismantlability of the target graph, the connectedness of the set of homomorphisms, good mixing properties of the corresponding spin system, and the uniqueness of Gibbs measure are all equivalent. In this paper we go a step further and demonstrate similar connections for arbitrary CSPs. This requires much deeper understanding of dismantling and the structure of the solution space in the case of relational structures, and new refined concepts of mixing introduced by Briceño. In addition, we develop properties related to the study of valid extensions of a given partially defined homomorphism, an approach that turns out to be novel even in the graph case. We also add to the mix the combinatorial property of finite duality and its logic counterpart, FO-definability, studied by Larose, Loten, and Tardif.

2012 ACM Subject Classification Mathematics of computing → Paths and connectivity problems

Keywords and phrases relational structure, constraint satisfaction problem, homomorphism, mixing properties, Gibbs measure

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.29

Category Track A: Algorithms, Complexity and Games

Related Version A full version of this paper is available at `http://arxiv.org/abs/1901.04398`.

Funding This work was done in part while the first three authors were visiting the Simons Institute for the Theory of Computing at University of California, Berkeley.
Raimundo Briceño: The first author was supported by ERC Starting Grants 678520 and 676970.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 29; pp. 29:1–29:15

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

Andrei A. Bulatov: This work was supported by an NSERC Discovery grant.
Víctor Dalmau: The third author was supported by MICCIN grant TIN2016-76573-C2-1P and Maria de Maeztu Units of Excellence Programme MDM-2015-0502.
Benoît Larose: The fourth author was supported by an NSERC Discovery grant and FRQNT.

1 Introduction

The Constraint Satisfaction Problem (CSP) provides a powerful framework in a wide range of areas of mathematics, computer science, statistical physics, and elsewhere. The goal in a CSP is to find an assignment to variables from a given set that satisfies a number of given constraints. The counting version of the problem asks about the number of such assignments. The CSP however appears in different forms: as the standard one outlined above in AI and computer science [19], as the homomorphism problem in graph and model theory [22, 27], as conjunctive query evaluation in logic and database theory [30], as computing the partition function of a spin system in statistical physics [39] and related areas, like symbolic dynamics and coding [35, 37].

The CSP allows for many approaches of diverse nature, and every application field exploits some of its many facets: structural properties of constraints for complexity and algorithms, probabilistic properties and the topology of the solution space in Random CSP and random structures, mixing properties in statistical physics and dynamical systems, decay of correlations and the uniqueness of probabilistic measures in approximate counting, and homomorphic duality and logical characterizations in model theory. In [12], Brightwell and Winkler observed that some of these properties are actually closely related, at least in the simple case of graph homomorphisms. In this paper we take this research direction a step further by extending Brightwell and Winkler's results to the general CSP, and by refining and widening the range of the properties involved.

We start off with a brief introduction of the features of the CSP considered in this paper. Afterwards, we provide a detailed account of the necessary background and a description of our results.

Every CSP involves a set of variables and a domain, a set of possible values for the variables. Assumptions about these two sets differ in different areas. The most studied case in combinatorics and complexity theory is when both sets are finite. However, many interesting problems such as scheduling and temporal and spatial reasoning involve infinite domains; see also extensive literature on infinite CSPs (see, for example, [4] and the references therein). In other cases such as in statistical physics, it is natural to choose the set of variables to be infinite (a lattice, for example). Then, it is also natural to study probability distributions over such assignments – where *Gibbs measures* and the problem of their *(non-)uniqueness* appear naturally [25] – and also study quantities such as *entropy* and *free energy* [3, 7].

Following [22], CSPs can be formulated as the problem of deciding the existence of a homomorphism from a finite relational structure $\mathbb{G}$ to a target relational structure $\mathbb{H}$, where $\mathbb{G}$ and $\mathbb{H}$ encode the variables and the values of the CSP. The complexity of this problem, especially the case when $\mathbb{H}$ is a fixed finite relational structure, has received a lot of attention, culminating with the proof of the Feder-Vardi conjecture [14, 44]. In the present paper we focus as well on the case when $\mathbb{H}$ is finite, although our main focus is not algorithmic but rather structural. In particular, we are interested in studying the space $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ of homomorphism from $\mathbb{G}$ to $\mathbb{H}$. Furthermore, following [12] we consider homomorphisms from both finite and infinite relational structures $\mathbb{G}$ (although [12] only considers graphs), a flexibility that turns out to be useful to see different aspects of homomorphism spaces $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ that otherwise would be meaningless.

There is a vast literature concerning graph homomorphisms and their properties through the lenses of statistical physics [5, 20, 11, 39]. In this context, it is very common to encode a *spin system* as a pair of relational structures $\mathbb{G}$ and $\mathbb{H}$, where $\mathbb{G}$ contains a set of variables/particles and $\mathbb{H}$ contains the set of values/spins that each particle could take, imposing *hard constraints* on them, i.e., disregarding configurations of values that do not satisfy all the given constraints. In practical terms, all this reduces to study – individually and as a set – the maps from $\mathbb{G}$ to $\mathbb{H}$ that are homomorphisms. In particular, many important parameters of a spin system such as free energy and entropy can be learned from studying such a set of homomorphisms.

In [12], Brightwell and Winkler observed that many of the properties of graph homorphisms used in the above areas are equivalent to a single structural property of graphs, namely, dismantlability. In this paper we follow a similar approach and study properties of CSPs over general relational structures that we put into basically three categories: (1) dismantlability, (2) connectedness, and (3) mixing. Furthermore, as a consequence of our results, we established a connection with a fourth notion not initially contemplated in [12]: (4) finite duality.

Dismantlability

A graph is said to be dismantlable if it can be reduced to a single vertex removing vertices whose neighborhood is contained in the neighborhood of some other vertex. Such transformations are called *folds*, and they can be viewed as *retractions* of a very particular kind. Dismantlable graphs were introduced in [41], based on ideas already present in [29] in the context of lattices, and have been intensively studied in combinatorics. Distamantlability can be generalized in a natural way to relational structures. Indeed, some variants of this notion have been used in the study of CSPs. In particular, dismantlability has been applied in [15] to the problem of enumerating all solutions of $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ with polynomial delay. Also, it has played a major role in the study of CSPs definable in first-order logic [18, 34].

Connectedness

When $\mathbb{G}$ is finite, it is often useful to convert $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ into a graph and explore the connectivity properties of this graph. The set of edges of $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ can be defined in a variety of ways, usually the most suitable to the problem at hand. For example, it is common to say that two elements from $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ are close (and therefore adjacent in the graph) if the Hamming distance between them is smaller than a certain threshold. The particular case when this threshold is 1 has been intensively studied, motivated initially by the fact that the connectedness of the solution space for SAT problems over random instances is linked to the performance of standard satisfiability algorithms, such as WalkSAT or DPLL [1, 32]. This has given rise to a general framework called *reconfiguration* [28] (see also [40] for a recent survey) that goes way beyond homomorphisms. Work in this area encompasses both structural questions (under which conditions is $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ connected?) and algorithmic ones (what is the complexity of, deciding, given $\mathbb{G}$ and $\mathbb{H}$ as input, whether $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ is connected? Its diameter? The shortest path between two given members of $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$? Etc.). In the context of spin systems, the connectedness of $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ is related to processes that consists on periodically updating the spin of a single or a small set of particles (e.g., irreducibility of *Glauber dynamics*).

We also consider an alternative way to define adjacency in $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ via *links* as in [34]. This notion of adjacency is linked to the so-called *finite duality property*, which is another of the main themes of our work.

Mixing

Mixing properties have been intensively studied in statistical physics and related areas (see [2, 6, 8, 10, 17, 42]), and are usually applied when the set of particles in $\mathbb{G}$ is very large or infinite. In this case, it can be very useful to be able to "glue" together partial homomorphisms, provided their domains are far from each other. There are several properties that formalize this phenomenon and it is common to establish hierarchies among them. More concretely, given a metric in $\mathbb{G}$, it is natural to ask whether there exists some uniform gap such that for any two subsets A and B of particles sufficiently far apart (in terms of the gap), and for any pair of homomorphisms $\phi, \psi \in \mathrm{Hom}(\mathbb{G}, \mathbb{H})$, we can find a third one, γ, such that restriction of γ to A and B coincides with the restrictions of ϕ and ψ on A and B, respectively. On the contrary, whenever the information content of a given set (at least partially) determines the information content of another set (i.e., the possible values that the variables on it can take), no matter how far it is, such a phenomenon has been called *long range action* in previous work (e.g., see [13]).

Similar phenomena are used in the related area of approximate computing of partition functions, where many algorithms are based on decay of correlations between values of remote elements of $\mathbb{G}$, which allows for approximation of partition functions based only on local neighborhoods of variables.

Finite duality and logic characterizations

Homomorphism duality often helps to design a solution algorithm for a CSP or establish its useful properties. A graph (or relational structure) $\mathbb{H}$ is said to have *homomorphism duality* if there is a set $\mathcal{O}$ of graphs, called obstructions, such that a graph $\mathbb{G}$ has a homomorphism to $\mathbb{H}$ if and only if no graph from $\mathcal{O}$ is homomorphic to $\mathbb{G}$. Sometimes the set of obstructions is very simple, say, any bipartite graph has homomorphic duality, where $\mathcal{O}$ is the set of all odd cycles. If $\mathcal{O}$ can be chosen finite, we say that $\mathbb{H}$ has *finite duality*.

Homomorphism duality is closely related to another property of CSPs. Let $\mathcal{L}$ be a logic language such as first order, second order, etc. The problem of deciding homomorphisms to a relational structure $\mathbb{H}$ is said to be *expressible in* $\mathcal{L}$ if there is a formula Φ in the language $\mathcal{L}$ such that $\mathbb{G}$ has a homomorphism to $\mathbb{H}$ if and only if Φ is true on $\mathbb{G}$. It is known, for instance, that $\mathbb{H}$ has a set of obstructions consisting of relational structures of bounded treewidth if and only if the corresponding homomorphism problem is expressible in Datalog [22], or that $\mathbb{H}$ has finite duality if and only if the corresponding problem is expressible in first order logic [34] (see, for example, [16] for a survey on dualities for CSP).

Our results

The main result of this paper is Theorem 10, that shows, for a relational structure $\mathbb{H}$, the equivalence of the following three conditions: (A) $\mathbb{H}^2$ dismantles to a substructure of its diagonal, that is, the substructure of $\mathbb{H}^2$ induced by the set $\{(a, a) \mid a \in \mathbb{H}\}$; (B) for any $\mathbb{G}$, the homomorphism graph $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ is connected; and (C) for any relational structure $\mathbb{G}$, the graph $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ satisfies certain mixing properties. These results generalize the results from [12] to the case of general relational structures. Observe that the case of graphs considered in [12] does not fully reflect the richness of the theory behind our result.

As a byproduct of our results, we obtain two applications. On the one hand, we establish a link with *strong spatial mixing* (e.g., see [21]) and *topological strong spatial mixing* (introduced in [7]). These two last properties have played an important role in the development of deterministic approximate counting algorithms. In this paper we address the

following question: What fixed targets $\mathbb{H}$ are suitable for both of these properties to hold for any $\mathbb{G}$? On the other hand, we establish a connection with finite duality, which allows us to reprove the main theorem in [34]. We hope that our work opens the possibility of developing new counting techniques based on this approach in a very general setting. We stress that many of these results are new even in the graph case.

Due to space restrictions proofs are ommitted. They can be found at the full version [9].

2 Preliminaries

Let H be a countable (finite or denumerable) set and k a positive integer. The set of k-tuples over H is denoted by H^k. A (k-ary) **relation** R over H is a subset $R \subseteq H^k$. The elements of a relation R will be denoted in boldface, e.g., $\mathbf{a}$, $\mathbf{b}$, etc., and $\mathbf{a}[i]$ will denote the ith entry of $\mathbf{a}$ for $1 \leq i \leq k$.

Given another countable set G and a map $\phi : G \to H$, for a k-tuple $\mathbf{a}$ over G we shall use $\phi(\mathbf{a})$ to denote the k-tuple over H obtained after applying ϕ to $\mathbf{a}$ componentwise. If $V \subseteq G$, we will denote by $\phi|_V$ the restriction of ϕ to V. Furthermore, if ψ is another map with domain H, we shall use $\psi \circ \phi$ to denote the composition of ψ with ϕ, i.e., the map $x \mapsto \psi(\phi(x))$.

A **signature** τ is a finite collection of relation symbols R, each of them with an associated arity. For a given signature τ, a **relational structure** (with signature τ) – or simply, a **τ-structure** – $\mathbb{H}$ consists of a countable set H called the **universe** of $\mathbb{H}$ and a relation $R(\mathbb{H})$ for each $R \in \tau$, such that the arity of $R(\mathbb{H})$ equals that of R. We shall use the same capital letter to denote the universe of a τ-structure, e.g., H is the universe of $\mathbb{H}$. We will usually consider τ to be a fixed signature, and $\mathbb{G}$ and $\mathbb{H}$ to be τ-structures with universes G and H, respectively.

A relational structure is said to be **finite** if its universe is finite and **locally finite** if every element in its universe occurs only in a finite number of its tuples.

▶ Remark 1. A digraph $\mathbb{G}$ (with self-loops allowed) is a very particular case of a relational structure, where the signature τ consists of a unique relation symbol E of arity 2. Moreover, graphs correspond to the digraph case where $E(\mathbb{G})$ is a symmetric relation.

A map $\phi : G \to H$ is said to be a **homomorphism** from $\mathbb{G}$ to $\mathbb{H}$ if, for every relation symbol $R \in \tau$,

$$\mathbf{a} \in R(\mathbb{G}) \;\Rightarrow\; \phi(\mathbf{a}) \in R(\mathbb{H}).$$

We will denote by $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ the set of all homomorphisms from $\mathbb{G}$ to $\mathbb{H}$.

▶ **Example 2.** A particular example of CSPs that cannot be represented in the setting of Brightwell and Winkler (that is, as homomorphisms of graphs) is the case of **d-dimensional nearest-neighbor (n.n.) shifts of finite type (SFTs)**, a fundamental object in dynamical systems and probability (see [35, 36, 37]). Given a positive integer d, consider the signature $\tau = \{R_1, \ldots, R_d\}$, where R_i is a 2-ary relation for all $1 \leq i \leq d$. We consider two τ-structures $\mathbb{G}$ and $\mathbb{H}$. Here, $\mathbb{G}$ will be an infinite relational structure with universe $G = \mathbb{Z}^d$ and relations $R_i(\mathbb{G})$, $1 \leq i \leq d$, representing the usual d-dimensional hypercubic lattice and the adjacency of pairs of elements in it. On the other hand, $\mathbb{H}$ will be a finite relational structure with universe H and $R_i(\mathbb{H})$ representing pairs of "colors" from H that are allowed to be adjacent in the canonical ith direction of the lattice, $1 \leq i \leq d$. Then, $X = \mathrm{Hom}(\mathbb{G}, \mathbb{H})$ is known as a d-dimensional n.n. SFT, a set of colorings of $\mathbb{Z}^d$ with not necessarily *isotropic* adjacency rules (i.e., we do not need to have the same restrictions in every direction), and any such object can be represented in this way.

A relational structure $\mathbb{J}$ is a **substructure** of $\mathbb{H}$ if $J \subseteq H$ and, for every relation symbol $R \in \tau$, we have that $R(\mathbb{J}) \subseteq R(\mathbb{H})$. Furthermore, if for every k-ary $R \in \tau$, we have that $R(\mathbb{J}) = R(\mathbb{H}) \cap J^k$, then we say that $\mathbb{J}$ is the substructure of $\mathbb{H}$ **induced by** J. If $J \subseteq H$ and $\phi : H \to J$ is a homomorphism acting as the identity on J, then ϕ is said to be a **retraction**.

The **product** of $\mathbb{H}_1$ and $\mathbb{H}_2$, denoted $\mathbb{H}_1 \times \mathbb{H}_2$, is the τ-structure with universe $H_1 \times H_2$ where, for every k-ary relation symbol $R \in \tau$, we have that $R(\mathbb{H}_1 \times \mathbb{H}_2)$ consists of all tuples $((a_1, b_1), \ldots, (a_k, b_k))$ with $(a_1, \ldots, a_k) \in R(\mathbb{H}_1)$ and $(b_1, \ldots, b_k) \in R(\mathbb{H}_2)$. We shall denote by $\mathbb{H}^2$ the product $\mathbb{H} \times \mathbb{H}$. The **projections** $\pi_1, \pi_2 : H^2 \to H$ are the maps $(a, b) \mapsto a$ and $(a, b) \mapsto b$, respectively, for $(a, b) \in H^2$. An element (a, b) of H^2 is *diagonal* if $a = b$. The **diagonal set** of H^2, denoted $\Delta(H^2)$, is the set of its diagonal elements. Similarly, the **diagonal structure** of $\mathbb{H}^2$, denoted $\Delta(\mathbb{H}^2)$, is the substructure of $\mathbb{H}^2$ induced by $\Delta(H^2)$. A substructure $\mathbb{K}$ of $\mathbb{H}^2$ is **symmetric** whenever $(a, b) \in K$ if and only if $(b, a) \in K$. Notice that $\mathbb{H}^2$ is always symmetric.

In this paper, we will study properties of $\mathbb{H}$ and how they relate to other properties of $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ for arbitrary $\mathbb{G}$. We mainly consider three families of properties, namely, *dismantling* of $\mathbb{H}$, connectedness of some particular *graphs with vertex set* $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$, and *mixing properties* of $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$.

2.1 Dismantling

Let $\mathbb{H}$ be a τ-structure and let a, b be elements in its universe H. We say that b **dominates** a (in $\mathbb{H}$) if for every k-ary $R \in \tau$, any $i \in \{1, \ldots, k\}$, and any $(a_1, \ldots, a_k) \in R(\mathbb{H})$ with $a_i = a$, we also have that $(a_1, \ldots, a_{i-1}, b, a_{i+1}, \ldots, a_k) \in R(\mathbb{H})$. Additionally, if $a \neq b$, then we say that a is **dominated** (in $\mathbb{H}$).

A sequence of τ-structures $\mathbb{J}_0, \ldots, \mathbb{J}_\ell$ is a **dismantling sequence** if for every $0 \leq j < \ell$ there exist $a_j, b_j \in J_j$ such that b_j dominates a_j in $\mathbb{J}_j$, and $\mathbb{J}_{j+1}$ is the substructure of $\mathbb{J}_j$ induced by $J_j \setminus \{a_j\}$. In this case, we say that $\mathbb{J}_0$ **dismantles to** $\mathbb{J}_\ell$. We can alternatively denote a dismantling sequence by giving the initial relational structure $\mathbb{J}_0$ and the sequence of elements $a_0, \ldots, a_{\ell-1}$. We say that $\mathbb{H}$ is **dismantlable** if it dismantles to a τ-structure such that its universe is a singleton.

Note that for every $0 \leq j < \ell$ there is a natural retraction r_j from $\mathbb{J}_j$ to $\mathbb{J}_{j+1}$, where r_j maps a_j to b_j and acts as the identity elsewhere. We call such retractions a **fold**. By successive composition, one can define a retraction (namely, $r_{j'-1} \circ \cdots \circ r_j$) from $\mathbb{J}_j$ to $\mathbb{J}_{j'}$ for every $j \leq j'$.

It is well known that if $\mathbb{H}$ dismantles to some substructure $\mathbb{K}$, then this dismantling can be found in a greedy manner. Formally,

▶ **Lemma 3** ([34, Lemma 5.1]). *If $\mathbb{H}$ dismantles to $\mathbb{K}$ and $a \in H \setminus K$ is dominated in $\mathbb{H}$, then the substructure of $\mathbb{H}$ induced by $H \setminus \{a\}$ dismantles to $\mathbb{K}$.*

Let $J \subseteq H$. We say that $\mathbb{H}$ is **J-non-foldable** if every dominated element in $\mathbb{H}$ belongs to J.

2.2 Walks in relational structures

We define a **walk** w in a τ-structure $\mathbb{H}$ to be a sequence

$$a_0, i_1, (R_1, \mathbf{a}_1), j_1, a_1, \ldots, a_{n-1}, i_n, (R_n, \mathbf{a}_n), j_n, a_n$$

for some $n \geq 0$, such that, for all $1 \leq \ell \leq n$,

- $R_\ell \in \tau$, $\mathbf{a}_\ell \in R_\ell(\mathbb{H})$, $i_\ell \neq j_\ell$, and
- $a_{\ell-1} = \mathbf{a}_\ell[i_\ell]$ and $a_\ell = \mathbf{a}_\ell[j_\ell]$.

In this case, we will say that w **joins** a_0 (the **starting point**) and a_n (the **ending point**), and that the **length** of the walk w is n. Notice that if a walk w joins a_0 and a_n, then there is another walk w' that joins a_n and a_0 obtained by just reversing the order of indices. The **distance** $\mathrm{dist}(a,b)$ between two elements $a,b \in H$ is defined to be the smallest length among all the walks w that join a and b. The distance $\mathrm{dist}(V,W)$ between sets $V,W \subseteq H$ is defined to be the minimum distance between an element from V and an element from W.

Note that the definition of walk above coincides with the standard definition of walk when $\mathbb{H}$ is a graph. However, in the case of graphs it will be convenient to describe the walk merely as the list $a_0,\dots,a_n$ of its nodes, as usual.

A τ-structure $\mathbb{H}$ is **connected** if there is a walk joining any pair of elements of its universe H and a **connected component** is any induced substructure that is connected and maximal in the sense of inclusion. A walk w is a **circuit** if $n>0$, the starting and ending points of w coincide, and for all $1 \le \ell < \ell' \le n$, we have that $(R_\ell, \mathbf{a}_\ell) \neq (R_{\ell'}, \mathbf{a}_{\ell'})$. A τ-structure $\mathbb{T}$ is a **τ-forest** if it has no circuits. If, additionally, it is connected then it is a **τ-tree**. Usually, τ-trees are defined using the notion of incidence multigraph (see for example [34]). It is easy to verify that the definition given here is equivalent.

2.3 Forest of walks

Given a τ-structure $\mathbb{H}$, we proceed to define a new τ-structure $\mathbb{T}_\mathbb{H}$. The universe $T_\mathbb{H}$ of $\mathbb{T}_\mathbb{H}$ consists of all the walks w in $\mathbb{H}$. For a k-ary $R \in \tau$, we define $R(\mathbb{T}_\mathbb{H})$ as follows: for all $\mathbf{a}=(a_1,\dots,a_k) \in R(\mathbb{H})$, for all $1 \le i \le k$, and for all walks w ending in a_i, we include in $R(\mathbb{T}_\mathbb{H})$ the tuple $(w_1,\dots,w_{i-1},w,w_{i+1},\dots,w_k)$, where w_j, $j \neq i$, is the walk obtained from w by extending it with $i,(R,\mathbf{a}),j,a_j$.

We note that $\mathbb{T}_\mathbb{H}$ does not have circuits and has exactly $|H|$ connected components, i.e., $|H|$ τ-trees. It is easy to check that for every substructure $\mathbb{I}$ of $\mathbb{H}$, the τ-structure $\mathbb{T}_\mathbb{I}$ is a substructure of $\mathbb{T}_\mathbb{H}$.

▶ Remark 4. If $\mathbb{H}$ is connected and we consider a slight modification of this previous definition, where the walks are asked to be non-backtracking (i.e., for every $1 \le \ell < n$, we have that either $i_\ell \neq j_{\ell+1}$, or $j_\ell \neq i_{\ell+1}$, or $(R_\ell,\mathbf{x}_\ell) \neq (R_{\ell+1},\mathbf{x}_{\ell+1})$), then we obtain that each connected component of the resulting τ-structure corresponds to the *universal covering tree* of $\mathbb{H}$ [31, 33] (in particular, they are all the same up to isomorphism).

Note that, by construction, the map $\rho_\mathbb{H} : T_\mathbb{H} \to H$ that sends every walk w in $T_\mathbb{H}$ to its ending point, that from now on we refer as the **label map**, defines a homomorphism from $\mathbb{T}_\mathbb{H}$ to $\mathbb{H}$. Furthermore,

▶ **Lemma 5.** *Assume that $\mathbb{H}$ is J-non-foldable for some $J \subseteq H$ and let U be a cofinite subset of $T_\mathbb{H}$ containing $\rho_\mathbb{H}^{-1}(J)$. Then, every homomorphism in $\mathrm{Hom}(\mathbb{T}_\mathbb{H},\mathbb{H})$ that agrees with $\rho_\mathbb{H}$ in U is identical to $\rho_\mathbb{H}$.*

Proof. Given $n \ge 0$, let W_n be the set of walks of length at least n in $\mathbb{H}$ (notice that $W_n \subseteq W_{n-1}$ and $W_0 = T_\mathbb{H}$). We shall show that any $\rho' \in \mathrm{Hom}(\mathbb{T}_\mathbb{H},\mathbb{H})$ that agrees with $\rho_\mathbb{H}$ in $W_n \cup \rho_\mathbb{H}^{-1}(J)$ for arbitrary n also agrees with $\rho_\mathbb{H}$ in W_{n-1} . Let w be any walk of length $n-1$ and let a be its ending point. We first show that $\rho'(w)$ dominates a in $\mathbb{H}$. Indeed, let $R \in \tau$ and let $\mathbf{a}=(a_1,\dots,a_k) \in R(\mathbb{H})$, where a appears, say, in the ith coordinate. By construction, $R(\mathbb{T}_\mathbb{H})$ contains the tuple $\mathbf{w}=(w_1,\dots,w_{i-1},w,w_{i+1},\dots,w_k)$, where for every $j \neq i$, w_j is obtained by concatenating $i,(R,\mathbf{x}),j,a_j$ at the end of w. Since w_j has length n for every $j \neq i$, it follows by assumption that $\rho'(w_j)=a_j$. That is, $\rho'(\mathbf{w})$ (which must be a tuple in $R(\mathbb{H})$) is obtained by replacing, in $\mathbf{a}$, a_i by $\rho'(w)$.

Hence, we have shown that $\rho'(w)$ dominates a in $\mathbb{H}$. Since $\mathbb{H}$ is J-non-foldable it follows that either $\rho'(w) = a$ (and, hence, $\rho'(w) = \rho_{\mathbb{H}}(w)$) or $a \in J$ (and, hence, $\rho'(w) = \rho_{\mathbb{H}}(w)$ since $w \in \rho_{\mathbb{H}}^{-1}(J)$). To conclude the proof it is only necessary to observe that, since U is a cofinite set containing $\rho_{\mathbb{H}}^{-1}(J)$, it follows that any homomorphism that agrees with $\rho_{\mathbb{H}}$ in U, agrees as well in $W_n \cup \rho_{\mathbb{H}}^{-1}(J)$ for sufficiently large n. ◀

2.4 Graphs of homomorphisms

Let $\mathbb{G}$ and $\mathbb{H}$ be τ-structures and suppose that $\mathbb{H}$ is finite. We define two different kinds of graphs with vertex set $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$. The first notion has been heavily studied, from an algorithmic perspective, in the context of the so-called CSP reconfiguration problem (see [26] and the references therein) and, in the special cases when $\mathbb{G}$ and $\mathbb{H}$ are graphs also from an structural point of view [12]. We define $C(\mathbb{G}, \mathbb{H})$ as the (reflexive) graph with vertex set $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ such that for every $\phi, \psi \in \mathrm{Hom}(\mathbb{G}, \mathbb{H})$, ϕ and ψ are adjacent if and only if ϕ and ψ differ in at most one value, i.e., there exists at most one $x \in G$ such that $\phi(x) \neq \psi(x)$. More generally, for any $n \geq 1$ we can define $C_n(\mathbb{G}, \mathbb{H})$ on $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ by declaring ϕ and ψ adjacent if they differ in at most n values (in particular, $C(\mathbb{G}, \mathbb{H}) = C_1(\mathbb{G}, \mathbb{H})$).

A second notion of graph of homomorphisms appears in [34] and uses the notion of *links*. The **1-link** $\mathbb{L}$ (with signature τ) is the τ-structure with universe $\{0, 1\}$, where $R(\mathbb{L}) = \{0, 1\}^k$ for every k-ary $R \in \tau$. Define a (di)graph $L(\mathbb{G}, \mathbb{H})$ with vertex set $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ as follows: set $\phi \to \psi$ – i.e., a directed edge starting from ϕ and ending in ψ – if for any k-ary $R \in \tau$ and any $(x_1, \ldots, x_k) \in R(\mathbb{G})$, we have that $(\gamma_1(x_1), \ldots, \gamma_k(x_k)) \in R(\mathbb{H})$ whenever $\gamma_1, \ldots, \gamma_k \in \{\phi, \psi\}$. Alternatively, one can say that there ϕ and ψ are joined by a directed edge if there exists a homomorphism from $\mathbb{L}$ to $\mathbb{H}^{\mathbb{G}}$, the *$\mathbb{H}$th power of* $\mathbb{G}$ (see [34, Section 5.2]), mapping 0 to ϕ and 1 to ψ. Notice that the symmetry in the definition of 1-link implies that $L(\mathbb{G}, \mathbb{H})$ is, in fact, an undirected graph.

Clearly, $C_n(\mathbb{G}, \mathbb{H})$ is a subgraph of $C_{n+1}(\mathbb{G}, \mathbb{H})$. In contrast, $C_n(\mathbb{G}, \mathbb{H})$ and $L(\mathbb{G}, \mathbb{H})$ are not included in one another in general.

Note that there is a one-to-one correspondence between the elements in $\mathrm{Hom}(\mathbb{L} \times \mathbb{G}, \mathbb{H})$ and the edges of $L(\mathbb{G}, \mathbb{H})$. More generally, for $\ell \geq 1$ we define the **ℓ-link** $\mathbb{L}_\ell$ (with signature τ) as the τ-structure with universe $\{0, 1, \ldots, \ell\}$, where $R(\mathbb{L}_\ell) = \cup_{i=0}^{\ell-1} \{i, i+1\}^k$, for every k-ary $R \in \tau$. In other words, the ℓ-link is a sequence of 1-links with their endpoints identified. Then the following result is immediate:

▶ **Lemma 6.** *For every map $\phi : \{0, 1, \ldots, \ell\} \times G \to H$ and every $1 \leq i \leq \ell$, let $\phi(i) : G \to H$ be the map defined by $\phi(i)(x) \mapsto \phi(i, x)$ for $x \in G$. Then, $\phi \in \mathrm{Hom}(\mathbb{L}_\ell \times \mathbb{G}, \mathbb{H})$ if and only if $\phi(0), \ldots, \phi(\ell)$ is a walk in $L(\mathbb{G}, \mathbb{H})$.*

2.5 Mixing properties

Given τ-structure $\mathbb{G}$ and $\mathbb{H}$, it is useful to study properties in $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ that allow us to *glue together* partially defined homomorphisms. This kind of properties are usually referred in the literature as *mixing properties* (e.g., see [8, 10]).

A natural mixing property is *irreducibility*. We say that $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ is **(V, W)-irreducible** for $V, W \subseteq G$, if for every $\phi, \psi \in \mathrm{Hom}(\mathbb{G}, \mathbb{H})$, there exists a map $\gamma \in \mathrm{Hom}(\mathbb{G}, \mathbb{H})$ that agrees with ϕ on V and agrees with ψ on W.

Given $g \geq 0$, we say that $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ is **strongly irreducible with gap g** if for every V, W such that $\mathrm{dist}(V, W) \geq g$ and for all $\phi, \psi \in \mathrm{Hom}(\mathbb{G}, \mathbb{H})$, there exists $\gamma \in \mathrm{Hom}(\mathbb{G}, \mathbb{H})$ that agrees with ϕ on V and agrees with ψ on W. We say that $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ is **strongly irreducible** if it is strongly irreducible with gap g for some g.

A strengthening of strong irreducibility is the following property, introduced in [7]. Given $g \geq 0$, we say that $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ is **topologically strong spatial mixing (TSSM) with gap** g if for every $V, W, S \subseteq G$ such that $\mathrm{dist}(V, W) \geq g$ and for all $\phi, \psi \in \mathrm{Hom}(\mathbb{G}, \mathbb{H})$ that agree on S, there exists $\gamma \in \mathrm{Hom}(\mathbb{G}, \mathbb{H})$ that agrees with ϕ on $V \cup S$ and agrees with ψ on $S \cup W$. We say that $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ is **topologically strong spatial mixing** if it is TSSM with gap g for some g.

Clearly, $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ is TSSM only if $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ is strongly irreducible but not viceversa (see [7, 8] for some counterexamples).

An antithesis of having good mixing properties is the existence of configurations which are *frozen*. We say that $\phi \in \mathrm{Hom}(\mathbb{G}, \mathbb{H})$ is a **frozen configuration** if for any cofinite set $U \subseteq G$, the only homomorphism $\psi \in \mathrm{Hom}(\mathbb{G}, \mathbb{H})$ such that $\psi|_U = \phi|_U$ is $\psi = \phi$ itself.

3 Dismantlability, Connectivity, and Irreducibility

In this section we present our main theorem, which characterizes in several ways a special class of relational structures. This theorem generalizes some of the equivalences characterizing *dismantlable graphs* that appear in [12, Theorem 4.1] – which were developed only for the case of graphs – to arbitrary relational structures.

3.1 The case of graphs

The following theorem is a rephrasing of the equivalences that appear in [12, Theorem 4.1] which are relevant to us. We will use this as a prototypical example of the kind of results that we are aiming for, where we split the properties in 3 main categories (A) dismantlability, (B) connectedness, and (C) mixing.

▶ **Theorem 7** ([12, Theorem 4.1]). *Let $\mathbb{H}$ be a graph. The following are equivalent:*

(A) *$\mathbb{H}$ is dismantlable;*

(B) *$C(\mathbb{G}, \mathbb{H})$ is connected for every locally finite graph $\mathbb{G}$;*

(C) *there exists $g \geq 0$ such that $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ is strongly irreducible with gap g for every graph $\mathbb{G}$.*

▶ **Lemma 8.** *A graph $\mathbb{H}$ is dismantlable if and only if $\mathbb{H}^2$ dismantles to a substructure of its diagonal.*

Proof. This follows from our own results. In Theorem 10, we prove that, for a finite τ-structure $\mathbb{H}$, we have that $\mathbb{H}^2$ dismantles to a substructure of its diagonal if and only if there exists $g \geq 0$ such that $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$ is strongly irreducible with gap g for all τ-structures $\mathbb{G}$. In particular, this applies if $\tau = \{E\}$, the usual binary relation of adjacency in graphs. Therefore, by Theorem 7, these two properties are also equivalent to $\mathbb{H}$ being dismantlable, and we conclude. ◀

In other words, thanks to Lemma 8, at least in the realm of graphs, we can freely replace "dismantlable" by "the square dismantles to a substructure of its diagonal", which will be the relevant class of general relational structures in this work.

▶ Remark 9. It is important to notice that the equivalence between "dismantlable" and "the square dismantles to a substructure of its diagonal" is not true for general relational structures. For example, given $\tau = \{R\}$ for R a binary relation symbol, we can take $\mathbb{H}$ such that $H = \{0, 1\}$ and $R(\mathbb{H}) = \{(0, 1)\}$. Then, $\mathbb{H}$ is not dismantlable, but $\mathbb{H}^2$ dismantles to its diagonal.

3.2 Main Theorem

The following theorem shows that different dismantling, connectedness, and mixing notions are equivalent. It can be seen as a generalization of Theorem 7 to relational structures.

▶ **Theorem 10.** *Let $\mathbb{H}$ be a finite τ-structure with universe H. Then the following are equivalent:*

(A1s) *$\mathbb{H}$ dismantles to a structure $\mathbb{I}$ such that $\mathbb{I}^2$ dismantles to its diagonal;*

(A2s) *$\mathbb{H}^2$ dismantles to a substructure of its diagonal;*

(B1s) *$C(\mathbb{G},\mathbb{H})$ is connected for every locally finite τ-structure $\mathbb{G}$;*

(B2s) *there exists some $n \geq 1$ such that $C_n(\mathbb{G},\mathbb{H})$ is connected for every finite τ-structure $\mathbb{G}$;*

(B3s) *$C(\mathbb{L}\times\mathbb{H}^2,\mathbb{H})$ is connected;*

(B4s) *$L(\mathbb{G},\mathbb{H})$ is connected for every finite τ-structure $\mathbb{G}$;*

(B5s) *the projections π_1 and π_2 are connected in $L(\mathbb{H}^2,\mathbb{H})$;*

(C1s) *there exists $g \geq 0$, such that $\mathrm{Hom}(\mathbb{G},\mathbb{H})$ is strongly irreducible with parameter g for every τ-structure $\mathbb{G}$; and*

(C2s) *there exists $g \geq 0$, such that $\mathrm{Hom}(\mathbb{T}_{\mathbb{H}^2},\mathbb{H})$ is $(\{x\},W)$-mixing with parameter g, for all $x \in T_{\mathbb{H}^2}$ and $W \subseteq T_{\mathbb{H}^2}$.*

The proof of Theorem 10 can be found in the full version. Indeed, we prove a more general version of it which will allow us to derive the applications contained in the rest of the paper. Although due to space restrictions we cannot state the theorem in its more general form we believe is interesting in its own. In particular, it is motivated by the fact that, sometimes, it is natural – particularly in the context of statistical physics – to work by forcing a certain subset of particles to take each of them a particular spin and work with the remaining ones. For example, this is a common scenario when the particles in the boundary of a given set in a lattice are fixed to take particular spins and we want to study the distribution of spins in the interior of the set, conditioned on such boundary configuration. These ideas inspired the most general version, which can be regarded as the study of *boundary long range actions*, i.e., long range action phenomena where some boundary configuration is fixed, very similar to the concept of *boundary phase transition* in relation to phase transitions (e.g., see [38]).

4 First application: Gibbs measures and mixing conditions

4.1 Basic definitions

Given a finite τ-structure $\mathbb{H}$ with universe H, a **weight function** for $\mathbb{H}$ is a map $\lambda : H \to \mathbb{R}^+$.

Let $\mathbb{G}$ be a locally finite τ-structure. If $V \subseteq G$ is a finite set and $\phi \in \mathrm{Hom}(\mathbb{G},\mathbb{H})$, we define $\mathbb{P}_{V,\phi}$ to be the probability measure on $\mathrm{Hom}(\mathbb{G},\mathbb{H})$ given by

$$\mathbb{P}_{V,\phi}(\{\psi\}) := \begin{cases} Z_{V,\phi}(\lambda)^{-1}\prod_{x\in V}\lambda(\psi(x)) & \text{if } \psi|_V\, \phi|_{G\setminus V} \in \mathrm{Hom}(\mathbb{G},\mathbb{H}), \\ 0 & \text{otherwise,} \end{cases}$$

for $\psi \in \mathrm{Hom}(\mathbb{G},\mathbb{H})$, where $\psi|_V\, \phi|_{G\setminus V}$ is the map that coincides with ψ in V and with ϕ in $G \setminus V$, and $Z_{V,\phi}(\lambda)$ is a normalization constant – the *partition function* – defined as

$$Z_{V,\phi}(\lambda) := \sum_{\substack{\psi\in\mathrm{Hom}(\mathbb{G},\mathbb{H}) \\ \psi|_V\,\phi|_{G\setminus V}\in\mathrm{Hom}(\mathbb{G},\mathbb{H})}} \prod_{x\in V}\lambda(\psi(x)).$$

We will call the collection of probability measures $\{\mathbb{P}_{V,\phi}\}$, the **Gibbs** $(\mathbb{G},\mathbb{H},\lambda)$**-specification**. The **boundary** of a set $V \subseteq G$, denoted by ∂V, is defined as the set of elements in G at distance exactly 1 from V. Notice that $\mathbb{P}_{V,\phi}$ depends exclusively on $\phi|_{\partial V}$. Now, consider events of the form

$$A(\phi, V) = \{\psi \in \mathrm{Hom}(\mathbb{G},\mathbb{H}) : \psi|_V = \phi|_V\}.$$

Next, consider the σ-algebra $\mathcal{F}$ generated by all events of the form $A(\phi, V)$ for V finite, and define $\mathcal{M}(\mathbb{G},\mathbb{H})$ to be the set of probability measures on $(\mathrm{Hom}(\mathbb{G},\mathbb{H}),\mathcal{F})$.

A measure $\mu \in \mathcal{M}(\mathbb{G},\mathbb{H})$ is a **Gibbs measure** for the Gibbs $(\mathbb{G},\mathbb{H},\lambda)$-specification if for any finite $V \subseteq G$ and for all $\phi_1 \in \mathrm{Hom}(\mathbb{G},\mathbb{H})$,

$$\mu\left(A(\phi_1, V) | A(\phi_2, G \setminus V)\right) = \mathbb{P}_{V,\phi_2}\left(\{\phi_1\}\right) \text{ for } \mu\text{-a.e. } \phi_2 \in \mathrm{Hom}(\mathbb{G},\mathbb{H}).$$

In other words, the probability distribution of a random ϕ_1 inside a finite V conditioned on its values outside V to coincide with those of ϕ_2, depends only on the values of $\phi_1|_V$ and on the boundary, $\phi_2|_{\partial V}$. Furthermore, the conditional distribution is the same as for $\mathbb{P}_{V,\phi_2}$ (see also [12, Definition 2.1]).

If $\mathrm{Hom}(\mathbb{G},\mathbb{H}) \neq \emptyset$, then there always exists at least one Gibbs measure [25, Chapter 4]. A fundamental question in statistical physics is whether there exists a unique Gibbs measure or multiple for a given Gibbs $(\mathbb{G},\mathbb{H},\lambda)$-specification.

4.2 Non-uniqueness and spatial mixing properties

In [12], it is shown that if $\mathbb{H}$ is a graph and it is dismantlable (or equivalently, by Lemma 8, its square dismantles to a subgraph of its diagonal), then, for any locally finite graph $\mathbb{G}$, there exists some λ such that there is a unique Gibbs measure [12, Theorem 7.2]. Conversely, in [12] it is also proven that if $\mathbb{H}$ is a non-dismantlable graph, then there exists $\mathbb{G}$ such that for any λ there exists multiple Gibbs measures [12, Theorem 8.2].

Here, following a similar path, we show that when extending this question to arbitrary relational structures, the first implication does not remain true in general, but the second still holds. More exactly,

▶ **Proposition 11.** *There exists a finite τ-structure $\mathbb{H}$ such that $\mathbb{H}^2$ dismantles to a substructure of its diagonal and a locally finite τ-structure $\mathbb{G}$ such that for any λ there exists multiple Gibbs measures for the Gibbs $(\mathbb{G},\mathbb{H},\lambda)$-specification. Moreover, $\mathbb{H}$ can be chosen so that $\mathbb{H}^2$ dismantles to its full diagonal $\Delta(\mathbb{H}^2)$.*

▶ **Proposition 12.** *Let $\mathbb{H}$ be a finite τ-structure. If $\mathbb{H}^2$ does not dismantle to a substructure of $\Delta(\mathbb{H}^2)$, then there exists a locally finite τ-structure $\mathbb{G}$ such that for any λ there exists multiple Gibbs measures for the Gibbs $(\mathbb{G},\mathbb{H},\lambda)$-specification.*

In what follows we introduce some *spatial mixing* properties related to our results.

▶ **Definition 13.** *Given $J \subseteq H$, we say that a Gibbs $(\mathbb{G},\mathbb{H},\lambda)$-specification satisfies* ***spatial J-mixing (J-SM)*** *if there exists constants $C, \alpha > 0$ such that for all $\phi_1, \phi_2 \in \mathrm{Hom}(\mathbb{G},\mathbb{H})$, for all finite $V \subseteq G$, and for all $x \in V$ and $a \in H$,*

$$|\mathbb{P}_{V,\phi_1}(\{\psi(x) = a\}) - \mathbb{P}_{V,\phi_2}(\{\psi(x) = a\})| \leq C \cdot \exp(-\alpha \cdot \mathrm{dist}(x, D_V^J(\phi_1,\phi_2))), \tag{1}$$

where

$$D_V^J(\phi_1,\phi_2) = \{x \in \partial V : (\phi_1(x),\phi_2(x)) \in H^2 \setminus \Delta(J^2)\}$$

and $\{\psi(x) = a\}$ refers to the event that a random ψ takes the value a at x.

The definition of J-SM unifies and interpolates two well-known properties. If $J = \emptyset$, then $D_V^{\emptyset}(\phi_1, \phi_2) = \partial V$ and Eq. (1) corresponds to the definition of **weak spatial mixing (WSM)**, i.e., $\emptyset$-SM. On the other hand, if $J = H$, then $D_V^{H}(\phi_1, \phi_2) = \{x \in \partial V : \phi_1(x) \neq \phi_2(x)\}$ and Eq. (1) corresponds to the definition of **strong spatial mixing (SSM)**, i.e., H-SM.

In general, spatial mixing properties are forms of *correlation decay* that have been of interest because of their many applications. On the one hand, WSM is related with uniqueness of Gibbs measures and the absence of phase transitions [21]. On the other hand, SSM is a strengthening of WSM and it is related to the absence of *boundary phase transitions* [38], and has connections with the existence of FPTAS for #P-hard counting problems [3, 43], mixing time of Glauber dynamics [21], and efficient approximation algorithms for thermodynamic quantities [24, 7].

In [8], there were explored sufficient and necessary conditions for a graph $\mathbb{H}$ to have, for any locally finite graph $\mathbb{G}$, the existence of a weight function λ such that the Gibbs $(\mathbb{G}, \mathbb{H}, \lambda)$-specification satisfies WSM and SSM. In particular, it was proven that dismantlability was equivalent to the existence of Gibbs $(\mathbb{G}, \mathbb{H}, \lambda)$-specifications satisfying WSM for all locally finite graph $\mathbb{G}$, and therefore uniqueness, since WSM implies it. In addition, it was observed that a direct consequence is that a necessary condition for SSM to hold is that $\mathbb{H}$ is dismantlable, because SSM implies WSM. However, it was also shown that it is not a sufficient condition. Here, we strengthen this necessary condition and extend it to the realm of relational structures.

▶ **Proposition 14.** *If $\mathbb{H}^2$ does not dismantle to a substructure of $\Delta(\mathbb{H}^2)$ whose universe contains $\Delta(J^2)$, then there exists a locally finite τ-structure $\mathbb{G}$ such that the Gibbs $(\mathbb{G}, \mathbb{H}, \lambda)$-specification does not satisfy J-SM for any λ.*

Two direct corollaries of this fact are the following.

▶ **Corollary 15.** *If $\mathbb{H}^2$ does not dismantle to some substructure of the diagonal $\Delta(\mathbb{H}^2)$, then there exists a locally finite τ-structure $\mathbb{G}$ such that the Gibbs $(\mathbb{G}, \mathbb{H}, \lambda)$-specification does not satisfy WSM for any λ.*

▶ **Corollary 16.** *If $\mathbb{H}^2$ does not dismantle to the full diagonal $\Delta(\mathbb{H}^2)$, then there exists a locally finite τ-structure $\mathbb{G}$ such that the Gibbs $(\mathbb{G}, \mathbb{H}, \lambda)$-specification does not satisfy SSM for any λ.*

5 Second application: finite duality revisited

Throughout this section all relational structures are assumed to be finite. We say a τ-structure $\mathbb{H}$ is a **core** if every homomorphism from $\mathbb{H}$ to $\mathbb{H}$ is one-to-one. An **obstruction** to $\mathbb{H}$ is a τ-structure $\mathbb{G}$ that admits no homomorphism to $\mathbb{H}$; the obstruction $\mathbb{G}$ is **critical** if every *proper* substructure (i.e., any substructure different from $\mathbb{G}$ itself) admits a homomorphism to $\mathbb{H}$. A relational structure $\mathbb{H}$ is said to have **finite duality** if it has only finitely many critical obstructions.

We say that a τ-structure $\mathbb{H}$ **contains all constants** if for every $a \in H$ there exists $R_a \in \tau$ such that $R_a(\mathbb{H}) = \{a\}$. Note that every such τ-structure is a core. It is well know that relational structures with constants allow us to specify the desired image of a given element. More formally, let $\mathbb{G}$ be any τ-structure, $x \in G$, and $a \in H$. It is immediate that the τ-structure $\mathbb{G}_a$ obtained from $\mathbb{G}$ by adding a to $R_a(\mathbb{G})$, satisfies the following property: For every $\phi : G \to H$,

$$\phi \in \mathrm{Hom}(\mathbb{G}_a, \mathbb{H}) \Leftrightarrow \phi \in \mathrm{Hom}(\mathbb{G}, \mathbb{H}) \text{ and } \phi(x) = a.$$

We shall say that $\mathbb{G}_a$ is obtained by *coloring x to a* in $\mathbb{G}$.

The main result in [34] states that a core relational structure $\mathbb{H}$ has finite duality if and only if $\mathbb{H}^2$ dismantles to its diagonal. It is not difficult ot see that this result follows from our work. In addition, in can be shown that, when $\mathbb{H}$ contains all constants, having finite duality is equivalent to having finitely many critical τ-tree obstructions, which was not previously known.

▶ **Theorem 17.** *Let $\mathbb{H}$ be a finite τ-structure which is a core. Then, the following are equivalent:*
(A1c) *$\mathbb{H}^2$ dismantles to its diagonal;*
(D1c) *$\mathbb{H}$ has finitely many critical obstructions.*

Furthermore, if $\mathbb{H}$ contains all the constants then the following condition is also equivalent:
(D2c) *$\mathbb{H}$ has finitely many critical τ-tree obstructions.*

It has been shown in [23] that if a τ-structure $\mathbb{H}$ has finite duality then there exists some finite set $\mathcal{F}$ of τ-trees such that for every τ-structure $\mathbb{I}$ not homomorphic to $\mathbb{H}$, there exists a τ-tree in $\mathcal{F}$ that is homomorphic to $\mathbb{I}$ but not homomorphic to $\mathbb{H}$. We want to note that the equivalence between conditions $(D2c)$ and $(D1c)$ does not follow from this fact. Indeed, direction $(D2c) \Rightarrow (D1c)$ does not even hold when we do not require that the τ-structure $\mathbb{H}$ is equipped with constants as witnessed by the case when $\mathbb{H}$ is the oriented 3-cycle. Note that, in this case, $\mathbb{H}$ satisfies $(D2c)$ since every τ-tree is homomorphic to $\mathbb{H}$ and, hence, $\mathbb{H}$ has no critical τ-tree obstructions at all. However, since any oriented cycle whose length is not a multiple of 3 is a critical obstruction of $\mathbb{H}$, it follows that $\mathbb{H}$ does not satisfy $(D1c)$.

References

1 Dimitris Achlioptas, Paul Beame, and Michael Molloy. Exponential bounds for DPLL below the satisfiability threshold. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 139–140, 2004.

2 Jung-Chao Ban and Chih-Hung Chang. Tree-shifts: Irreducibility, mixing, and the chaos of tree-shifts. *Trans. Amer. Math. Soc.*, 369(12):8389–8407, 2017.

3 Antar Bandyopadhyay and David Gamarnik. Counting without sampling: Asymptotics of the log-partition function for certain statistical physics models. *Random Structures & Algorithms*, 33(4):452–479, 2008.

4 Manuel Bodirsky. The Complexity of Constraint Satisfaction Problems (Invited Talk). In *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, pages 2–9, 2015.

5 Christian Borgs, Jennifer Chayes, László Lovász, Vera T Sós, and Katalin Vesztergombi. Counting graph homomorphisms. In *Topics in discrete mathematics*, pages 315–371. Springer, 2006.

6 Mike Boyle, Ronnie Pavlov, and Michael Schraudner. Multidimensional sofic shifts without separation and their factors. *Trans. Amer. Math. Soc.*, 362(9):4617–4653, 2010.

7 Raimundo Briceño. The topological strong spatial mixing property and new conditions for pressure approximation. *Ergodic Theory Dynam. Systems*, 38(5):1658–1696, 2018.

8 Raimundo Briceño and Ronnie Pavlov. Strong spatial mixing in homomorphism spaces. *SIAM J. Discrete Math.*, 31(3):2110–2137, 2017.

9 Raimundo Briceño, Andrei Bulatov, Víctor Dalmau, and Benoit Larose. Long range actions, connectedness, and dismantlability in relational structures. *CoRR*, abs/1901.04398, 2019. `arXiv:1901.04398`.

10 Raimundo Briceño, Kevin McGoff, and Ronnie Pavlov. Factoring onto $\mathbb{Z}^d$ subshifts with the finite extension property. *Proc. Amer. Math. Soc.*, 146(12):5129–5140, 2018.

11 Graham R. Brightwell and Peter Winkler. Graph homomorphisms and phase transitions. *J. Combin. Theory Ser. B*, 77(2):221–262, 1999.

12 Graham R. Brightwell and Peter Winkler. Gibbs measures and dismantlable graphs. *J. Combin. Theory Ser. B*, 78(1):141–166, 2000.

13 Graham R. Brightwell and Peter Winkler. Graph homomorphisms and long range action. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 63:29–48, 2004.

14 Andrei A. Bulatov. A Dichotomy Theorem for Nonuniform CSPs. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 319–330, 2017.

15 Andrei A. Bulatov, Víctor Dalmau, Martin Grohe, and Dániel Marx. Enumerating homomorphisms. *J. Comput. Syst. Sci.*, 78(2):638–650, 2012.

16 Andrei A. Bulatov, Andrei A. Krokhin, and Benoit Larose. Dualities for Constraint Satisfaction Problems. In *Complexity of Constraints - An Overview of Current Research Themes [Result of a Dagstuhl Seminar]*, pages 93–124, 2008.

17 Tullio Ceccherini-Silberstein and Michel Coornaert. On the density of periodic configurations in strongly irreducible subshifts. *Nonlinearity*, 25(7):2119, 2012.

18 Víctor Dalmau, Andrei A. Krokhin, and Benoit Larose. First-Order Definable Retraction Problems for Posets and Reflexive Graph. In *19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings*, pages 232–241, 2004.

19 Rina Dechter. *Constraint processing*. Morgan Kaufmann, 2003.

20 Martin Dyer and Catherine Greenhill. The complexity of counting graph homomorphisms. *Random Structures & Algorithms*, 17(3-4):260–289, 2000.

21 Martin Dyer, Alistair Sinclair, Eric Vigoda, and Dror Weitz. Mixing in time and space for lattice spin systems: A combinatorial view. *Random Structures & Algorithms*, 24(4):461–479, 2004.

22 Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing*, 28(1):57–104, 1998.

23 Jan Foniok, Jaroslav Nesetril, and Claude Tardif. Generalised dualities and maximal finite antichains in the homomorphism order of relational structures. *Eur. J. Comb.*, 29(4):881–899, 2008.

24 David Gamarnik and Dmitriy Katz. Sequential cavity method for computing free energy and surface pressure. *Journal of Statistical Physics*, 137(2):205, 2009.

25 Hans-Otto Georgii. *Gibbs Measures and Phase Transitions*, volume 9 of *De Gruyter Studies in Mathematics*. Berlin, 2 edition, 2011.

26 Tatsuhiko Hatanaka, Takehiro Ito, and Xiao Zhou. Complexity of Reconfiguration Problems for Constraint Satisfaction. *CoRR*, abs/1812.10629, 2018.

27 Pavol Hell and Jaroslav Nešetřil. *Graphs and homomorphisms*, volume 28 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2004.

28 Takehiro Ito, Erik D. Demaine, Nicholas J. A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theor. Comput. Sci.*, 412(12-14):1054–1065, 2011.

29 David Kelly and Ivan Rival. Crowns, Fences, and Dismantlable Lattices. *Canadian Journal of Mathematics*, 26(5):1257–1271, 1974.

30 P. Kolaitis and M. Vardi. Conjunctive-Query Containment and Constraint Satisfaction. *Journal of Computer and System Sciences*, 61:302–332, 2000.

31 Marcin Kozik. Weak consistency notions for all the CSPs of bounded width. In *31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16*, pages 633–641, 2016.

32 Florent Krzakała, Andrea Montanari, Federico Ricci-Tersenghi, Guilhem Semerjian, and Lenka Zdeborová. Gibbs states and the set of solutions of random constraint satisfaction problems. *PNAS*, 104(25):10318–10323, 2007.

33 Gábor Kun, Ryan O'Donnell, Suguru Tamaki, Yuichi Yoshida, and Yuan Zhou. Linear programming, width-1 CSPs, and robust satisfaction. In *Innovations in Theoretical Computer Science, ICT '12*, pages 484–495, 2012.

34 Benoit Larose, Cynthia Loten, and Claude Tardif. A characterisation of first-order constraint satisfaction problems. *Log. Methods Comput. Sci.*, 3(4):4:6, 22, 2007.

35 Douglas Lind and Brian Marcus. *An introduction to symbolic dynamics and coding*. Cambridge university press, 1995.

36 Douglas Lind and Klaus Schmidt. Symbolic and algebraic dynamical systems. In *Handbook of dynamical systems*, volume 1, pages 765–812. Elsevier, 2002.

37 Brian Marcus and Joachim Rosenthal. *Codes, systems, and graphical models*, volume 123. Springer Science & Business Media, 2012.

38 Fabio Martinelli, Enzo Olivieri, and Roberto H Schonmann. For 2-D lattice spin systems weak mixing implies strong mixing. *Communications in Mathematical Physics*, 165(1):33–47, 1994.

39 Marc Mézard and Andrea Montanari. *Information, Physics, and Computation*. Oxford University press, 2009.

40 Naomi Nishimura. Introduction to Reconfiguration. *Algorithms*, 11(4):52, 2018.

41 Richard Nowakowski and Peter Winkler. Vertex-to-vertex pursuit in a graph. *Discrete Mathematics*, 43(2-3):235–239, 1983.

42 Ronnie Pavlov and Michael Schraudner. Entropies realizable by block gluing $\mathbb{Z}^d$ shifts of finite type. *Journal d'Analyse Mathématique*, 126(1):113–174, 2015.

43 Dror Weitz. Counting Independent Sets Up to the Tree Threshold. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, STOC '06, pages 140–149, New York, NY, USA, 2006. ACM.

44 Dmitriy Zhuk. A Proof of CSP Dichotomy Conjecture. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 331–342, 2017.

Sign-Rank Can Increase Under Intersection

Mark Bun
Simons Institute for the Theory of Computing, Berkeley, CA, USA
Boston University, MA, USA

Nikhil S. Mande
Georgetown University, Washington, DC, USA

Justin Thaler
Georgetown University, Washington, DC, USA

Abstract

The communication class $\mathbf{UPP}^{\mathrm{cc}}$ is a communication analog of the Turing Machine complexity class $\mathbf{PP}$. It is characterized by a matrix-analytic complexity measure called sign-rank (also called dimension complexity), and is essentially the most powerful communication class against which we know how to prove lower bounds.

For a communication problem f, let $f \wedge f$ denote the function that evaluates f on two disjoint inputs and outputs the AND of the results. We exhibit a communication problem f with $\mathbf{UPP}^{\mathrm{cc}}(f) = O(\log n)$, and $\mathbf{UPP}^{\mathrm{cc}}(f \wedge f) = \Theta(\log^2 n)$. This is the first result showing that $\mathbf{UPP}$ communication complexity can increase by more than a constant factor under intersection. We view this as a first step toward showing that $\mathbf{UPP}^{\mathrm{cc}}$, the class of problems with polylogarithmic-cost $\mathbf{UPP}$ communication protocols, is not closed under intersection.

Our result shows that the function class consisting of intersections of two majorities on n bits has dimension complexity $n^{\Omega(\log n)}$. This matches an upper bound of (Klivans, O'Donnell, and Servedio, FOCS 2002), who used it to give a quasipolynomial time algorithm for PAC learning intersections of polylogarithmically many majorities. Hence, fundamentally new techniques will be needed to learn this class of functions in polynomial time.

2012 ACM Subject Classification Theory of computation → Communication complexity; Theory of computation → Boolean function learning

Keywords and phrases Sign rank, dimension complexity, communication complexity, learning theory

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.30

Category Track A: Algorithms, Complexity and Games

Related Version https://eccc.weizmann.ac.il/report/2019/027/

Acknowledgements Nikhil Mande and Justin Thaler were supported by NSF Grant CCF-1845125.

1 Introduction

The unbounded-error communication complexity model $\mathbf{UPP}^{\mathrm{cc}}$ was introduced by Paturi and Simon [21] as a natural communication analog of the Turing Machine complexity class $\mathbf{PP}$. In a $\mathbf{UPP}^{\mathrm{cc}}$ communication protocol for a Boolean function $f(x, y)$, there are two parties, one with input x and one with input y. The two parties engage in a private-coin randomized communication protocol, at the end of which they are required to output $f(x, y)$ with probability strictly greater than $1/2$. The cost of the protocol is the number of bits exchanged by the two parties. As is standard, we use the notation $\mathbf{UPP}^{\mathrm{cc}}$ not only to denote the communication model, but also the class of functions solvable in the model by protocols of cost polylogarithmic in the size of the input.

Observe that success probability $1/2$ can be achieved with no communication by random guessing, so the $\mathbf{UPP}^{\mathrm{cc}}$ model merely requires a strict improvement over this trivial solution.


Owing to this liberal acceptance criterion, $\mathbf{UPP}^{\mathrm{cc}}$ is a very powerful communication model, essentially the most powerful one against which we know how to prove lower bounds. In particular, $\mathbf{UPP}^{\mathrm{cc}}$ is powerful enough to simulate many other models of computing, and this makes $\mathbf{UPP}^{\mathrm{cc}}$ *lower bounds* highly useful. As one example, any function $f(x, y)$ computable by a Threshold-of-Majority circuit of size s has $\mathbf{UPP}^{\mathrm{cc}}$ complexity at most $O(\log s)$, and this connection has been used to translate $\mathbf{UPP}^{\mathrm{cc}}$ lower bounds into state of the art lower bounds against threshold circuits (see, for example, [10, 22, 8, 28, 6]).

$\mathbf{UPP}^{\mathrm{cc}}$ also happens to be characterized by a natural matrix-analytic complexity measure called *sign-rank* [21]. Here, the sign-rank of a matrix $M \in \{-1, 1\}^{N \times N}$ is the minimum rank of a real matrix whose entries agree in sign with M. Equivalently, $\mathrm{sr}(M) := \min_A \mathrm{rk}(A)$, where the minimum is over all matrices A such that $A_{i,j} \cdot M_{i,j} > 0$ for all $i, j \in [N]$. Paturi and Simon [21] showed the following tight connection between $\mathbf{UPP}^{\mathrm{cc}}$ and sign-rank: if we associate a function $f(x, y)$ with the matrix $M = [f(x, y)]_{x,y}$, then the $\mathbf{UPP}^{\mathrm{cc}}$ communication complexity of f equals $\log(\mathrm{sr}(M)) \pm \Theta(1)$.

While lower bounds on $\mathbf{UPP}^{\mathrm{cc}}$ complexity (equivalently, sign-rank) are useful in complexity theory, *upper bounds* on these quantities imply state of the art learning algorithms, including the fastest known algorithms for PAC learning DNFs and read-once formulas [17, 1]. More specifically, suppose we want to learn a concept class $\mathcal{C}$ of functions mapping $\{-1, 1\}^n$ to $\{-1, 1\}$. $\mathcal{C}$ is naturally associated with a $|\mathcal{C}| \times 2^n$ matrix M, whose ith row equals the truth table of the ith function in $\mathcal{C}$. Then $\mathcal{C}$ can be distribution-independently PAC learned in time polynomial in the sign-rank of M. (The sign-rank of M is often referred to in the learning theory literature as the *dimension complexity* of $\mathcal{C}$.) Moreover, the resulting learning algorithm is robust to random classification noise, a property not satisfied by the handful of known PAC learning algorithms that are *not* based on dimension complexity.

For the purpose of our work, one particularly important application of the dimension-complexity approach to PAC learning was derived by Klivans et al. [16], who showed that the concept class consisting of intersections of 2 majority functions has dimension complexity at most $\binom{n}{O(\log n)} \leq n^{O(\log n)}$. They thereby obtained a quasipolynomial time algorithm for PAC learning intersections of two majority functions.[1] Prior to our work, it was consistent with current knowledge that the dimension complexity of this concept class is in fact $\mathrm{poly}(n)$, which would yield a polynomial time PAC learning algorithm for intersections of constantly many majority functions.

1.1 Our Results

Despite considerable effort, progress on understanding sign-rank (equivalently, $\mathbf{UPP}^{\mathrm{cc}}$) has been slow. Our lack of knowledge is highlighted via the following well-known open question (cf. Göös et al. [13]). Throughout, for any function $f\colon \{-1, 1\}^n \to \{-1, 1\}$, $f \wedge f$ denotes the function on twice as many inputs obtained by evaluating f on two disjoint inputs and outputting -1 only if both copies of f evaluate to -1, i.e., $(f \wedge f)(x_1, x_2) := f(x_1) \wedge f(x_2)$.

▶ **Question 1.** *Is the class* $\mathbf{UPP}^{cc}$ *closed under intersection? In other words, suppose the function* $f(x, y)\colon \{-1, 1\}^n \times \{-1, 1\}^n \to \{-1, 1\}$ *satisfies* $\mathbf{UPP}^{cc}(f) = O((\log n)^c)$ *for some constant* c*. Is there always some constant* c_1 *(which may depend on* c*) such that* $\mathbf{UPP}(f \wedge f) \leq O\left((\log n)^{c_1}\right)$*? More generally and informally, if* $\mathbf{UPP}^{cc}(f)$ *is "small", does this imply any non-trivial upper bound on* $\mathbf{UPP}^{cc}(f \wedge f)$*?*

[1] In fact, their algorithm runs in quasipolynomial time for intersections of polylogarithmic many majorities.

Prior to our work, essentially nothing was known about Question 1. In particular, we are not aware of prior work ruling out the possibility that $\mathbf{UPP}^{\mathrm{cc}}(f \wedge f) \leq O(\mathbf{UPP}^{\mathrm{cc}}(f))$. On the other hand, for reasons that will become apparent in Section 1.2, there is good reason to suspect that there exists a function f with $\mathbf{UPP}^{\mathrm{cc}}(f) = O(\log n)$, yet $\mathbf{UPP}^{\mathrm{cc}}(f \wedge f) \geq \Omega(n)$. While we do not obtain a full resolution of Question 1, we do show for the first time that $\mathbf{UPP}^{\mathrm{cc}}$ complexity can increase significantly under intersection.

Babai, Frankl and Simon [2] observed that there are two natural communication complexity analogs of the Turing machine class $\mathbf{PP}$, namely $\mathbf{PP}^{\mathrm{cc}}$ and $\mathbf{UPP}^{\mathrm{cc}}$. It is well known [3] that $\mathbf{PP}^{\mathrm{cc}}$ *is* closed under intersection. Our work can be viewed as a first step towards showing that, in contrast, $\mathbf{UPP}^{\mathrm{cc}}$ is *not* closed under intersection.

▶ **Theorem 1.** *There is a function* $f(x, y)\colon \{-1,1\}^n \times \{-1,1\}^n \to \{-1,1\}$ *such that* $\mathbf{UPP}^{cc}(f) = O(\log n)$*, yet* $\mathbf{UPP}^{cc}(f \wedge f) = \Theta(\log^2 n)$*.*

In fact, for each fixed $x \in \{-1,1\}^n$, the function $f(x, y)$ from Theorem 1 simply outputs the majority of some subset of the bits of y. This yields the following corollary.

▶ **Corollary 2.** *Let* $\mathcal{C}$ *be the concept class in which each concept is the intersection of two majorities on* n *bits. Then* $\mathcal{C}$ *has dimension complexity* $n^{\Theta(\log n)}$*.*

Corollary 2 shows that the dimension complexity upper bound of Klivans et al. [16] is tight for intersections of two majorities, and new approaches will be needed to PAC learn this concept class in polynomial time. For context, we remark that learning intersections of majorities is a special case of the more general problem of learning intersections of many *halfspaces*.[2] The latter is a central and well-studied challenge in learning theory, as intersections of halfspaces are powerful enough to represent any convex set, and they contain many basic problems (like learning DNFs) as special cases. In contrast to the well-understood problem of learning a single halfspace, for which many efficient algorithms are known, no $2^{o(n)}$-time algorithm is known for PAC learning even the intersection of two halfspaces. There have been considerable efforts devoted to showing that learning intersections of halfspaces is a hard problem [18, 9, 15, 4], but these results apply only to intersections of many halfspaces, or make assumptions about the form of the output hypothesis of the learner. Our work can be seen as a new form of evidence that learning intersections of even two majorities is hard.

1.2 Our Techniques

$\mathbf{UPP}^{\mathrm{cc}}$ has a *query complexity* analog, denoted $\mathbf{UPP}^{\mathrm{dt}}$ and defined as follows. A $\mathbf{UPP}^{\mathrm{dt}}$ algorithm is a randomized algorithm which on input x, queries bits of x, and must output $f(x)$ with probability strictly greater than $1/2$; the cost of the protocol is the number of bits of x queried. How $\mathbf{UPP}^{\mathrm{dt}}$ behaves under intersection is now well understood. More specifically, it is known [25] that there is a function $f\colon \{-1,1\}^n \to \{-1,1\}$ (in fact, a halfspace) such that $\mathbf{UPP}^{\mathrm{dt}}(f) = O(1)$, yet $\mathbf{UPP}^{\mathrm{dt}}(f \wedge f) = \Theta(n)$. Define the Majority function, which we denote by MAJ, to be -1 if at least half of its input bits are -1. It is also known [26, 20] that MAJ satisfies $\mathbf{UPP}^{\mathrm{dt}}(\mathrm{MAJ}) = O(1)$, yet $\mathbf{UPP}^{\mathrm{dt}}(\mathrm{MAJ} \wedge \mathrm{MAJ}) = \Theta(\log n)$. Our goal in this paper is, to the extent possible, to show that the $\mathbf{UPP}^{\mathrm{cc}}$ communication model behaves similarly to its query complexity analog.

[2] A halfspace is any function of the form $\mathrm{sgn}\left(\sum_{i=1}^n w_i \cdot x_i + w_0\right)$ for some real numbers $w_0, \ldots, w_n$.

Over the course of the last decade, there has been considerable progress in proving *lifting theorems* [23, 12, 11]. These theorems seek to show that if a function f has large complexity in some query model $\mathbf{C}$, then for some "sufficiently complicated" function g on a "small" number of inputs, the composition $f \circ g$ has large complexity in the associated communication model (ideally, $\mathbf{C}^{\mathrm{cc}}(f \circ g) \gtrsim \mathbf{C}^{dt}(f)$).

Unfortunately, a "generic" lifting theorem for $\mathbf{UPP}$ complexity is not known. That is, it is not know how to take an arbitrary function f with high $\mathbf{UPP}^{\mathrm{dt}}$ complexity, and by composing it with a function g on a small number of inputs, yield a function with high $\mathbf{UPP}^{\mathrm{cc}}$ complexity.

However, as we now explain, some significant partial results have been shown in this direction. It is well-known that $\mathbf{UPP}^{\mathrm{dt}}(f)$ is equivalent to an approximation-theoretic notion called *threshold degree*, denoted $\deg_{\pm}(f)$ (which we do not define here). The threshold degree of f can in turn be expressed as the value of a certain (exponentially large) linear program. Linear programming duality then implies that one can prove lower bounds on $\deg_{\pm}(f)$ by exhibiting good solutions to the dual linear program. We refer to such dual solutions as *dual witnesses* for threshold degree. Sherstov [24] and Razborov and Sherstov [22] showed that if $\deg_{\pm}(f)$ is large, and moreover this can be exhibited by a dual witness satisfying a certain *smoothness* condition, then there is a function g defined on a constant number of inputs such that $f \circ g$ *does* have large $\mathbf{UPP}^{\mathrm{cc}}$ complexity. Several recent works [6, 5, 7, 28] have managed to prove new $\mathbf{UPP}^{\mathrm{cc}}$ lower bounds by constructing, for various functions f, smooth dual witnesses exhibiting the fact that $\deg_{\pm}(f)$ is large.

Our key technical contribution is to bring this approach to bear on the function $F(x, y) = \mathrm{MAJ}(x) \wedge \mathrm{MAJ}(y)$. Specifically, we show that the (known) threshold degree lower bound $\deg_{\pm}(F) \geq \Omega(\log n)$ can be exhibited by a smooth dual witness.

We do this as follows. Sherstov [26] showed that for any function $f\colon \{-1,1\}^n \to \{-1,1\}$, the threshold degree of the function $F = f \wedge f$ is characterized by the *rational* approximate degree of f, i.e., the least total degree of real polynomials p and q such that $|f(x) - p(x)/q(x)| \leq 1/3$ for all $x \in \{-1,1\}^n$. He then showed that the rational approximate degree of MAJ is $\Omega(\log n)$, thereby concluding that $F(x, y)$ has threshold degree $\Omega(\log n)$.

From Sherstov's arguments, one can derive a dual witness ψ for the fact that the rational approximate degree of MAJ is $\Omega(\log n)$, and then transform ψ into a dual witness ϕ for the fact that $F(x, y)$ has threshold degree $\Omega(\log n)$. Unfortunately, neither ψ nor ϕ satisfies the type of smoothness condition required by Razborov and Sherstov's machinery to yield $\mathbf{UPP}^{\mathrm{cc}}$ lower bounds.

The smoothness condition required for the Razborov-Sherstov machinery to work essentially states that the the mass of the dual witness ψ has to be "relatively large" (a reasonably large fraction of what mass the uniform distribution would have placed) on a "large" set of inputs (the fraction of inputs which do not have large mass has to be small).

To construct a *smooth* dual witness ψ' for F, our primary technical contribution is to construct a *smooth* dual witness ϕ' for the fact that the rational approximate degree of MAJ is $\Omega(\log n)$. We then apply a different transformation, due to Sherstov [27], of ϕ' into a dual witness for the fact that the threshold degree of F is $\Omega(\log n)$, and we show that this transformation *preserves the smoothness* of ψ'.

In a nutshell, our smooth dual witness for MAJ is obtained in two steps: first we define for all inputs x whose Hamming weight lies in $[n/2 - \lfloor n^{2/3} \rfloor, n/2 + \lfloor n^{2/3} \rfloor]$, a dual witness ϕ'_x that places a large mass on x and not too much mass on other points. Next, we define the final dual witness $\phi'(x)$ to be a certain weighted average over x of all the dual witnesses thus obtained. The resulting mass on $\phi'(x)$ for each x of Hamming weight

in $[n/2 - \lfloor n^{2/3} \rfloor, n/2 + \lfloor n^{2/3} \rfloor]$ is large enough, and the fraction of inputs whose Hamming weight is not in $[n/2 - \lfloor n^{2/3} \rfloor, n/2 + \lfloor n^{2/3} \rfloor]$ is small enough, to allow us to use the Razborov-Sherstov framework (Theorem 5) to prove the desired sign-rank lower bound on the pattern matrix of F.

2 Preliminaries

All logarithms in this paper are taken base 2. We use the notation $\exp(x)$ to denote e^x, where e is Euler's number. Given any finite set X and any functions $f, g : X \to \mathbb{R}$, define $\|f\|_1 := \sum_{x \in X} |f(x)|$ and $\langle f, g \rangle := \sum_{x \in X} f(x)g(x)$. We refer to $\|f\|_1$ as the ℓ_1-norm of f. For any $x \in \{-1,1\}^n$, we use the notation $|x|$ to denote the *Hamming weight* of x, which is the number of -1's in the string x.

Paturi and Simon [21] showed the following equivalence between the sign-rank of a matrix and the $\mathbf{UPP}^{\mathrm{cc}}$ cost of its corresponding communication game.

▶ **Theorem 3.** *For any $F : \{-1,1\}^{2n} \times \{-1,1\}^n \to \{-1,1\}$, let M_F denote its* communication matrix*, defined by $M_F(x,y) = F(x,y)$. Then, $\mathbf{UPP}^{cc}(F) = \log \mathrm{sr}(M_F) \pm O(1)$.*

Let n, N be positive integers such that n divides N. Partition the set $[N] := \{1, \ldots, N\}$ into n disjoint blocks $\{1, 2, \ldots, N/n\}, \{N/n + 1, \ldots, 2N/n\}, \ldots, \{(n-1)N/n + 1, \ldots, N\}$. Define the set $\mathcal{P}(N,n)$ to be the collection of subsets of $[N]$ which contain exactly one element from each block. For $x \in \{-1,1\}^n$ and $S \in \mathcal{P}(N,n)$, let $x|_S = (x_{s_1}, \ldots, x_{s_n})$, where $s_1 < s_2 < \cdots < s_n$ are the elements of S.

▶ **Definition 4** (Pattern matrix)**.** *For any function $\phi : \{-1,1\}^n \to \mathbb{R}$, the (N, n, ϕ)-pattern matrix M is defined as follows.*

$$M = [\phi(x|_S) \oplus w]_{x \in \{-1,1\}^N, (S,w) \in \mathcal{P}(N,n) \times \{-1,1\}^n}.$$

Note that M is a $2^N \times (N/n)^n 2^n$ matrix.

In a breakthrough result, Forster [10] proved that an upper bound on the spectral norm of a sign matrix implies a lower bound on its sign-rank. Razborov and Sherstov [22] established a generalization of Forster's theorem [10] that can be used to prove sign-rank lower bounds for pattern matrices. Specifically, we require the following result, implicit in their work [22, Theorem 1.1].

▶ **Theorem 5** (Implicit in [22])**.** *Let $f : \{-1,1\}^n \to \{-1,1\}$ be any Boolean function and $\alpha > 1$ be a real number. Suppose there exists a function $\phi : \{-1,1\}^n \to \mathbb{R}$ satisfying the following conditions.*

- $\sum_{x \in \{-1,1\}^n} |\phi(x)| = 1$.
- *For all polynomials p of degree at most d, $\sum_{x \in \{-1,1\}^n} \phi(x)p(x) = 0$.*
- $f(x) \cdot \phi(x) \geq 0 \ \forall x \in \{-1,1\}^n$.
- $|\phi(x)| \geq \gamma$ *for all but a Δ fraction of inputs $x \in \{-1,1\}^n$.*

Then, the sign-rank of the (N, n, f)-pattern matrix M can be bounded below as

$$\mathrm{sr}(M) \geq \frac{\gamma}{\frac{1}{2^n}\left(\frac{n}{N}\right)^{d/2} + \gamma\Delta}.$$

We require the following well-known combinatorial identity.

▷ Claim 6. For every polynomial p of degree less than $2n$, we have $\sum_{t=-n}^{n} (-1)^t \binom{2n}{n+t} p(t) = 0$.

Recall from Section 1.2 that the rational ϵ-approximate degree of f is the least degree of two polynomials p and q such that $|f(x) - p(x)/q(x)| \leq \epsilon$ for all x in the domain of f. Sherstov [27, Theorem 6.9] showed that a dual witness to the rational approximate degree of any function f can be converted to a threshold degree dual witness for $\mathrm{OR}_n \circ f$. Implicit in his theorem is the fact that a *smooth* dual witness to the rational approximate degree of f can be converted to a *smooth* dual witness for the threshold degree of $\mathrm{OR}_n \circ f$. More precisely, the following result is established by the proof of [27, Theorem 6.9].

▶ **Theorem 7** (Sherstov [27]). *Let $f : \{-1,1\}^n \to \{-1,1\}$ be any function. Let F denote $\mathrm{OR}_t \circ f : \{-1,1\}^{nt} \to \{-1,1\}$, and $\delta > \varepsilon > 0$ be any real numbers.*

Suppose there exist functions $\psi_0, \psi_1 \colon \{-1,1\}^n \to \mathbb{R}$ that are not identically 0 and satisfy the following properties:

$$f(x) = 1 \implies \psi_0(x) \geq \delta|\psi_1(x)|, \tag{1}$$

$$f(x) = -1 \implies \psi_1(x) \geq \delta|\psi_0(x)|, \tag{2}$$

$$\deg(p) < d \implies \langle \psi_0, p\rangle = 0 \text{ and } \langle \psi_1, p\rangle = 0. \tag{3}$$

Then there exist functions $A, B : \{-1,1\}^{nt} \to \mathbb{R}$ such that $\Psi = \frac{1}{\delta}A - \frac{1}{\varepsilon}B$ satisfies the following properties.

$$\deg(p) \leq \min\left\{\lfloor \varepsilon^2 t \rfloor d, d\right\} \implies \langle \Psi, p\rangle = 0. \tag{4}$$

$$F(x) \cdot \Psi(x_1, \ldots, x_t) \geq (\delta - \varepsilon)^{2t} \prod_{i=1}^{t} |\psi_0(x_i)| \text{ for all } x \in \{-1,1\}^{nt}. \tag{5}$$

$$|A(x_1, \ldots, x_t)| \leq \prod_{i=1}^{t} |\psi_0(x_i)| \text{ for all } x = (x_1, \ldots, x_t) \in \{-1,1\}^{nt}. \tag{6}$$

$$|B(x_1, \ldots, x_t)| \leq \prod_{i : f(x_i) = 0} |\psi_0(x_i)| \cdot \prod_{i : f(x_i) = 1} \delta\psi_1(x_i) + \prod_{i=1}^{t} (|\psi_0(x_i)| - \delta\psi_1(x_i))$$
$$\text{for all } x = (x_1, \ldots, x_t) \in \{-1,1\}^{nt}. \tag{7}$$

3 A Smooth Dual Witness for Majority

Our main technical contribution in this paper is captured in Theorem 8 below. This theorem constructs a *smooth* dual witness R for the hardness of rationally approximating the sign function on $\{0, \pm 1, \ldots, \pm n\}$. We defer the proof until Section 4.

▶ **Theorem 8.** *Let $1 \leq d \leq \frac{1}{3}\log n$ and let n be odd. There exists a function $R : \{0, \pm 1, \ldots, \pm n\} \to \mathbb{R}$ such that*

- $$\sum_{t=-n}^{n} |R(t)| = 1. \tag{8}$$
- *For $\delta = \exp(-18/(n^{1/(6d)}))$ and every $t = 1, 2, \ldots, n$,*

  $$R(t) \geq \delta|R(-t)|. \tag{9}$$

- *If $p : \{0, \pm 1, \ldots, \pm n\} \to \mathbb{R}$ is any polynomial of degree less than $d - 2$, then*

  $$\langle R, p\rangle = 0. \tag{10}$$

- *For every* $t \in \{0, \pm 1, \pm 2, \ldots, \pm\lfloor n^{2/3} \rfloor\}$ *we have*

$$|R(t)| \geq \Omega\left(\frac{1}{n^{20}}\right). \tag{11}$$

The following theorem shows how to convert the (univariate) function R from Theorem 8 into a dual witness for the (multivariate) MAJ function.

▶ **Theorem 9.** *Let* $1 \leq d \leq \frac{1}{3}\log n$ *and let* n *be odd. Let* $R : \{0, \pm 1, \ldots, \pm n\} \to \mathbb{R}$ *be any function obtained in Theorem 8. Then, the multivariate polynomial* $R' : \{-1,1\}^{2n} \to \mathbb{R}$ *defined by* $R'(x) = R(n - |x|)/\binom{2n}{|x|}$ *satisfies the following properties.*

- $\|R'\|_1 = 1.$ (12)
- *For* $\delta = \exp(-18/(n^{1/(6d)}))$ *and every* $t = 1, 2, \ldots, n,$

$$R'(x) \geq \delta |R'(y)| \tag{13}$$

for any $x, y \in \{-1,1\}^{2n}$ *such that* $|x| = n - t, |y| = n + t.$
- *For any polynomial* p *of degree at most* $d - 2,$

$$\langle R', p \rangle = 0. \tag{14}$$

- *For all* $x \in \{-1,1\}^{2n}$ *such that* $n - \lfloor n^{2/3} \rfloor \leq |x| \leq n + \lfloor n^{2/3} \rfloor,$

$$|R'(x)| \geq \Omega\left(\frac{1}{n^{20} \cdot 2^{2n}}\right). \tag{15}$$

Proof. To establish Equation (12), observe:

$$\begin{aligned}\|R'\|_1 &= \sum_{x \in \{-1,1\}^{2n}} |R'(x)| = \sum_{t=0}^{2n} \left(\sum_{x \in \{-1,1\}^{2n} : |x| = t} |R'(x)| \right) \\ &= \sum_{t=0}^{2n} \binom{2n}{t} |R(n-t)| / \binom{2n}{t} = \sum_{t=-n}^{n} |R(t)| = 1,\end{aligned}$$

where the last equality follows from Equation (8). Equation (13) follows directly from Equation (9) and the definition of R'.

To establish Equation (14), consider any polynomial $p : \{-1,1\}^{2n} \to \mathbb{R}$ of degree at most $d - 2$. For any permutation $\sigma \in S_{2n}$, define the polynomial p_σ by $p_\sigma(x_1, \ldots, x_{2n}) = p(x_{\sigma(1)}, \ldots, x_{\sigma(2n)})$. Note that, since R' is symmetric, $\langle R', p_\sigma \rangle = \langle R', p \rangle$ for all $\sigma \in S_{2n}$. Define $q = \mathbb{E}_{\sigma \in S_{2n}}[p_\sigma]$. Note that q is symmetric and $\langle R', p \rangle = \langle R', q \rangle$. It is a well-known fact (cf. [19]) that q can be written as a polynomial q' of degree at most $d - 2$ in the variable $\sum_{i=1}^{2n} x_i$, and so can R'. Hence, $\langle R', p \rangle = \langle R', q \rangle = \sum_{t=0}^{2n} \binom{2n}{t} \frac{R(n-t)}{\binom{2n}{t}} \cdot q'(t) = 0$, where the final equality holds by Equation (10).

To establish Equation (15), observe that by Equation (11) and the definition of R', we have that for all $x \in \{-1,1\}^{2n}$ such that $|x| \in \left[n - \lfloor n^{2/3} \rfloor, n + \lfloor n^{2/3} \rfloor\right]$, $|R'(x)| \geq \Omega\left(\frac{1}{n^{20} \cdot \binom{2n}{|x|}}\right) \geq \Omega\left(\frac{1}{n^{20} \cdot 2^{2n}}\right).$ ◀

We are ready to derive a lower bound on the sign-rank of the $(4n^2, 4n, \mathrm{OR}_2 \circ \mathrm{MAJ}_{2n})$-pattern matrix.

▶ **Theorem 10.** *The* $(4n^2, 4n, \mathrm{OR}_2 \circ MAJ_{2n})$*-pattern matrix* M *satisfies* $\mathrm{sr}(M) \geq n^{\Omega(\log n)}.$

Proof. Let F denote the function $\mathrm{OR}_2 \circ \mathrm{MAJ}_{2n}$ in this proof. Set $d = \log n/100$ and consider the function $R : \{0, \pm 1, \ldots, \pm n\} \to \mathbb{R}$ obtained via Theorem 8. Define the function $R' : \{0, \pm 1, \ldots, \pm n\} \to \mathbb{R}$ by $R'(t) = R(-t)$. Define the functions $\psi_0, \psi_1 : \{-1,1\}^{2n} \to \mathbb{R}$ by $\psi_1(x) = R(n - |x|)/\binom{2n}{|x|}$, and $\psi_0(x) = R'(n - |x|)/\binom{2n}{|x|}$. We now verify that ψ_0, ψ_1 satisfy the conditions in Theorem 7 for $\delta = \exp(-18/(n^{1/(6d)})) = \exp(-18/n^{100/6 \log n}) = \exp(-18/2^{100/6}) > 0.99$. Set $\varepsilon = \delta \cdot c$, where $c > 0$ is a constant such that $0.98 > \delta \cdot c > 1/\sqrt{2}$.

- By the definitions of ψ_0, ψ_1 and Equation (13), Properties (1) and (2) in the statement of Theorem 7 are satisfied.
- Equation (14) implies that $\langle \psi_0, p \rangle = \langle \psi_1, p \rangle = 0$ for any polynomial p of degree at most $d - 2$, and hence Property (3) is satisfied.

Moreover, Equation (15) implies that $|\psi_0(x)|, |\psi_1(x)| \geq \Omega\left(\frac{1}{n^{20} \cdot 2^{2n}}\right)$ for all $x \in \{-1,1\}^{2n}$ such that $n - \lfloor n^{2/3} \rfloor \leq |x| \leq n + \lfloor n^{2/3} \rfloor$, and Equation (12) implies $\|\psi_0\|_1 = \|\psi_1\|_1 = 1$. Theorem 7 now implies the existence of a function Ψ satisfying the following properties.

- By Equation (4), $\deg(p) < \min\left\{\lfloor 2\varepsilon^2 \rfloor \cdot ((\log n)/100 - 2), (\log n)/100 - 2\right\} \implies \langle \Psi, p \rangle = 0$. Since $\varepsilon > 1/\sqrt{2}$, this implies that

$$\deg(p) < (\log n)/100 - 2 \implies \langle \Psi, p \rangle = 0.$$

- By Equation (5), $\Psi(x) \cdot F(x) \geq 0$ for all $x \in \{-1,1\}^{2n} \times \{-1,1\}^{2n}$.
- We now note that the functions A and B obtained in Theorem 7 have ℓ_1-norm at most a constant. Since $\|\psi_0\|_1 = \|\psi\|_1 = 1$, we use Equation (6) to conclude that

$$\sum_{x_1, x_2 \in \{-1,1\}^{2n} \times \{-1,1\}^{2n}} |A(x_1, x_2)| \leq \sum_{x_1 \in \{-1,1\}^{2n}} |\psi_0(x_1)| \cdot \sum_{x_2 \in \{-1,1\}^{2n}} |\psi_0(x_2)| = 1.$$

  By Equation (7), we have

$$\sum_{x_1, x_2 \in \{-1,1\}^{2n}} |B(x_1, x_2)| \leq \max\left\{\|\psi_0\|_1, \delta\|\psi_1\|_1\right\}^2 + \|\psi_0\|_1^2 + \delta\|\psi_0\|_1\|\psi_1\|_1 + \delta^2\|\psi_1\|_1^2,$$

  which is at most a constant, since $\delta = O(1)$.
  Combined with the fact that ε is a constant, we conclude $\|\Psi\|_1 \leq \frac{1}{\delta}\|A\|_1 + \frac{1}{\varepsilon}\|B\|_1 \leq O(1)$.
- By Equation (5), $F(x) \cdot \Psi(x_1, x_2) \geq (\delta - \varepsilon)^4 |\psi_0(x_1)| \cdot |\psi_0(x_2)| \ \forall x \in \{-1,1\}^{4n}$. This implies that for $|x_1|, |x_2| \in [n - \lfloor n^{2/3} \rfloor, n + \lfloor n^{2/3} \rfloor]$,

$$|\Psi(x_1, x_2)| \geq \Omega\left(\frac{1}{n^{40} \cdot 2^{4n}}\right),$$

  since $\delta - \varepsilon = \Omega(1)$.
- By a standard Chernoff bound, the number of inputs in $\{-1,1\}^{2n} \times \{-1,1\}^{2n}$ such that $|x_1|, |x_2| \in [n - \lfloor n^{2/3} \rfloor, n + \lfloor n^{2/3} \rfloor]$ is at least $(1 - 2\exp(-n^{1/3}/3)) \cdot 2^{4n}$.

Plugging $f = \mathrm{OR}_2 \circ \mathrm{MAJ}_{2n}$ and $\phi = \frac{\Psi}{\|\Psi\|_1}$ into Theorem 5, we conclude that the sign-rank of the $(4n^2, 4n, \mathrm{OR}_2 \circ \mathrm{MAJ}_{2n})$ pattern matrix M is bounded below as

$$\mathrm{sr}(M) \geq \Omega\left(\frac{\frac{1}{n^{40}} \cdot \frac{1}{2^{4n}}}{\left(\frac{1}{n^{(\log n/200)-1}} \cdot \frac{1}{2^{4n}}\right) + \left(\frac{1}{n^{40}} \cdot \frac{1}{2^{4n}} \cdot 2\exp(-n^{1/3}/3)\right)}\right) \geq n^{\Omega(\log n)}. \qquad \blacktriangleleft$$

We are now ready to prove Theorem 1.

Proof of Theorem 1. Note that the function $\mathrm{AND} \circ \mathrm{MAJ}(x) = \overline{\mathrm{OR} \circ \mathrm{MAJ}(\overline{x})}$. Consider the dual witness $\phi = \frac{\Psi}{\|\Psi\|_1}$ obtained for the threshold degree of $\mathrm{OR}_2 \circ \mathrm{MAJ}_{2n}$ in the previous proof. Note that the function ϕ' defined by $\phi'(x) = -\phi(\overline{x})$ acts as a dual witness for the threshold degree of $\mathrm{AND}_2 \circ \mathrm{MAJ}_{2n}$, and satisfies all the conditions in Theorem 5 with the same parameters as in the proof of Theorem 10. Proceeding in exactly the same way as in the previous proof, we conclude that sign-rank of the $(4n^2, 4n, \mathrm{AND}_2 \circ \mathrm{MAJ}_{2n})$ pattern matrix M' is bounded below as

$$\mathrm{sr}(M') \geq n^{\Omega(\log n)}. \tag{16}$$

Denote by f the communication game corresponding to the $(2n^2, 2n, \mathrm{MAJ}_{2n})$ pattern matrix. For completeness, we now sketch a standard $\mathbf{UPP}^{\mathrm{cc}}$ protocol of cost $O(\log n)$ for f. Note that Alice holds $2n^2$ input bits, and Bob holds a $(2n \cdot \log n)$-bit string indicating the "relevant bits" in each block of Alice's input and a $2n$-bit string w. Bob sends Alice the index of a uniformly random relevant bit using $\log(2n^2)$ bits of communication. Alice responds with her value b of that input bit, and Bob outputs $b \oplus w_i$. It is easy to check that this is a valid $\mathbf{UPP}^{\mathrm{cc}}$ protocol, and it has cost $O(\log n)$.

One can verify by the definition of pattern matrices (Definition 4) that the communication game corresponding to the $(4n^2, 4n, \mathrm{AND}_2 \circ \mathrm{MAJ}_{2n})$ pattern matrix M' equals $f \wedge f$. By Theorem 3 and Equation (16), we obtain that

$$\mathbf{UPP}(f \wedge f) = \Theta(\log \mathrm{sr}(M')) = \Omega(\log^2 n).$$

As mentioned in Section 1, the result of Klivans et al. [16] implies that $\mathrm{sr}(M') = n^{O(\log n)}$. Thus, the function f satisfies $\mathbf{UPP}^{\mathrm{cc}}(f) = O(\log n)$, but $\mathbf{UPP}^{\mathrm{cc}}(f \wedge f) = \Theta(\log^2 n)$. ◀

Corollary 2 follows immediately from the previous proof and the definition of pattern matrices.

4 Proof of Theorem 8

The rest of this paper is dedicated towards proving Theorem 8. Before proving the theorem, we describe the main auxiliary construction and prove some preliminary facts about it.

Let $\Delta = \lfloor n^{1/(3d)} \rfloor \geq 2$. Fix any $u \in \{1, \ldots, \lfloor n^{2/3} \rfloor - 1, \lfloor n^{2/3} \rfloor\}$. Define the set

$$S_u = \{\pm u, \pm u\Delta, \pm u\Delta^2, \ldots, \pm u\Delta^{d-1}\}.$$

Define the polynomial $r_u : \{0, \pm 1, \ldots, \pm n\} \to \mathbb{R}$ by

$$r_u(t) = \frac{1}{(2n)!} \prod_{i=0}^{d-1} \left(t - \left(u\Delta^i \sqrt{\Delta}\right)\right) \prod_{s \notin S_u} (t - s).$$

Since n is odd, notice that $\mathrm{sgn}(r_u(t)) = (-1)^t$, for $t \in \{u, u\Delta, u\Delta^2, \ldots, u\Delta^{d-1}\}$, and $r_u(t) = 0$ for $t \notin S_u$.

Define

$$p_u(t) = \binom{2n}{n+t} r_u(t) = \begin{cases} (-1)^{n-t} \cdot \dfrac{\prod_{i=0}^{d-1} \left(t - \left(u\Delta^i \sqrt{\Delta}\right)\right)}{\prod_{\substack{s \in S_u \\ s \neq t}} (t - s)} & \text{if } t \in S_u \\ 0 & \text{otherwise.} \end{cases}$$

The following claim tells us that for any $u \in \{1, \ldots, \lfloor n^{2/3} \rfloor\}$, the function p_u places a reasonably large mass on input $-u$.

▷ Claim 11.

$$|p_u(-u)| \geq \frac{\sqrt{\Delta}+1}{2} \cdot u^{-(d-1)} \cdot \Delta^{-(d-1)^2/2}.$$

Proof. We calculate

$$\begin{aligned}
|p_u(-u)| &= \frac{u(\sqrt{\Delta}+1)}{2u} \cdot \prod_{i=1}^{d-1} \frac{u(\Delta^i\sqrt{\Delta}+1)}{u^2(\Delta^i+1)(\Delta^i-1)} \\
&\qquad \text{(pairing terms corresponding to } u\Delta^i \text{ and } -u\Delta^i\text{)} \\
&= \frac{\sqrt{\Delta}+1}{2} \cdot u^{-(d-1)} \cdot \prod_{i=1}^{d-1} \frac{\Delta^{i+\frac{1}{2}}+1}{\Delta^{2i}-1} \geq \frac{\sqrt{\Delta}+1}{2} \cdot u^{-(d-1)} \cdot \Delta^{(d-1)/2} \cdot \prod_{i=1}^{d-1} \frac{\Delta^i}{\Delta^{2i}} \\
&= \frac{\sqrt{\Delta}+1}{2} \cdot u^{-(d-1)} \cdot \Delta^{-(d-1)^2/2}.
\end{aligned}$$

◁

The next claim tells us that the mass placed by p_u on other points in its support is small.

▷ Claim 12. For every $j = 1, 2, \ldots, d-1$,

$$|p_u(-u\Delta^j)| \leq e^4 \cdot \Delta^{-(j^2-3j-2)/2} \cdot \left(\frac{\sqrt{\Delta}+1}{2} \cdot u^{-(d-1)} \cdot \Delta^{-(d-1)^2/2} \right).$$

Proof. We calculate

$$\begin{aligned}
|p_u(-u\Delta^j)| &= \frac{u(\Delta^j\sqrt{\Delta}+\Delta^j)}{2u\Delta^j} \cdot \prod_{i=0}^{j-1} \frac{u(\Delta^i\sqrt{\Delta}+\Delta^j)}{u^2(\Delta^i+\Delta^j)(\Delta^j-\Delta^i)} \cdot \prod_{i=j+1}^{d-1} \frac{u(\Delta^i\sqrt{\Delta}+\Delta^j)}{u^2(\Delta^i+\Delta^j)(\Delta^i-\Delta^j)} \\
&\qquad \text{(pairing terms corresponding to } u\Delta^i \text{ and } -u\Delta^i\text{)} \\
&\leq \frac{\sqrt{\Delta}+1}{2} \cdot u^{-(d-1)} \cdot \prod_{i=0}^{j-1} \frac{\sqrt{\Delta}}{\Delta^j-\Delta^i} \cdot \prod_{i=j+1}^{d-1} \frac{\sqrt{\Delta}}{\Delta^i-\Delta^j} \\
&\leq \frac{\sqrt{\Delta}+1}{2} \cdot (\sqrt{\Delta} \cdot u^{-1})^{d-1} \cdot \prod_{i=0}^{j-1} \frac{\Delta^{j-i} \cdot \Delta^{-j}}{\Delta^{j-i}-1} \cdot \prod_{i=j+1}^{d-1} \frac{\Delta^{-i} \cdot \Delta^{i-j}}{\Delta^{i-j}-1} \\
&\leq \frac{\sqrt{\Delta}+1}{2} \cdot (\sqrt{\Delta} \cdot u^{-1})^{d-1} \cdot \prod_{i=0}^{j-1} \Delta^{-j} \cdot \prod_{i=j+1}^{d-1} \Delta^{-i} \cdot \left(\prod_{k=1}^{\infty} \frac{\Delta^k}{\Delta^k-1} \right)^2 \\
&\leq \frac{\sqrt{\Delta}+1}{2} \cdot (\sqrt{\Delta} \cdot u^{-1})^{d-1} \cdot \Delta^{-j^2-(d(d-1)-(j+2)(j+1))/2} \cdot \exp\left(2\sum_{k=1}^{\infty} \frac{1}{\Delta^k-1} \right) \\
&\qquad \text{(since } 1+x \leq e^x \text{ for all } x \in \mathbb{R}\text{)} \\
&\leq \frac{\sqrt{\Delta}+1}{2} \cdot u^{-(d-1)} \cdot \Delta^{-(j^2-3j-2)/2} \cdot \Delta^{-(d-1)^2/2} \cdot \exp\left(4\sum_{k=1}^{\infty} \frac{1}{\Delta^k} \right) \quad \text{(since } \Delta \geq 2\text{)} \\
&\leq e^4 \cdot \frac{\sqrt{\Delta}+1}{2} \cdot u^{-(d-1)} \cdot \Delta^{-(j^2-3j-2)/2} \cdot \Delta^{-(d-1)^2/2}. \qquad \text{(again using } \Delta \geq 2\text{)}
\end{aligned}$$

◁

The following claim tells us that for each u and j, the masses placed by r_u (and hence p_u) on $u\Delta^j$ and $-u\Delta^j$ are comparable.

▷ Claim 13. For every $j = 0, 1, \ldots, d-1$, we have

$$|r_u(-u\Delta^j)| \geq |r_u(u\Delta^j)| \geq \exp(-18/\sqrt{\Delta})|r_u(-u\Delta^j)|$$
$$\text{and } |p_u(-u\Delta^j)| \geq |p_u(u\Delta^j)| \geq \exp(-18/\sqrt{\Delta})|p_u(-u\Delta^j)|.$$

Proof. We may write the ratio

$$\frac{|p_u(u\Delta^j)|}{|p_u(-u\Delta^j)|} = \frac{|r_u(u\Delta^j)|}{|r_u(-u\Delta^j)|} = \prod_{i=0}^{j-1} \frac{u(\Delta^j - \Delta^i\sqrt{\Delta})}{u(\Delta^j + \Delta^i\sqrt{\Delta})} \cdot \prod_{i=j}^{d-1} \frac{u(\Delta^i\sqrt{\Delta} - \Delta^j)}{u(\Delta^j + \Delta^i\sqrt{\Delta})}.$$

This is a product of terms smaller than 1, yielding the first inequality. For the second, we follow Sherstov's argument [26, Theorem 5.3] and note that this product is at least

$$\left(\prod_{i=1}^{\infty} \frac{\Delta^{i/2}-1}{\Delta^{i/2}+1}\right)^2 \geq \exp\left(-5\sum_{i=1}^{\infty} \frac{1}{\Delta^{i/2}}\right)$$
since $(a-1)/(a+1) > \exp(-2.5/a)$ for $a \geq \sqrt{2}$
$$= \exp\left(\frac{-5}{\sqrt{\Delta}}\sum_{i=0}^{\infty} \frac{1}{\Delta^{i/2}}\right) \geq \exp\left(\frac{-5}{\sqrt{\Delta}} \cdot \frac{1}{1-1/\sqrt{2}}\right) \quad \text{since } \Delta \geq 2$$
$$\geq \exp\left(-\frac{18}{\sqrt{\Delta}}\right). \qquad \triangleleft$$

Putting the three claims together, we obtain the following conclusion, which states that the mass placed by p_u on $-u$ and u is a relatively large fraction of its ℓ_1-norm.

▶ **Lemma 14.** *$|p_u(-u)| \geq \|p_u\|_1/(8\Delta^2 e^4)$ and $|p_u(-u)| \geq |p_u(u)| \geq \frac{\exp(-18/\sqrt{\Delta}-4)}{8\Delta^2} \cdot \|p_u\|_1$.*

Proof. We bound the ratio

$$\frac{\|p_u\|_1}{|p_u(-u)|} \leq 2\sum_{j=0}^{d-1} \frac{|p_u(-u\Delta^j)|}{|p_u(-u)|} \quad \text{by the first inequality in Claim 13}$$
$$\leq 2\left(1 + \sum_{j=0}^{d-1} e^4 \Delta^{-(j^2-3j-2)/2}\right) \quad \text{by Claims 11 and 12}$$
$$\leq 2 + 2e^4\left(\sum_{j=0}^{3} \Delta^{-(j^2-3j-2)/2} + \sum_{j=4}^{\infty} \Delta^{-(j^2-3j-2)/2}\right)$$
$$\leq 8\Delta^2 \cdot e^4 \cdot \sum_{k=1}^{\infty} \Delta^{-k} \leq 8 \cdot \Delta^2 \cdot e^4. \quad \text{since } \Delta \geq 2$$

By the above and the second inequality in Claim 13,

$$|p_u(u)| \geq \exp(-18/\sqrt{\Delta})|p_u(-u)| \geq \frac{\exp(-18/\sqrt{\Delta}-4)}{8\Delta^2} \cdot \|p_u\|_1. \qquad \blacktriangleleft$$

We are now ready to prove Theorem 8.

Proof of Theorem 8. Define the function $P(t) = \sum_{u=1}^{\lfloor n^{2/3} \rfloor} u^{20} \cdot \frac{p_u(t)}{\|p_u\|_1}$. We claim that the function $R : \{0, \pm 1, \ldots, \pm n\} \to \{-1, 1\}$ defined by $R(t) = \frac{(-1)^t P(t)}{\|P\|_1}$ satisfies the conditions in Theorem 8.

- Clearly, $\sum_{t=-n}^{n} |R(t)| = 1$, i.e., R satisfies Equation (8).

- By Claim 13, for every $u = 1, \ldots, \lfloor n^{2/3} \rfloor$ and every $t = 1, \ldots, n$, $(-1)^t p_u(t) \geq \delta |p_u(-t)|$ for $\delta = \exp(-18/\sqrt{\Delta}) = \exp(-18/n^{(1/6d)})$. Therefore, for all such t we also have $(-1)^t P(t) \geq \delta |P(t)|$, which implies $R(t) \geq \delta |R(-t)|$ for every $t = 1, 2, \ldots, n$.
- We have

$$R(t) = \frac{(-1)^t P(t)}{\|P\|_1} = \frac{(-1)^t}{\|P\|} \sum_{u=1}^{\lfloor n^{2/3} \rfloor} u^{20} \cdot \frac{p_u(t)}{\|p_u\|_1} = \frac{(-1)^t}{\|P\|_1} \binom{2n}{n+t} \sum_{u=1}^{\lfloor n^{2/3} \rfloor} u^{20} \cdot \frac{r_u(t)}{\|p_u\|_1}.$$

  Since each r_u is a polynomial of degree at most $(2n+1) - d$, Claim 6 implies that for any polynomial p of degree at most $d-2$, $\langle R, p \rangle = 0$.
- It now remains to verify the smoothness condition. Fix a point $v \in \{1, \ldots, \lfloor n^{2/3} \rfloor\}$. Since $\operatorname{sgn}(p_u(v)) = (-1)^v$ for all u and for all $v > 0$, we have that

$$\begin{aligned} \frac{|P(v)|}{\|P\|_1} &\geq v^{20} \cdot \frac{|p_v(v)| \cdot \|p_v\|_1^{-1}}{\sum_{u=1}^{\lfloor n^{2/3} \rfloor} u^{20}} \geq \frac{\exp(-18/\sqrt{\Delta} - 4)/8\Delta^2}{\lfloor n^{2/3} \rfloor \cdot (\lfloor n^{2/3} \rfloor)^{20}} && \text{by Lemma 14} \\ &\geq \frac{\exp(-18/\sqrt{2} - 4)}{8n^{15}} \geq \frac{e^{-15}}{8n^{15}}. && \text{since } n^{1/3} \geq \Delta = \lfloor n^{1/3d} \rfloor \geq 2 \end{aligned}$$

  If $v < 0$, the argument needs some more care because we do not have the guarantee that $\operatorname{sgn}(p_u(v)) = (-1)^v$. The large mass placed by p_{-v} on the point v plays a crucial role.

$$\begin{aligned} |P(v)| &\geq \frac{(-v)^{20} \cdot |p_{-v}(v)|}{\|p_{-v}\|_1} - \sum_{\substack{u=1 \\ u \neq v}}^{\lfloor n^{2/3} \rfloor} u^{20} \cdot \frac{p_u(-v)}{\|p_u\|_1} \\ &\geq \frac{(-v)^{20}}{8\Delta^2 e^4} - \sum_{j=1}^{\lfloor \log_\Delta(-v) \rfloor} (-v\Delta^{-j})^{20} \cdot \frac{p_{-v\Delta^{-j}}(v)}{\|p_{-v\Delta^{-j}}\|_1} \\ &&& \text{by Lemma 14, the definition of } p_u \text{ and its support} \\ &\geq (-v)^{20} \left[\frac{1}{8\Delta^2 e^4} - e^4 \sum_{j=1}^{\infty} \Delta^{-20j} \cdot \Delta^{(-j^2+3j+2)/2} \right] && \text{by Claims 11 and 12} \\ &= (-v)^{20} \left[\frac{1}{8\Delta^2 e^4} - e^4 \sum_{j=1}^{\infty} \Delta^{(-j^2-37j+2)/2} \right] \geq (-v)^{20} \left[\frac{1}{8\Delta^2 e^4} - e^4 \sum_{j=1}^{\infty} \Delta^{-18j} \right] \\ &\geq (-v)^{20} \left[\frac{1}{8\Delta^2 e^4} - \frac{e^4}{\Delta^{17}} \right] = (-v)^{20} \left[\frac{\Delta^{15} - 8e^8}{8\Delta^{17} e^4} \right] \geq \frac{20}{\Delta^{17}} \geq \frac{20}{n^6} \\ &&& \text{since } n^{1/3} \geq \Delta \geq 2 \text{ and } (-v) \geq 1 \end{aligned}$$

  Thus, we have that for $v < 0$,

$$\frac{|P(v)|}{\|P\|_1} \geq \frac{20}{n^6 \sum_{u=1}^{\lfloor n^{2/3} \rfloor} u^{20}} \geq \frac{20}{n^6 \lfloor n^{2/3} \rfloor \cdot (\lfloor n^{2/3} \rfloor)^{20}} \geq \frac{20}{n^{20}}.$$

◀

5 Conclusion

We have exhibited a communication problem f with $\mathbf{UPP}^{\mathrm{cc}}(f) = O(\log n)$, and $\mathbf{UPP}^{\mathrm{cc}}(f \wedge f) = \Theta(\log^2 n)$. This is the first result showing that **UPP** communication complexity can increase by more than a constant factor under intersection. As a consequence, we have concluded that the dimension-complexity-based quasipolynomial time PAC learning algorithm of [16] for learning intersections of polylogarithmically many majorities is optimal. That is, new learning algorithms not based on dimension complexity will be required to learn this class in polynomial time.

A glaring open question left by our work is whether the class of problems with polylogarithmic $\mathbf{UPP}^{cc}$ complexity is closed under intersection. Our results represent an important first step in this direction. It would also be very interesting to extend our result that dimension-complexity-based algorithms cannot PAC learn intersections of two majorities in polynomial time, to rule out an even larger class of learning algorithms. Specifically, it would be very interesting to show that no algorithm working in the important *statistical query* model [14] can learn this concept class in polynomial time.

References

1 Andris Ambainis, Andrew M Childs, Ben W Reichardt, Robert Špalek, and Shengyu Zhang. Any AND-OR formula of size N can be evaluated in time $N^{1/2+o(1)}$ on a quantum computer. *SIAM Journal on Computing*, 39(6):2513–2530, 2010.

2 László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory (preliminary version). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 337–347, 1986. doi:10.1109/SFCS.1986.15.

3 Richard Beigel, Nick Reingold, and Daniel A. Spielman. PP Is Closed under Intersection. *J. Comput. Syst. Sci.*, 50(2):191–202, 1995. doi:10.1006/jcss.1995.1017.

4 Arnab Bhattacharyya, Suprovat Ghoshal, and Rishi Saket. Hardness of Learning Noisy Halfspaces using Polynomial Thresholds. In *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018.*, pages 876–917, 2018. URL: http://proceedings.mlr.press/v75/bhattacharyya18a.html.

5 Adam Bouland, Lijie Chen, Dhiraj Holden, Justin Thaler, and Prashant Nalini Vasudevan. On the Power of Statistical Zero Knowledge. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 708–719, 2017. doi:10.1109/FOCS.2017.71.

6 Mark Bun and Justin Thaler. Improved Bounds on the Sign-Rank of AC^0. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 37:1–37:14, 2016. doi:10.4230/LIPIcs.ICALP.2016.37.

7 Mark Bun and Justin Thaler. The Large-Error Approximate Degree of AC^0. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:143, 2018. URL: https://eccc.weizmann.ac.il/report/2018/143.

8 Arkadev Chattopadhyay and Nikhil S. Mande. A Short List of Equalities Induces Large Sign Rank. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 47–58, 2018. doi:10.1109/FOCS.2018.00014.

9 Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. New results for learning noisy parities and halfspaces. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 563–574. IEEE, 2006.

10 Jürgen Forster. A linear lower bound on the unbounded error probabilistic communication complexity. *J. Comput. Syst. Sci.*, 65(4):612–625, 2002. doi:10.1016/S0022-0000(02)00019-3.

11 Mika Göös, Pritish Kamath, Toniann Pitassi, and Thomas Watson. Query-to-Communication Lifting for P^{NP}. In *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, pages 12:1–12:16, 2017. doi:10.4230/LIPIcs.CCC.2017.12.

12 Mika Göös, Toniann Pitassi, and Thomas Watson. Query-to-Communication Lifting for BPP. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 132–143, 2017. doi:10.1109/FOCS.2017.21.

13 Mika Göös, Toniann Pitassi, and Thomas Watson. The Landscape of Communication Complexity Classes. *Computational Complexity*, 27(2):245–304, 2018. doi:10.1007/s00037-018-0166-6.

14 Michael Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)*, 45(6):983–1006, 1998.

15 Subhash Khot and Rishi Saket. On the hardness of learning intersections of two halfspaces. *Journal of Computer and System Sciences*, 77(1):129–141, 2011.

16 Adam R. Klivans, Ryan O'Donnell, and Rocco A. Servedio. Learning intersections and thresholds of halfspaces. *J. Comput. Syst. Sci.*, 68(4):808–840, 2004. doi:10.1016/j.jcss.2003.11.002.

17 Adam R Klivans and Rocco A Servedio. Learning DNF in time $2^{O(n^{1/3})}$. *Journal of Computer and System Sciences*, 68(2):303–318, 2004.

18 Adam R Klivans and Alexander A Sherstov. Cryptographic hardness for learning intersections of halfspaces. *Journal of Computer and System Sciences*, 75(1):2–12, 2009.

19 Marvin Minsky and Seymour Papert. *Perceptrons*. MIT Press, 1969.

20 Ryan O'Donnell and Rocco A Servedio. New degree bounds for polynomial threshold functions. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 325–334. ACM, 2003.

21 Ramamohan Paturi and Janos Simon. Probabilistic Communication Complexity. *J. Comput. Syst. Sci.*, 33(1):106–123, 1986. doi:10.1016/0022-0000(86)90046-2.

22 Alexander A. Razborov and Alexander A. Sherstov. The Sign-Rank of AC^0. *SIAM J. Comput.*, 39(5):1833–1855, 2010. doi:10.1137/080744037.

23 Alexander A. Sherstov. The Pattern Matrix Method. *SIAM J. Comput.*, 40(6):1969–2000, 2011. doi:10.1137/080733644.

24 Alexander A Sherstov. The unbounded-error communication complexity of symmetric functions. *Combinatorica*, 31(5):583–614, 2011.

25 Alexander A. Sherstov. Optimal bounds for sign-representing the intersection of two halfspaces by polynomials. *Combinatorica*, 33(1):73–96, 2013. doi:10.1007/s00493-013-2759-7.

26 Alexander A. Sherstov. The Intersection of Two Halfspaces Has High Threshold Degree. *SIAM J. Comput.*, 42(6):2329–2374, 2013. doi:10.1137/100785260.

27 Alexander A Sherstov. Breaking the Minsky–Papert Barrier for Constant-Depth Circuits. *SIAM Journal on Computing*, 47(5):1809–1857, 2018.

28 Alexander A. Sherstov and Pei Wu. Near-Optimal Lower Bounds on the Threshold Degree and Sign-Rank of AC^0. *CoRR*, abs/1901.00988, 2019. To appear in STOC 2019. arXiv:1901.00988.

Covert Computation in Self-Assembled Circuits

Angel A. Cantu
Department of Computer Science, University of Texas – Rio Grande Valley, USA
angel.cantu01@utrgv.edu

Austin Luchsinger
Department of Computer Science, University of Texas – Rio Grande Valley, USA
austin.luchsinger01@utrgv.edu

Robert Schweller
Department of Computer Science, University of Texas – Rio Grande Valley, USA
robert.schweller@utrgv.edu

Tim Wylie
Department of Computer Science, University of Texas – Rio Grande Valley, USA
timothy.wylie@utrgv.edu

Abstract

Traditionally, computation within self-assembly models is hard to conceal because the self-assembly process generates a crystalline assembly whose computational history is inherently part of the structure itself. With no way to remove information from the computation, this computational model offers a unique problem: how can computational input and computation be hidden while still computing and reporting the final output? Designing such systems is inherently motivated by privacy concerns in biomedical computing and applications in cryptography.

In this paper we propose the problem of performing "covert computation" within tile self-assembly that seeks to design self-assembly systems that "conceal" both the input and computational history of performed computations. We achieve these results within the growth-only restricted abstract tile assembly model (aTAM) with positive and negative interactions. We show that general-case covert computation is possible by implementing a set of basic covert logic gates capable of simulating any circuit (functionally complete). To further motivate the study of covert computation, we apply our new framework to resolve an outstanding complexity question; we use our covert circuitry to show that the unique assembly verification problem within the growth-only aTAM with negative interactions is coNP-complete.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases self-assembly, covert circuits

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.31

Category Track A: Algorithms, Complexity and Games

Funding This research was supported in part by National Science Foundation Grant CCF-1817602.

Acknowledgements We would like to thank the anonymous reviewers for their careful review of our work and for their constructive feedback.

1 Introduction

Since the discovery of DNA over half a century ago, humans have been continually working to understand and harness the vast amount of information it contains. The Human Genome Project [16], which began in 1990 and took a decade, was the first major attempt to fully sequence the human genome. In the years since, sequencing has become extremely cheap and easy, and our ability to manipulate DNA has emerged as a central tool for many applications related to nanotechnology and biomedical engineering.

Although this progress has many benefits, as we learn more about the information, we also must be careful with the shared data. There are databases of anonymous DNA sequences, which can sometimes be deanonymized with only small amounts of information such as a surname [13], or by reconstructing physical features from the DNA [6]. In order to address these issues, there has been work on cryptographic schemes aimed at obscuring results related to DNA or the input/output [7, 11, 14, 26].

In this work we take the first steps in addressing some of these issues within self-assembling systems by proposing a new style of computation termed *covert computation* with important motivations for private biomedical computing and cryptography. Self-assembly is the process by which systems of simple objects autonomously organize themselves through local interactions into larger, more complex objects. Understanding how to design and efficiently program molecular self-assembly systems is fundamental for the future of nanotechnology. The Abstract Tile Self-Assembly Model (aTAM) [8, 18], motivated by a DNA implementaiton [12], has become the premiere model for the study of the computational power of self-assembling systems. In the aTAM, system monomers are modeled by four-sided Wang tiles which randomly combine and attach if the respective bonding domains on tile edges are sufficiently strong. The aTAM is known to be computationally universal [25] and intrinsically universal [10].

Covert Computation. As a computational model, tile self-assembly differs from traditional models of computation in that computational steps are defined by permanently placing particular tile types at specific locations in geometric space. A history of each computational step is thereby recorded in the final assembled structure. This presents a unique problem to this type of computation: is it possible to conceal the input and history of a computation within the final assembly while still computing and reporting the output of the computation? Concealing the computational histories of the self-assembly process in this way requires designing a computational system which encodes computational steps in the *order* of tile placement, rather than the type and location of tile placements. We use the term *covert*[1] to describe this concealment of inputs and computational histories. This method of computing is different than previous tile self-assembly computing methods and requires novel techniques.

Also, while the reader may notice many parallels between our work and traditional secure multiparty computation [5], it should be made clear that our main result is the secure computation of a function with only a single party. The challenges presented above make this an interesting problem for tile self-assembly.

Motivation. The concept of covert computation within self-assembly has many potential applications. We briefly outline a few biomedical computing applications. Consider a set of diagnostic tiles sent to a patient as a droplet of DNA to which the patient adds some biological input such as a blood sample. From this the diagnostic system could compute some desired function that outputs specific diagnostic statistics. The patient sends the combined product to a medical facility for interpretation. With covert computation, only the results can be read by the lab and the user's biological input is obscured ensuring privacy.

Another potential use involves implementing a cryptography system within a molecular computing framework. The ability to covertly compute allows users to provide a personal key input that may be combined with a publicly available covert system where the combination

[1] It is important to note that the term *covert* has specific meaning in cryptography which does not apply here.

Table 1 The complexity of Unique Assembly Verification in the aTAM in relation to negative glues. $|A|$ refers to the size of an assembly and $|T|$ is the number of tile types.

Model	Negative Glues	Detachment	Complexity	Theorem
aTAM	No	No	$O(\|A\|^2 + \|A\|\|T\|)$	Thm. 3.2 in [1]
aTAM	Yes	No	coNP-complete	Thm. 3
aTAM	Yes	Yes	Undecidable	[9]

verifies some computable property of the input key without revealing any additional details of the key. This style of cryptographic scheme fits well when the input keys are biological based inputs.

A final potential biological application might be engineering a system for unlocking key biological properties within bio-engineered crops. For example, by releasing a hidden "key" input, covert computation might allow a field of crops to become fertile. A company owning the patent on this type of activation might desire the security of ensuring that the release key cannot be deciphered from the activated crop based on a covert molecular computation.

The final motivation of covert computation is within algorithmic self-assembly. We believe the concept of covert computation is fundamental and hope that our novel design techniques will be applicable to a number of future problems in the area. As evidence towards this, we apply our techniques to resolve the complexity of the fundamental question of verifying whether a tile system uniquely assembles a given assembly within the growth-only negative-glue aTAM.

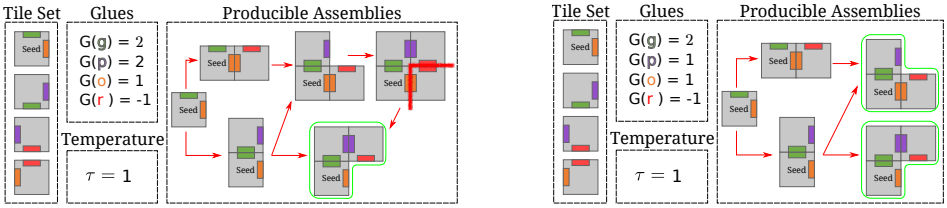
Contributions. After formally defining the concept of covert computation in tile self-assembly, we implement several covert logic gates within the negative-glue growth-only abstract Tile Assembly Model (this growth-only restriction to negative glues has been seen in the 2HAM [4], and negative glues in tile assembly have received extensive study [3, 9, 17, 20, 19, 21, 22]), and show these gates may be combined to create general circuits, thereby showing that general covert computation is possible. Finally, we apply our techniques and framework to address the fundamental problem of deciding if a negative-glue aTAM system uniquely produces a given assembly. We show this problem is coNP-complete. Table 1 outlines how our result compares to what was previously known.

2 Definitions

We begin with an overview of the Abstract Tile-Assembly Model (aTAM) and then give the new definitions introducing covert computation. Due to space constraints, we only give a high-level overview of the aTAM.

2.1 Abstract Tile Assembly Model

Figure 1 gives a high-level overview of the models with a couple of example systems. Essentially, we have non-rotating square *tiles* that have a *glue* label on each edge. The tile with its labels is a *tile type*. The *tile set* is all the tile types. A glue function determines the strength of matching glue labels. An *assembly* is a single tile or a finite set of tiles that have combined via the glues. If the combined strength of the glue labels of a single attaching tile to an assembly is greater than or equal to the *temperature* τ, the tile may attach. A *producible* assembly is any assembly that might be achieved by beginning with the *seed* (the



(a) Negative aTAM. **(b)** Growth aTAM.

Figure 1 High-level overview of the aTAM with repulsive forces. Both systems have tiles that can attach to the seed tile given they can attach with τ strength. The arrows show the possible assembly paths from the seed tile with the terminal assembly being outlined. (a) A negative aTAM system that has a possible assembly path causing disassembly. One path is growth-only, but the other path can attach the tile with the purple/red glues, which causes the orange/red tile to become unstable and detach. (b) A growth only aTAM system where negative glues are used to block, but never cause disassembly. The only difference is that the purple glue attaches with strength 1, $G(p) = 1$. This yields two possible terminal assemblies, neither of which include disassembly.

specified starting assembly) and attaching tiles. A producible assembly is further said to be *terminal* if no further tile attachment is possible. A tile system is said to *uniquely produce* a (terminal) assembly A if all producible assemblies will eventually grow into A. A tile system is formally represented as an ordered triplet $\gamma = (T, s, \tau)$ representing the tile set, seed assembly, and temperature parameter of the system respectively.

In a standard aTAM system, all glues are positive integral values, but here we look at the negative aTAM where the glues may be negative/repulsive. Such repulsive forces may be used to block the attachment of tiles despite the presence of strong attractive glues. Moreover, the inclusion of repulsive forces may yield unstable producible assemblies where a subassembly could detach because it no longer has enough binding strength. While this type of detachment has been studied in the literature [9, 22], we avoid this feature in this work as it's inclusion drastically changes the complexity of the model by making most types of verification problem undecidable, and may require more sophisticated techniques for experimental implementation. Thus, we consider a system to be a *valid growth-only* system if all producible assemblies are τ-stable. In this paper we restrict our consideration to valid growth-only systems.

2.2 Covert Computation

Here, we provide formal definitions for computing a function with a tile system, and the further requirement for covert computation of a function. Our formulation of computing functions is based on that of [15] but modified to allow for each bit to be represented by a sub-assembly potentially larger than a single tile.

Informally, a Tile Assembly Computer (TAC) for a function f consists of a set of tiles, along with a format for both input and output. The input format is a specification for how to build an input seed to the system that encodes the desired input bit-string for function f. We require that each bit of the input be mapped to one of two assemblies for the respective bit position: a sub-assembly representing "0", or a sub-assembly representing "1". The input seed for the entire string is the union of all these sub-assemblies. This seed, along with the tile set of the TAC, forms a tile system. The output of the computation is the final terminal assembly this system builds. To interpret what bit-string is represented by the output, a second *output* format specifies a pair of sub-assemblies for each bit. The bitstring represented by the union of these subassemblies within the constructed assembly is the output of the system.

For a TAC to *covertly* compute f, the TAC must compute f and produce a unique assembly for each possible output of f. We note that our formulation for providing input and interpreting output is quite rigid and may prohibit more exotic forms of computation. We acknowledge this, but caution that any formulation must take care to prevent "cheating" that could allow the output of a function to be partially or completely encoded within the input, for example. To prevent this, some type of *uniformity* constraint, similar to what is considered in circuit complexity [24], should be enforced. We now provide the formal definitions of function computing and covert computation.

Input/Output Templates. An n-bit input/output template over tile set T is a sequence of ordered pairs of assemblies over T: $A = (A_{0,0}, A_{0,1}), \ldots, (A_{n-1,0}, A_{n-1,1})$. For a given n-bit string $b = b_0, \ldots, b_{n-1}$ and n-bit input/output template A, the *representation* of b with respect to A is the assembly $A(b) = \bigcup_i A_{i,b_i}$. A template is valid for a temperature τ if this union never contains overlaps for any choice of b, and is always τ-stable. An assembly $B \supseteq A(b)$, which contains $A(b)$ as a subassembly, is said to represent b as long as $A(d) \nsubseteq B$ for any $d \neq b$.

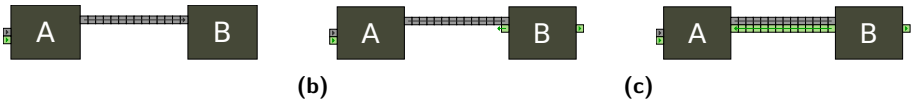
Function Computing Problem. A *tile assembly computer* (TAC) is an ordered quadruple $\Im = (T, I, O, \tau)$ where T is a tile set, I is an n-bit input template, and O is a k-bit output template. A TAC is said to compute function $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^k$ if for any $b \in \mathbb{Z}_2^n$ and $c \in \mathbb{Z}_2^k$ such that $f(b) = c$, then the tile system $\Gamma_{\Im,b} = (T, I(b), \tau)$ uniquely assembles a set of assemblies which all represent c with respect to template O.

Covert Computation. A TAC *covertly* computes a function $f(b) = c$ if 1) it computes f, and 2) for each c, there exists a unique assembly A_c such that for all b, where $f(b) = c$, the system $\Gamma_{\Im,b} = (T, I(b), \tau)$ uniquely produces A_c. In other words, A_c is determined by c, and every b where $f(b) = c$ has the exact same final assemby.

3 Covert Circuits

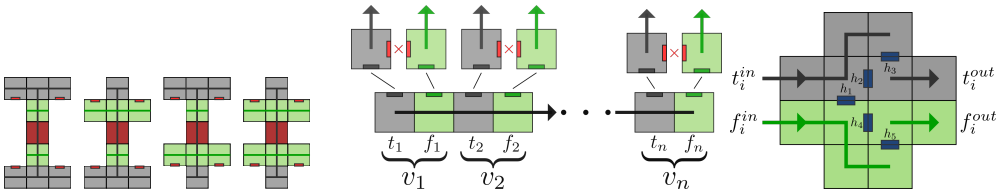
Here we cover the machinery for making covert gadgets and the covert gadgets needed for functional completeness in circuits based on a dual-rail logic implementation: variables, wires, fanouts, and NANDs. We cover a NOT gate as a primitive used in the NAND construction. Traditionally, a crossover is also given, and we discuss why this is unnecessary in Section 4. For simplicity, we give some other common gates in Section 5.

Some Conventions. All solid lines through two neighboring tiles indicate strength-2 glues between them. The arrows indicate the build order (which may branch). Blue single glues are strength 1, and red are strength -1. Following the variable gadget (Figure 3b), all variables have a true and false path adjacent to each other (dual-rail logic), but only one may be traversed at a time until the next gadget. The true value is always to the left or on top of the false value, and for most gadgets, the true input is colored grey while the false input is colored green. Once a variable wire, true or false, reaches the next gadget, the unused variable wire is *backfilled* so that both wires are present. This is a key concept used in all constructions and is further explained in Figure 2.



(a) (b) (c)

Figure 2 Backfilling in covert computation. Given two gadgets A and B. (a) If true is output from Gadget A, that wire assembles to the next gadget. (b) Gadget B builds, and based on its function, outputs the true or false wire (false in this case). Once it received the input, it backfills the false wire towards A. (c) The false wire finishes assembling and both Gadget A and B have true and false paths filled. The true output wire of Gadget B will be backfilled from the next gadget. In this way, the input to B/output from A is "hidden".



(a) Possible Input Seeds. **(b)** Variable. **(c)** Logic Diode.

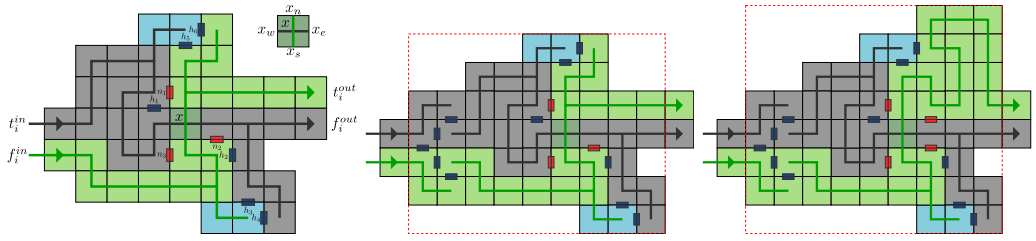
Figure 3 (a) Example of the 4 possible input seeds for a half-adder from Section 5.1. (a) Variables are represented by a true and a false line where only one may exist. The variables build off the seed, but only the t_i or the f_i tile may attach due to the negative glue between the two tiles. (b) A gadget referred to as a logic diode. This ensures input from one direction and stops tiles from assembling in the wrong direction.

3.1 Variables and Wires

A variable in our system is represented by two lines of connected tiles where only one exists at a time when the wire is in use (dual rail). Figure 3a shows an example of the possible input seeds on 2-bits used in a half-adder. Figure 3b demonstrates how the variables might be set nondeterministically, although generally the specific bits desired would already be attached as part of the input seed (as in Figure 3a). Each variable v_i has a sequence of tiles t_i representing a true setting and f_i a false setting. The first tiles have a negative glue of strength -1 meaning only the t_i or the f_i tile can attach. The other shown glues are strength 2. Once the variable is set, the setting travels to the gadget as a *wire*.

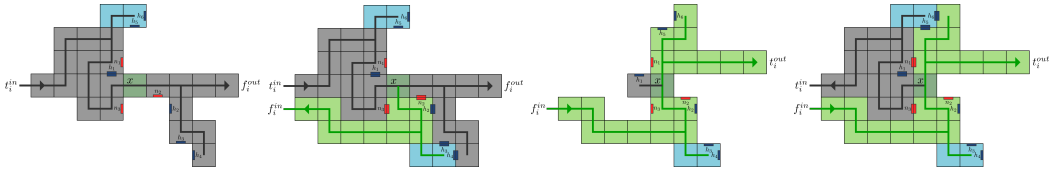
The variable setup in Figure 3b is used in one of two ways: In the case of providing an input to a covert computation, this variable setup defines the *input template* for the computation, with the seed for a given binary input being the seed assembly with either a true or false tile (but not both) placed at each bit position. An example system (a half-adder) with a big seed input is shown in Section 5. Alternatively, the seed begins as a single seed tile that nondeterministically creates a valid input over all possible n-bit inputs. This approach is used in Section 4 to show coNP-completeness for unique assembly verification.

Figure 3c shows what we refer to as a *logic diode*, and prevents timing issues. These appear in every gadget and serves two purposes: if backfilling, this stops the filling at the gadget level so it does not backfill a wire that has not been set, and second it ensures that a gadget must have input from the wire. All shown glues are strength 1 and the lines are strength 2. This gadget is important for later constructions.



(a) Basic NOT. **(b)** NOT. **(c)** H-NOT.

Figure 4 (a) Basic NOT gate (b) NOT gate with the logic diode on the input (c) A covert NOT gate with an additional negative horizontal glue on the output to prevent incorrect backfilling. This modification is needed when using this gate for the construction of the NAND gate.



(a) True Input 1. **(b)** True Input 2. **(c)** False Input 1. **(d)** False Input 2.

Figure 5 (a) A NOT gadget with true input t_i^{in} and output f_i^{out}. The true output can not place from tile x due to the negative glues n_1 and n_3 of strength -1. (b) Once the NOT gadget passes the false output, glues h_3, h_4 cooperatively allow the false portion and wire to backfill. Glue h_2 is needed to fill in the tile with n_2. (c) A NOT gadget with false input f_i^{in} and output t_i^{out}. The false output can not attach due to the negative glue n_2. The tile to the west of x may attach, but due to glue n_3, no other tile can attach. (d) Once the NOT gadget passes the true output, glues h_5, h_6 allow the true portion and wire to backfill. Glue h_1 is needed to counteract the n_1 glue when backfilling that tile.

3.2 Covert NOT Gadget

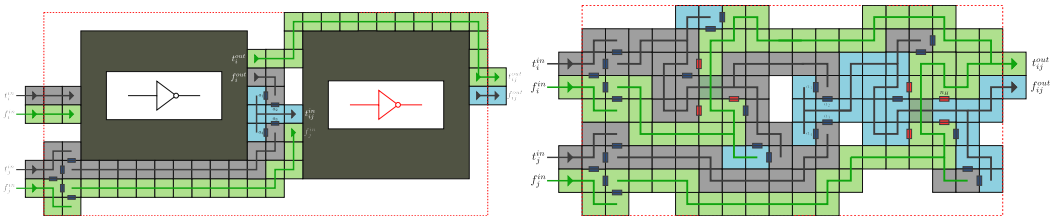
The first covert gadget we introduce is a NOT gadget. This gadget displays some of the key insights needed for covert computation, such as how blocking with negative glue adds power to the system. The NOT gadget is also used as a submodule within our NAND gadget. The NOT gadget in Figure 4a is the basic gadget with 4b only adding the logic diode on the input to ensure no backfill happens past the gadget and that the gadget had input.

Given the variables and wires work as shown, the difficulty in a dual-rail NOT is that there must be at least one crossing tile that both the true and false paths place. This tile can be thought of as where the signals cross or switch. Figure 4a shows the basic NOT gadgets, and the tile shared by both paths is labelled x. The negative glues allow blocking around this tile so that only one path is possible once x is placed.

The properies of the not gadget guarantee that it works correctly and that the gadget is covert (the gadget looks indistinguishable before the output regardless of the input), and that the backfill works correctly. Figure 5 discusses these elements and walks through how the true/false inputs block and crossover correctly. The Figure does not show the logic diode though.

3.3 Covert NAND Gadget

The basic idea for the NAND gadget is to flip one of the inputs using a covert NOT, and then we can compare the two true input lines to see if both inputs were true. Since a NAND is false only when both inputs are true, this is the only path that should result in a false



(a) NAND Block Diagram. **(b)** NAND.

Figure 6 (a) Diagram of the covert NAND gate with NOTs shown as blocks. The boxes for the NOT blocks are shown outlined in Figures 4b and 4c. The left box is the standard NOT gadget and the right box is the H-NOT gadget. (b) The full NAND gate with the two NOT gadgets filled in and compacted.

output. The basic idea for the gadget is shown in Figure 6a with a representative block for the NOT gadget already discussed. The second NOT block is the modified NOT gadget (H-NOT) from Figure 4c. Both false inputs are routed to the true output. One must go through another NOT in order to flip to the top output position, while the other false line skips this NOT and ties directly to the true output.

Once we flip the top input, we can use cooperative binding to compare the two true inputs, and only if both are true do we send it as true into the second NOT block (so the gadget outputs false). All other input combinations output true.

We will show why NOT and H-NOT are both necessary. Looking at Figure 6b, the negative glue n_H is necessary in H-NOT to ensure that t_i^{out}, which skips the second NOT gadget, does not set the output t_{ij}^{out}, and then also set f_{ij}^{out} based on the assembly order. Essentially, this protects from incorrect backfilling and setting both outputs. However, the n_H glue should not exist in the standard NOT gadget, or it may backfill and could cause a tile to break off depending on build order. Given we want a purely growth model, this would not be allowed. It is possible to create a single NOT that incorporates these properties, but we prefer to avoid the added complexity.

Finally, the logic diodes on the inputs (Figure 3c) ensure that if we only have one input, the gadget does not backfill down the other input wire. Even if the gadget has already been set, that input will wait until either the true or false wire comes before backfilling the wire.

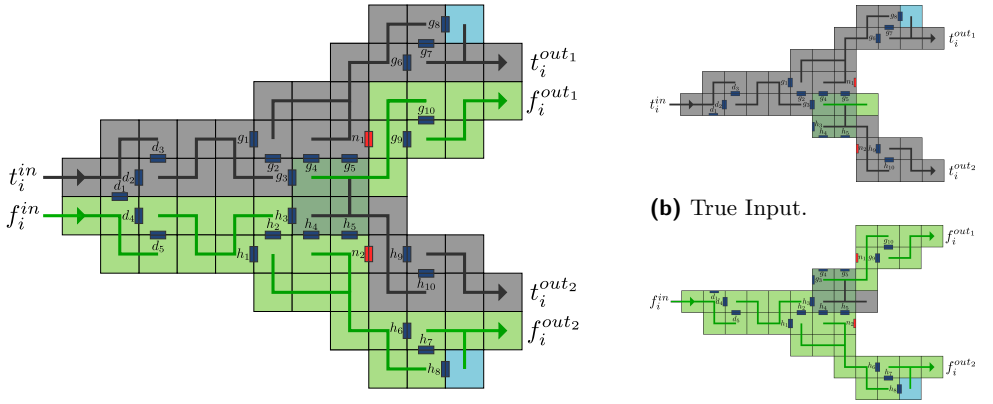
3.4 Covert FANOUT Gadget

The FANOUT gadget needs to duplicate the geometric wire, and also needs to only backfill once both outgoing wires have backfilled. Figure 7a shows the FANOUT gadget. Similar to the NOT, there is a shared set of tiles placed by both the true and the false path. Figures 7b and 7c show the true and false paths without any backfilling, respectively.

4 Covert Computation and Unique Assembly Verification

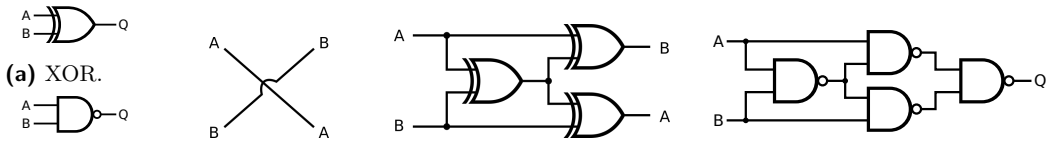
In this section we establish our main results related to covert computation in self-assembly systems. We first utilize our covert circuitry to show that any function is covertly computable (Thm. 1). We then apply covert circuitry to show that the open problem of Unique Assembly Verification within the growth-only negative glue aTAM is coNP-complete (Thm. 3).

▶ **Theorem 1.** *For any function f computed by a boolean circuit, there exists a tile assembly computer (TAC) that covertly computes f.*



(a) FANOUT. **(b)** True Input. **(c)** False Input.

Figure 7 (a) FANOUT gadget. (b) True input wire for the FANOUT gadget t_i^{in} results in output wires $t_i^{out_1}$ and $t_i^{out_2}$. (c) False input wire for the FANOUT gadget f_i^{in} results in output wires $f_i^{out_1}$ and $f_i^{out_2}$.



(a) XOR. **(b)** NAND. **(c)** Crossover. **(d)** XOR Crossover. **(e)** XOR with NANDs.

Figure 8 Constructing planar crossover gadgets with NAND gates. (a) XOR symbol. (b) NAND symbol. (c) Two wires in a circuit that cross making it non-planar. (d) A planar circuit using XOR gates that act as a crossover. (e) A planar circuit using only NAND gates that implement an XOR gate.

Proof. The proof of this theorem consists of a direct simulation of boolean circuits by way of a series of covert gadget implementations for various logic gates and how to connect them. The proof follows from the gadgets and machinery given in Section 3. ◀

We now prove that Unique Assembly Verification (UAV) in a growth-only negative glue aTAM system is coNP-complete by utilizing our covert gadgets. Without the growth-only constraint, UAV in the atam with negative glues is undecidable as a Turing machine simulation could use negative interactions to break down produced assemblies into a final unique terminal assembly exactly when the Turing machine halts [9]. With no negative glues however, the problem is in P [1]. We prove that with the ability to temporarily block, the problem becomes coNP-complete. This result is achieved with a reduction from Circuit SAT. Unique Assembly Verification in our model is formally defined as follows:

▶ **Problem** (Unique Assembly Verification (growth only)). *Given a tile-system* $\Gamma = (T, S, \tau)$ *with the promise that it is a growth-only system, and an assembly* A*. Does* Γ *uniquely assemble* A*?*

A reduction from Circuit SAT generally requires a functionally universal set of gates and variable, wire, fanout, and crossover gadgets. Both NAND and NOR are functionally complete gates, so given either, all gates can be made. A crossover gadget is redundant since it can be made with XOR gates and XOR gates can be made with NAND gates [23]. Figure 8 shows this derivation. Finally, Circuit SAT requires a DAG, and thus there are no cycles,

and so the gadgets can be topologically sorted so that there are no crossovers that cause a loop (the output of a gadget can not crossover one of its input lines). Thus, a reduction from Planar Circuit SAT is equivalent to a reduction from Circuit SAT.

▶ **Definition 2** (Planar Circuit SAT). *Instance: A planar directed acyclic graph (DAG) $G = (V, E)$ with n boolean inputs, one output, and all gates are NAND gates (or NOR gates). Every $v \in V$ is either a NAND gate ($deg^-(v) = 2$, $deg^+(v) = 1$) or a fanout ($deg^-(v) = 1$, $deg^+(v) = 2$). The source vertices, $v_i \in V$ s.t. $deg^-(v_i) = 0$ and $1 \leq i \leq n$, are the variables. The sink vertex, $s \in V$ s.t. $deg^+(v_i) = 0$ is the "output" of the boolean circuit.*
Question: Does there exist a setting of the inputs such that the output to the circuit is 1?

▶ **Theorem 3.** *Unique Assembly Verification in the aTAM with repulsive forces in a growth only system is coNP-complete.*

Proof. We first observe that Unique Assembly Verification with repulsive forces is in coNP as any failure to uniquely assemble a target assembly A comes in the form of a polynomially sized assembly that is inconsistent with A. The producibility of this assembly can be verified in polynomial time, and thus serves as a certificate for "no" instances to the UAV problem.

We now show coNP-hardness by a reduction from Planar Circuit SAT. Given an instance of planar Circuit SAT C with inputs $i_1, \ldots, i_n$ where $i \in \{0, 1\}$, i.e., a boolean circuit. By our definition we assume there are only NAND gates, fanouts, input variables and an ouput variable in the planar DAG.

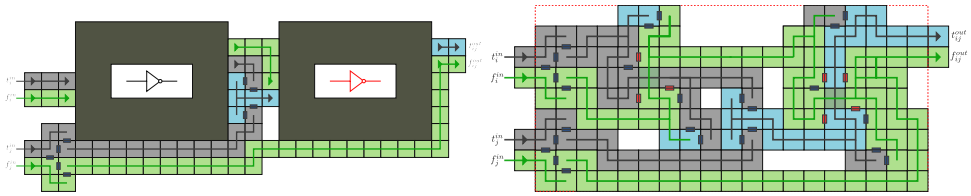
For our reduction, we build a tileset T by adding tiles corresponding to the covert gadgets and connections described in Section 3. Replace each NAND gate with a unique set of tiles implementing a NAND gadget, and each FANOUT gate with a unique set of tiles implementing a FANOUT gadget. For each edge, a unique sequence of tiles is added to T that connects the two gadgets representing the two gates the edge connected.

This yields a tile assembly computer (TAC), $\Im = (T, I, O, \tau)$, for covertly computing the circuit C. The key modification to show coNP-hardness is the utilization of a seed that non-deterministically grows any one of the possible n-bit input seeds for this TAC, and then evaluates the circuit. If the circuit is not-satisfiable, then the final computation will be false regardless of the guessed input, and therefore will yield the unique "no" assembly of the TAC based on the fact that the circuit is computed covertly. On the other hand, if there exists some satisfying n-bit input, there will be at least one final assembly that differs from the "no" assembly. Thus, the "no" assembly is uniquely produced if and only if the circuit C is not satisfiable, thereby showing coNP-hardness.

Non-deterministic input selection. To non-deterministically form the possible input bits, we include the tile types and seed tile described in Figure 3b. The seed grows a length $O(n)$ line with each bit being encoded by a pair of adjacent locations which expose a glue on the north edge. For each pair of positions, the presence of the left tile denotes a "1" for the respective bit, and the placement of the right tile denotes a "0". The "1" and "0" tiles share a negative strength 1 glue, making their mutual placement impossible until the covert gadgets have passed on the computed signal and backfilled. ◀

Given that UAV is coNP-complete with negative glues by way of covert circuitry, yet UAV is in P without negative glues [1], it is reasonable to conjecture that the use of negative interactions is needed to perform covert computation.

▶ **Conjecture.** *For some function f computed by a boolean circuit, there does not exist a tile assembly computer (TAC) that covertly computes f in the aTAM without negative glues.*



(a) AND Block Diagram. **(b)** AND.

Figure 9 (a) Diagram of the covert AND gate with NOTs shown as blocks. The left box is the standard NOT gadget and the right box is the V-NOT gadget (has an additional vertical glue). (b) The full AND gate with the two NOT gadgets filled in and some simplification for space.

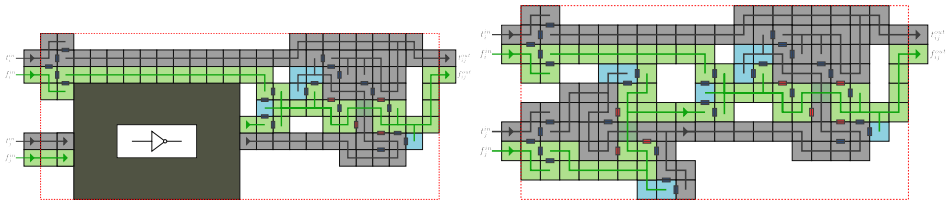
5 Further Motivation

Here, we give a few more motivating examples and some simplified gadgets. There is a lot of future work in this vein of research that is extremely relevant to modern society. We first cover the covert AND and OR gadgets.

Simplified Gadgets. Even though NAND gates alone are functionally complete, for some gates the circuit is larger than desired. Here, we give compact direct versions of some other useful gadgets and gates. This does not affect the complexity, but does help build a more efficient covert computation toolkit.

Covert AND Gadget. The covert AND gadget is nearly identical to the NAND gadget. The only real difference is which two inputs the second NOT takes in. Also, similar to the H-NOT needed for the NAND, we create a V-NOT, which is a NOT with one additional vertically aligned negative glue. Figure 9a shows the AND gadget with the blocks in place of NOTs for clarity, and Figure 9b shows the full gadget.

Covert OR Gadget. The covert OR gadget still uses a NOT to flip one of the inputs, but does several checks on the second flip to the point of drastically differing from a NOT. Figure 10a shows the AND gadget with the blocks in place of NOTs for clarity, and Figure 10b shows the full gadget.

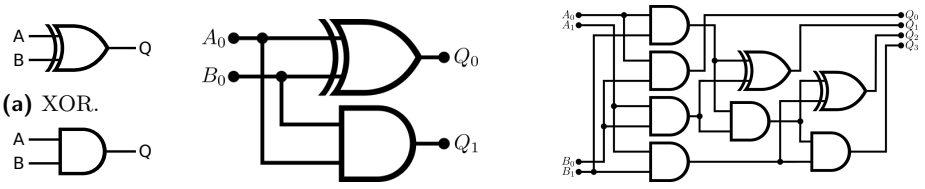


(a) Block OR. **(b)** OR.

Figure 10 (a) Block diagram for the OR gadget. (b) The covert OR gadget with the NOT gadget filled in.

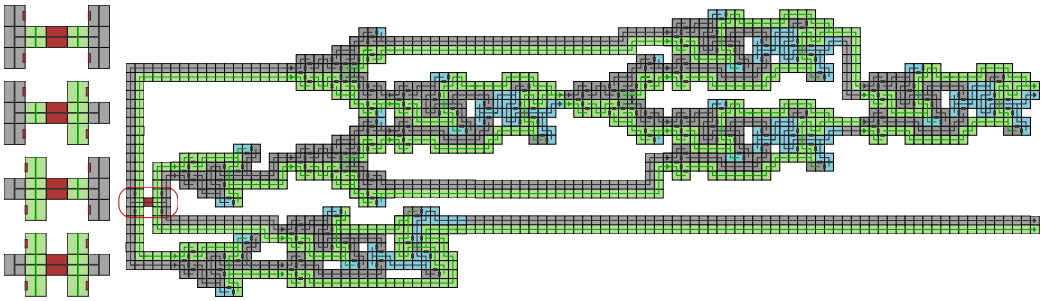
5.1 Encryption and Cryptography

Several encryption methods are based off problems that we believe to be "hard" computationally. One of the most common is factoring the product of large prime numbers, which is the basis for several encryption schemes. Although factoring may be difficult, the function



(a) XOR. **(b)** AND. **(c)** Half-Adder. **(d)** 2-Bit Multiplier.

Figure 11 Constructing covert circuits for arithmetic building up to cryptography examples. (a) XOR symbol. (b) AND symbol. (c) A half-adder, which has two 1-bit numbers as input and a 2-bit number as output. (d) A 2-bit multiplier which has two 2-bit numbers as input and outputs a 4-bit number that is their product. This can be expanded to use two large primes resulting in a large number that would be hard to factor.



(a) Seed. **(b)** Covert Half-Adder.

Figure 12 Covert Half-Adder made with 4 NANDs, 3 FANOUTs, 2 NOTs, and 1 AND. The seed input is highlighted and all 4 possible seeds are shown in (a). Regardless of the seed, the final assembly will look identical except the final T/F representing the bits of the numbers added. This implements the schematic shown in Figure 11c and the XOR is implemented with NANDs as shown in Figure 8e.

to generate the number is simple multiplication, which can be accomplished with simple circuits. Figure 11d shows a simple 6-gate circuit implementing a 2-bit number multiplier resulting in a 4-bit output number. An n-bit multiplier scales linearly (in the number of bits) with additional AND gates and full and half adders.

Implementing the multiplier with covert gates is not difficult, but the resulting assembly is large due to the inefficient crossover gadget used. Instead, we demonstrate a simple half-adder. The schematic for a half-adder is in Figure 11c. A covert half-adder as a TAC is shown in Figure 12b. The XOR has been replaced by the 4 NAND gates as shown in Figure 8e. Further, 3 FANOUTs were needed, an AND gadget as shown above in Section 5, and 2 NOT gadgets were used to flip the input for the gadgets. Figure 12a shows the four possible input seeds to build the assembly. A half-adder is simple enough to know which seed was used if 00 or 10 are output, but if 01 is output there is no way to know.

6 Conclusions and Future Work

We have introduced the concept of covert computation in self-assembly and provided a general scheme to implement any boolean circuit under this restriction. Beyond potential applications to biomedical privacy, cryptography, and intellectual property, our techniques and framework promise to impact self-assembly theory itself. As a first example we have

applied our techniques to the fundamental problem of Unique Assembly Verification in the negative glue aTAM, and shown it to be coNP-complete with growth-only systems, essentially as a corollary of our covert computation theory.

A number of future directions stem from our work. Having established the general computation power of covert computation, a natural next step is the consideration of efficiency for computing classes of functions. The *time* complexity of self-assembly computation has been studied [2, 15] and shown to allow for a substantial amount of parallelism. Can similar results be achieved under the covert constraint? What general connections exists between the time complexity for unrestricted self-assembly computation versus that of covert computation? Other natural metrics include minimizing the number of distinct tile types, along with the space taken up by the final assembly of the computation.

References

1 Leonard M. Adleman, Qi Cheng, Ashish Goel, Ming-Deh A. Huang, David Kempe, Pablo Moisset de Espanés, and Paul W. K. Rothemund. Combinatorial optimization problems in self-assembly. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 23–32, 2002.

2 Yuriy Brun. Arithmetic computation in the tile assembly model: Addition and multiplication. *Theoretical Comp. Sci.*, 378:17–31, 2007.

3 Cameron Chalk, Erik D. Demiane, Martin L. Demaine, Eric Martinez, Robert Schweller, Luis Vega, and Tim Wylie. Universal Shape Replicators via Self-Assembly with Attractive and Repulsive Forces. In *Proc. of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'17)*, 2017.

4 Cameron Chalk, Austin Luchsinger, Robert Schweller, and Tim Wylie. Self-Assembly of Any Shape with Constant Tile Types using High Temperature. In *Proc. of the 26th Annual European Symposium on Algorithms*, ESA'18, 2018.

5 David Chaum, Claude Crépeau, and Ivan Bjerre Damgård. Multiparty Unconditionally Secure Protocols (Abstract). In *Proc. of the 20th Annual ACM Symposium on Theory of Computing (STOC'88)*, pages 11–19, 1988.

6 Peter Claes, Denise K. Liberton, and Katleen et al. Daniels. Modeling 3D Facial Shape from DNA. *PLOS Genetics*, 10(3):1–14, March 2014. doi:10.1371/journal.pgen.1004224.

7 Emiliano De Cristofaro, Sky Faber, and Gene Tsudik. Secure Genomic Testing with Size- and Position-hiding Private Substring Matching. In *Proc. of the 12th ACM Workshop on Privacy in the Electronic Society*, WPES'13, pages 107–118. ACM, 2013.

8 David Doty. Theory of Algorithmic Self-Assembly. *Communications of the ACM*, 55(12):78–88, 2012.

9 David Doty, Lila Kari, and Benoît Masson. Negative Interactions in Irreversible Self-assembly. *Algorithmica*, 66(1):153–172, 2013.

10 David Doty, Jack H. Lutz, Matthew J. Patitz, Robert Schweller, Scott M. Summers, and Damien Woods. The Tile Assembly Model is Intrinsically Universal. In *Proc. of the 53rd IEEE Conf. on Foun. of Comp. Sci.*, FOCS '12, 2012.

11 N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. Manual for Using Homomorphic Encryption for Bioinformatics. *Proceedings of the IEEE*, 105(3):552–567, March 2017.

12 Constantine Evans. *Crystals that Count! Physical Principles and Experimental Investigations of DNA Tile Self-Assembly*. PhD thesis, California Inst. of Tech., 2014.

13 Melissa Gymrek, Amy L. McGuire, David Golan, Eran Halperin, and Yaniv Erlich. Identifying Personal Genomes by Surname Inference. *Science*, 339(6117):321–324, 2013.

14 Z. Huang, E. Ayday, J. Fellay, J. Hubaux, and A. Juels. GenoGuard: Protecting Genomic Data against Brute-Force Attacks. In *2015 IEEE Symposium on Security and Privacy*, pages 447–462, May 2015. doi:10.1109/SP.2015.34.

15 Alexandra Keenan, Robert Schweller, Michael Sherman, and Xingsi Zhong. Fast arithmetic in algorithmic self-assembly. *Natural Computing*, 15(1):115–128, March 2016.

16 Eric S. Lander, Lauren M. Linton, and Bruce Birren et al. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, February 2001.

17 Austin Luchsinger, Robert Schweller, and Tim Wylie. Self-assembly of shapes at constant scale using repulsive forces. *Natural Computing*, August 2018. doi:10.1007/s11047-018-9707-9.

18 Matthew J. Patitz. An introduction to tile-based self-assembly and a survey of recent results. *Natural Computing*, 13(2):195–224, June 2014.

19 Matthew J. Patitz, Trent A. Rogers, Robert Schweller, Scott M. Summers, and Andrew Winslow. Resiliency to Multiple Nucleation in Temperature-1 Self-Assembly. In *Proc. of DNA Computing and Molecular Programming*, DNA'16, pages 98–113, 2016.

20 Matthew J. Patitz, Robert T. Schweller, and Scott M. Summers. Exact Shapes and Turing Universality at Temperature 1 with a Single Negative Glue. In *DNA Comp. and Molecular Prog.*, volume 6937 of *LNCS*, pages 175–189. Springer, 2011.

21 John H. Reif, Sudheer Sahu, and Peng Yin. Complexity of graph self-assembly in accretive systems and self-destructible systems. *Theoretical Comp. Sci.*, 412(17):1592–1605, 2011.

22 Robert Schweller and Michael Sherman. Fuel Efficient Computation in Passive Self-Assembly. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'13, pages 1513–1525. SIAM, 2013.

23 Allan Scott, Ulrike Stege, and Iris van Rooij. Minesweeper May Not Be NP-Complete but Is Hard Nonetheless. *The Mathematical Intelligencer*, 33(4):5–17, December 2011.

24 Heribert Vollmer. *Introduction to Circuit Complexity: A Uniform Approach*. Springer-Verlag, Berlin, Heidelberg, 1999.

25 Erik Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, California Institute of Technology, June 1998.

26 Jing Yang, Jingjing Ma, Shi Liu, and Cheng Zhang. A molecular cryptography model based on structures of DNA self-assembly. *Chinese Science Bulletin*, 59(11):1192–1198, April 2014. doi:10.1007/s11434-014-0170-4.

Randomness and Intractability in Kolmogorov Complexity

Igor Carboni Oliveira
Department of Computer Science, University of Oxford, UK
igor.carboni.oliveira@cs.ox.ac.uk

Abstract

We introduce *randomized* time-bounded Kolmogorov complexity (rKt), a natural extension of Levin's notion [24] of Kolmogorov complexity. A string w of low rKt complexity can be decompressed from a short representation via a time-bounded algorithm that outputs w with high probability.

This complexity measure gives rise to a decision problem over strings: MrKtP (The Minimum rKt Problem). We explore ideas from pseudorandomness to prove that MrKtP and its variants cannot be solved in randomized quasi-polynomial time. This exhibits a natural string compression problem that is provably intractable, even for randomized computations. Our techniques also imply that there is no $n^{1-\varepsilon}$-approximate algorithm for MrKtP running in randomized quasi-polynomial time.

Complementing this lower bound, we observe connections between rKt, the power of randomness in computing, and circuit complexity. In particular, we present the first hardness magnification theorem for a natural problem that is unconditionally hard against a strong model of computation.

2012 ACM Subject Classification Theory of computation

Keywords and phrases computational complexity, randomness, circuit lower bounds, Kolmogorov complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.32

Category Track A: Algorithms, Complexity and Games

Acknowledgements I am grateful to Ján Pich, Eric Allender, Ryan Williams, Shuichi Hirahara, Michal Koucký, Rahul Santhanam, and Jan Krajíček for discussions. Part of this work was completed while the author was visiting the Simons Institute for the Theory of Computing. This work was supported in part by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2014)/ERC Grant Agreement no. 615075.

1 Introduction

The Kolmogorov complexity of a string w is the length of the shortest program that prints w. This concept has found connections to a variety of topics in mathematics and computer science. Notably, Kolmogorov complexity can be used to derive Gödel's incompleteness theorems (see e.g. [12, 20, 22] and references therein), and the associated *incompressibility method* has numerous applications in areas such as graph theory, combinatorics, probability, and number theory (see [25] for a comprehensive treatment of the subject).

It is well known that computing the Kolmogorov complexity of a string is undecidable. Indeed, it is easy to see that if it were computable, then it would be possible to inspect all strings of length n and print the first string z that has complexity at least n. The resulting program provides a shorter description of z, which is contradictory.

Despite its many applications, the uncomputability of Kolmogorov complexity can render it useless in situations where an upper bound on the running time of algorithms is desirable. A time-bounded variant of Kolmogorov complexity introduced by Levin [24] has been very influential in algorithms and complexity theory (see e.g. [1, 2, 13]). In Levin's definition, the complexity of a string w takes into account not only the description length of a program generating w, but also its running time. A bit more formally, we use $\mathsf{Kt}(w)$ to denote the

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 32; pp. 32:1–32:14


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

minimum over $|M| + \log t$, where M is a machine that prints w when it computes for t steps. The choice of $\log t$ in this definition can be justified by its applications in theory of computation, such as Levin's optimal universal search (see [1]).

Kolmogorov complexity and Levin complexity are important measures of the "randomness", or "information", of a string. But while the computability aspects of Kolmogorov complexity are well understood, the complexity-theoretic aspects of time-bounded Kolmogorov complexity remain mysterious. It is easy to see that $\mathsf{Kt}(w)$ can be computed in exponential time $2^{O(|w|)}$. Note however that the argument presented above for the uncomputability of Kolmogorov complexity simply does not work when one takes into account running time.

Let MKtP (The Minimum Kt Problem) denote the problem of deciding the Kt complexity of an input string. The question of whether $\mathsf{MKtP} \in \mathsf{P}$ was explicitly posed in [4]. There is evidence that the problem is hard, since under standard cryptographic assumptions it follows that $\mathsf{MKtP} \notin \mathsf{P}$. The best known upper bound on the complexity of MKtP is its inclusion in $\mathsf{E} = \mathsf{DTIME}[2^{O(n)}]$. Since it is known that $\mathsf{E} \nsubseteq \mathsf{P}$ by the deterministic time hierarchy theorem, unconditionally proving that $\mathsf{MKtP} \notin \mathsf{P}$ might be within reach of existing techniques.

In this work, we investigate time-bounded Kolmogorov complexity in the presence of *randomness*. More precisely, we consider a natural extension of Kt complexity obtained when one allows the algorithm generating the string to be randomized. The only requirement is that it generates the desired string (in some fixed time bound t) with high probability. Thus we let $\mathsf{rKt}(w)$ denote the minimum over $|M| + \log t$, where M is a *probabilistic* machine that prints w with probability at least $2/3$ when it computes for t steps.

This extension of Kt complexity is motivated from several perspectives. First, it is in line with the ubiquitous role of probabilistic algorithms in modern theoretical computer science. Second, it allows many results on time-bounded Kolmogorov complexity to be extended to the randomized setting. (For instance, it is not hard to see that if $\mathsf{SAT} \in \mathsf{BPTIME}[t]$, then every satisfiable formula ϕ admits a satisfying assignment of (conditional) rKt complexity at most $O(\log t + \log |\phi|)$.[1] This allows one to define an optimal *randomized* universal search, in the spirit of Levin's result [23].) Moreover, rKt complexity can be interpreted as an extension of Kt complexity to the *pseudodeterministic* setting (see [15] and papers citing this reference), an active research direction in algorithms and complexity. Finally, by interpreting time-bounded Kolmogorov complexity as a measure of data compression, it becomes rather natural to admit representations that can be decoded via randomized algorithms. This might allow better compression rates and faster decompression procedures.[2]

Several basic questions pose themselves: What is the computational complexity of deciding $\mathsf{rKt}(w)$? Does randomization provide better compression, in the sense that $\mathsf{rKt}(w)$ might be substantially smaller than $\mathsf{Kt}(w)$ for some strings w? How does rKt and its associated decision problem relate to the complexity of deciding MKtP?

In addition to putting forward the concept of randomized time-bounded Kolmogorov complexity, our work contains the following contributions.

---

[1] It is possible to use the assumption to find the lexicographic first satisfying assignment of ϕ given the description of ϕ in probabilistic time $\mathsf{poly}(t, |\phi|)$.

[2] While the definition of rKt appears to be rather natural in hindsight, to our knowledge it has not been previously considered in the literature, despite the many variants of time-bounded Kolmogorov complexity investigated in other works (see e.g. [4, 5]). Intuitively, allowing randomness in the computation is somewhat counter-intuitive, given that Kolmogorov complexity tries to capture how far from random the output string is. This may explain in part why this concept had not been identified before this work. It is worth noting that our definition is influenced by the emerging area of pseudodeterministic algorithms. Indeed, rKt is a candidate definition for the "pseudodeterministic complexity" of a string. This might explain why defining rKt is more evident at this point compared to previous works.

Our Results. In order to state our results in a general form, we let rKt_λ denote the minimum over $|M| + \log t$, where M is a probabilistic machine that prints w with probability at least λ when it computes for t steps.[3] We let $\mathsf{MrKtP}[\beta, \alpha, s]$ denote the promise problem of distinguishing whether $\mathsf{rKt}_\beta(w) \leq s(|w|)$ or $\mathsf{rKt}_\alpha(w) > s(|w|)$, where $1/2 < \alpha \leq \beta < 1$ and $s \colon \mathbb{N} \to \mathbb{N}$. The problem is easier the larger the gap between α and β, but our lower bound applies to all settings of the two parameters.

It is not hard to prove that MrKtP can be solved in randomized exponential time if $\alpha < \beta$. Our main technical result is the following *unconditional* lower bound.

▶ **Theorem 1.** *Let* $1/2 < \alpha \leq \beta < 1$ *and* $s(n) = n^\gamma$*, where* $0 < \gamma < 1$*. Then* $\mathsf{MrKtP}[\beta, \alpha, s] \notin \mathsf{Promise\text{-}BPTIME}[n^{\mathsf{poly}(\log n)}]$.

Note that MrKtP is a total function if $\alpha = \beta$, and that the lower bound also holds in this regime. Theorem 1 presents a natural string compression problem that is provably intractable, even with randomness. While it is known that $\mathsf{BPEXP} \not\subseteq \mathsf{BPQP}$, existing proofs of this separation and its extensions only produce artificial computational problems (see e.g. [19, 7, 14, 9] for more background). The proof of Theorem 1 employs different techniques, and the argument is robust enough to establish the hardness of several variants of the problem. We will discuss one of these extensions later in this section.

The main technique used in the proof of Theorem 1 is indirect diagonalization. The argument makes use of results from the theory of pseudorandomnenss, and relies on recent insights from the investigation of pseudodeterministic algorithms [30] and connections between learning algorithms and lower bounds [29]. While pseudorandomness has been explored in the context of Kolmogorov complexity at least since the work of [4], these new perspectives were crucial in the discovery of this unconditional lower bound.

Theorem 1 can be extended to running times that are larger than quasi-polynomial, but it is unclear how to adapt the proof to show a lower bound against randomized algorithms running in time 2^{n^ε} for a small $\varepsilon > 0$. (Similarly, it is not known if $\mathsf{BPTIME}[2^n] \subseteq \mathsf{BPTIME}[2^{n^\varepsilon}]$.) A sub-exponential lower bound is open even with respect to *deterministic* algorithms. Nevertheless, we can prove a weaker lower bound in this direction that relates the deterministic complexities of MKtP and $\mathsf{MrKtP}[\beta, \alpha, s]$. For convenience, we let MrKtP denote the problem with parameters $\beta = 3/4$, $\alpha = 2/3$, and $s(n) = n/2$.

▶ **Theorem 2.** *Either* $\mathsf{MKtP} \notin \mathsf{P}$ *or* $\mathsf{MrKtP} \notin \mathsf{Promise\text{-}EXP}$.

Since MrKtP can be computed in $\mathsf{Promise\text{-}BPE}$, this result shows a weakness of deterministic algorithms solving these problems. The proof of Theorem 2 combines previous results on MKtP that also rely on pseudorandomness with some observations about rKt and MrKtP.

Theorems 1 and 2 indicate that these problems are good candidates for non-uniform circuit lower bounds. In order to discuss our next result, it is convenient to introduce a variant of MrKtP. For a string $w \in \{0,1\}^n$, let $\mathsf{rKt}(w) \stackrel{\text{def}}{=} \mathsf{rKt}_\lambda(w)$ for $\lambda = 2/3$. For functions $s_1, s_2 \colon \mathbb{N} \to \mathbb{N}$, we let $\mathsf{Gap\text{-}MrKtP}[s_1, s_2]$ be the (promise) problem of distinguishing between $\mathsf{rKt}(w) \leq s_1(n)$ versus $\mathsf{rKt}(w) > s_2(n)$. Again, it is not hard to solve this problem in randomized exponential time if there is a certain (small) gap between $s_1(n)$ and $s_2(n)$.

We obtain the following complexity results for $\mathsf{Gap\text{-}MrKtP}$.

[3] We assume for definiteness that M is a "clocked" machine that runs in time at most t on all computation paths. This is not essential, and does not significantly affect the asymptotics of rKt.

▶ **Theorem 3.** *Let $C \geq 1$ be a sufficiently large constant. The following results hold.*

(i) *For every constructive $s\colon \mathbb{N} \to \mathbb{N}$,* $\mathsf{Gap\text{-}MrKtP}[s(n), s(n) + C\log n] \in \mathsf{Promise\text{-}BPTIME}[2^{O(n)}]$.

(ii) *For every $0 < \gamma < 1$,* $\mathsf{Gap\text{-}MrKtP}[n^{\gamma}, n/2] \notin \mathsf{Promise\text{-}BPTIME}[n^{\mathsf{poly}(\log n)}]$.

(iii) *If there is $\varepsilon > 0$ such that for every $0 < \gamma < 1$,* $\mathsf{Gap\text{-}MrKtP}[n^{\gamma}, n^{\gamma} + C\log n] \notin \mathsf{SIZE}[n^{1+\varepsilon}]$, *then* $\mathsf{Promise\text{-}BPEXP} \not\subseteq \mathsf{SIZE}[\mathsf{poly}]$.

Theorem 3 (*ii*) implies a strong *inapproximability* result for computing rKt (see [16] for a recent work where inapproximability of "complexity" plays a role). On the other hand, Theorem 3 (*iii*) proves that weak non-uniform lower bounds for $\mathsf{Gap\text{-}MrKtP}$ can be "magnified" (cf. [31]) to super-polynomial lower bounds for a problem in $\mathsf{Promise\text{-}BPEXP}$. (Such lower bounds are only known for languages in MAEXP [10], which combines *randomness* and *nondeterminism* in the exponential-time regime.)

In contrast to previous work (cf. [28] and references therein), Theorem 3 provides the first hardness magnification theorem for a natural problem that is *provably hard* against a strong model of computation (randomized polynomial-time algorithms).[4] The proof of Theorem 3 (*ii*) is similar to the proof of Theorem 1, while part (*iii*) follows by an adaptation of a version of the result established for $\mathsf{Gap\text{-}MKtP}$ in [28]. Note that Theorem 3 exhibits an interesting contrast between proving uniform and non-uniform lower bounds.

Finally, we consider the relation between Kt and rKt. In other words, can we have shorter descriptions if we allow randomized decoding?[5] As a concrete example, the results in [30] imply that infinitely many prime numbers (represented as binary strings) have sub-polynomial rKt complexity. This is not known to hold with respect to Kt complexity.

We employ standard techniques to establish two results that relate rKt and Kt. The first result links the deterministic complexity of MKtP to the gap between Kt and rKt.

▶ **Theorem 4.** *If $\mathsf{MKtP} \in \mathsf{P}$ then there is a sequence $\{w_n\}_{n\geq 1}$ with $w_n \in \{0,1\}^n$ such that $\mathsf{rKt}(w_n) = O(\log n)$ and $\mathsf{Kt}(w_n) = \Omega(n)$.*

On the other hand, the next theorem (roughly) shows that Kt and rKt are linearly related for every string if and only if randomized exponential time computations can be derandomized. (We refer to [2] for similar results involving other notions of time-bounded Kolmogorov complexity.)

▶ **Theorem 5.** *The following implications hold.*

(i) *If $\mathsf{Promise\text{-}BPE} \subseteq \mathsf{Promise\text{-}E}$, then $\mathsf{Kt}(w) = O(\mathsf{rKt}(w))$ for every string w.*

(ii) *If $\mathsf{Kt}(w) = O(\mathsf{rKt}(w))$ for every string w, then $\mathsf{BPE} \subseteq \mathsf{E}/O(n)$.*

In particular, rKt and Kt are linearly related if E requires exponential size boolean circuits.

This result implies that, under the standard derandomization assumption that $\mathsf{Promise\text{-}BPE}$ is contained in $\mathsf{Promise\text{-}E}$, the problems MrKtP and MKtP essentially coincide. Therefore, our unconditional results for MrKtP and its variants provide strong evidence that MKtP is intractable.

---

[4] Discussions on the feasibility of previous magnification results as an approach toward new non-uniform lower bounds relied either on conjectured separations between complexity classes or on cryptographic assumptions.

[5] Note that it is possible to recover with high probability a string w from its description in time at most $2^{O(\mathsf{rKt}(w))}$. Additionally, one can *exactly* recover w (i.e. with probability 1) by cycling through all choices of the randomness and taking a majority vote.

Related Work. Pseudodeterministic algorithms and hardness magnification are active research areas. We refer to the references in [15, 28] and to papers citing these works for more details. Quantum versions of Kolmogorov complexity have been proposed in [32, 26, 35, 8]. Before this work, unconditional lower bounds were shown for a non-deterministic formulation of Kt, where it was proved that the corresponding decision problem is in P^{NE} but not in $\mathsf{NP} \cap \mathsf{coNP}$. We refer to [5] for more information. Finally, there is a huge literature on time-bounded Kolmogorov complexity and its applications in theory of computation. A recent reference such as [3] contains pointers to many other works in the area.

Concluding Remarks. We view the unconditional lower bounds in Theorem 1 and Theorem 3 (*ii*) as a step toward understanding the hardness of computing the "complexity" of strings. Such problems are important in computer science. In particular, the conjectured security of modern cryptography implies that distinguishing "structured" strings from "random" strings (according to different measures) is hard. In this work, the complexity of a string is explored from the perspective of rKt, which is likely to be essentially equivalent to Kt complexity (as suggested by Theorem 5). Previous unconditional lower bounds on the associated decision problems applied only to strong measures, such as the non-deterministic version of Kolmogorov complexity studied in [5]. Our work is the first to show an unconditional lower bound for a notion of complexity that appears to be equivalent to Levin's seminal Kt complexity. Our techniques are also robust, and lead to a hardness of approximation result. We mention that an average-case lower bound in the sense of [17] can be proved as well.

We leave open the problem of showing an exponential lower bound on the complexity of deciding rKt complexity. Theorem 3 (*iii*) and its extensions to different circuit classes also suggest that investigating non-uniform lower bounds for this problem might be a fruitful direction.

Organization. The next section formalizes some definitions and observations mentioned above, and discusses a couple of basic facts and examples related to randomized time-bounded Kolmogorov complexity. The proofs of Theorems 1 and 2 appear in Section 3. This is followed by a sketch of the proof of Theorem 3 in Section 4. Section 5 discusses Theorems 4 and 5.

2 Preliminaries

For background in (time-bounded) Kolmogorov complexity and related topics, we refer to [25]. We fix a reasonable representation of Turing machines, and let $|M|$ denote the length of the binary encoding of a machine M. Our results are not sensitive to particular encoding choices. We assume that machines have an extra tape with random bits. We let $\boldsymbol{M}_{\leq t}$ denote the random variable that represents the content of the output tape of M when it computes for (at most) t steps over the empty string.

▶ **Definition 6** (Kt_λ Complexity). *For $\lambda \in [0,1]$ and $w \in \{0,1\}^*$, we let*

$$\mathsf{Kt}_\lambda(w) = \min_{M,t}\{|M| + \lceil \log t \rceil \mid \Pr[\boldsymbol{M}_{\leq t} = w] \geq \lambda\}.$$

The randomized time-bounded Kolmogorov complexity of w is given by $\mathsf{rKt}(w) \stackrel{\text{def}}{=} \mathsf{Kt}_{2/3}(w)$.

As a concrete example, the main result of [30] implies that for every $\varepsilon > 0$, there is a sequence $\{p_m\}_{m \geq 1}$ of increasing prime numbers such that $\mathsf{rKt}(p_m) \leq |p_m|^\varepsilon$ for every m, where $|p_m|$ denotes the length of the binary representation of p_m. For the reader familiar

with the ideas from [15] and subsequent work, the randomized time-bounded Kolmogorov complexity of a string can be seen as a measure of its "pseudodeterministic" complexity.

It is easy to see that the definition of $\mathsf{rKt}(w)$ does not change substantially if we use another threshold parameter $1/2 < \lambda < 1$.[6] The (deterministic) time-bounded Kolmogorov complexity of a string w corresponds to $\mathsf{Kt}_\lambda(w)$ for $\lambda = 1$. Note that if $\alpha \leq \beta$ then $\mathsf{Kt}_\alpha(w) \leq \mathsf{Kt}_\beta(w)$.

▶ **Definition 7** ($\mathsf{MrKtP}[\beta, \alpha, s]$)**.** *For $0 < \alpha \leq \beta \leq 1$ and $s\colon \mathbb{N} \to \mathbb{N}$, we let $\mathsf{MrKtP}[\beta, \alpha, s]$ be the promise problem $(\mathcal{YES}_n, \mathcal{NO}_n)_{n\in\mathbb{N}}$, where*

$$\begin{aligned} \mathcal{YES}_n &= \{w \in \{0,1\}^n \mid \mathsf{Kt}_\beta(w) \leq s(n)\}, \\ \mathcal{NO}_n &= \{w \in \{0,1\}^n \mid \mathsf{Kt}_\alpha(w) > s(n)\}. \end{aligned}$$

For concreteness, we let MrKtP denote $\mathsf{MrKtP}[\beta, \alpha, s]$ for $\beta = 3/4$, $\alpha = 2/3$, and $s = n/2$.

We will tacitly assume that s is constructive in all results.

▶ **Lemma 8.** *For rationals $0 < \alpha < \beta \leq 1$ and a function $s\colon \mathbb{N} \to \mathbb{N}$, $\mathsf{MrKtP}[\beta, \alpha, s] \in \mathsf{Promise\text{-}BPE}$.*

Proof Sketch. Let $\alpha < \eta < \beta$, for a fixed rational η. For all appropriate machines M and running times t, estimate with confidence at least $1 - 2^{-\omega(n)}$ the probability that M generates w when it computes for t steps. Consider M and its time bound t to be "good" if this probability estimate is at least η. Accept w if and only if a good pair has combined complexity at most s.

The correctness of the algorithm follows by a concentration bound and a standard union bound. The upper bound on its running time uses that $\mathsf{Kt}_\lambda(w)$ is at most $O(|w|)$ for every string w and $\lambda \in [0,1]$. ◀

Note that if $\beta = \alpha$ then $\mathsf{MrKtP}[\beta, \alpha, s]$ is a total problem. However, it is unclear if the problem is in BPE for this choice of parameters.

It is also convenient to consider a close variant of MrKtP. Recall that $\mathsf{rKt}(w) = \mathsf{Kt}_{2/3}(w)$.

▶ **Definition 9** ($\mathsf{Gap\text{-}MrKtP}[s_1, s_2]$)**.** *Let $s_1, s_2\colon \mathbb{N} \to \mathbb{N}$, where $s_1(n) \leq s_2(n)$ for every $n \in \mathbb{N}$. We let $\mathsf{Gap\text{-}MrKtP}[s_1, s_2]$ be the promise problem $(\mathcal{YES}_n, \mathcal{NO}_n)_{n\in\mathbb{N}}$, where*

$$\begin{aligned} \mathcal{YES}_n &= \{w \in \{0,1\}^n \mid \mathsf{rKt}(w) \leq s_1(n)\}, \\ \mathcal{NO}_n &= \{w \in \{0,1\}^n \mid \mathsf{rKt}(w) > s_2(n)\}. \end{aligned}$$

▶ **Lemma 10.** *Suppose that $s_1(n) + c \log n \leq s_2(n)$, where $c \geq 1$ is a large enough constant. Then $\mathsf{Gap\text{-}MrKtP}[s_1, s_2] \in \mathsf{Promise\text{-}BPE}$.*

Proof. Given Lemma 8, it is enough to reduce $\mathsf{Gap\text{-}MrKtP}[s_1, s_2]$ to $\mathsf{MrKtP}[2/3, 3/5, s_1]$. Clearly, the set of positive instances of both problems coincide. On the other hand, it is easy to see that $\mathsf{Kt}_{2/3}(w) \leq \mathsf{Kt}_{3/5}(w) + c \log n$ if c is a sufficiently large universal constant, by amplification of the underlying randomized algorithm. As a consequence,

$$\{w \in \{0,1\}^n \mid \mathsf{rKt}(w) > s_2(n)\} \subseteq \{w \in \{0,1\}^n \mid \mathsf{Kt}_{3/5}(w) > s_1(n)\},$$

since if $\mathsf{rKt}(w) > s_2(n)$ then $\mathsf{Kt}_{3/5}(w) > s_2(n) - c \log n$, and by assumption $s_2(n) - c \log n \geq s_1(n)$. In other words, the set of negative instances of $\mathsf{Gap\text{-}MrKtP}[s_1, s_2]$ is contained in the set of negative instances of $\mathsf{MrKtP}[2/3, 3/5, s_1]$. ◀

[6] It is not hard either to prove this claim for a constant $0 < \lambda \leq 1/2$, and we leave it as an exercise. (Hint: Use a short advice string to distinguish w from any other string that is output with probability $\geq \lambda/2$.)

▶ Remark. For simplicity of the exposition, we might abuse notation in some statements and compare a promise problem with a standard complexity class. However, in all proofs the distinction between the two cases is carefully considered.

3 The computational hardness of MrKtP

3.1 MrKtP is not in BPP

The main result established in this section is the following lower bound.

▶ **Theorem 11.** *Let $1/2 < \alpha \leq \beta < 1$ and $n^\gamma \leq s(n) \leq n/2$ for every large enough $n \in \mathbb{N}$, where $\gamma > 0$ is fixed but arbitrary. Then $\mathsf{MrKtP}[\beta, \alpha, s] \notin \mathsf{BPTIME}[n^{\mathsf{poly}(\log n)}]$. In other words, no randomized algorithm running in quasi-polynomial time accepts with probability $\geq 2/3$ the positive instances of $\mathsf{MrKtP}[\beta, \alpha, s]$ and rejects with probability $\geq 2/3$ the negative instances of $\mathsf{MrKtP}[\beta, \alpha, s]$.*

The proof given here requires the following results, which assume parameters α, β, and s as in Theorem 11. (We refer to [4] for applications of similar techniques.)

▶ **Lemma 12.** $\mathsf{BPE} \leq_{\mathsf{tt}}^{\mathsf{P/poly}} \mathsf{MrKtP}[\beta, \alpha, s]$. *In particular, given any sequence $\{g_n\}_{n\geq 1}$ of total boolean functions $g_n \colon \{0,1\}^n \to \{0,1\}$ that compute $\mathsf{MrKtP}[\beta, \alpha, s]$, every language in BPE can be computed by (deterministic) polynomial size oracle circuits with access to $\{g_n\}_{n\geq 1}$.*

▶ **Lemma 13.** $\mathsf{PSPACE} \subseteq \mathsf{BPP}^{\mathsf{MrKtP}[\beta,\alpha,s]}$. *More precisely, given any fixed oracle $\mathcal{O} \subseteq \{0,1\}^*$ that agrees with $\mathsf{MrKtP}[\beta, \alpha, s]$ over the relevant input strings, $\mathsf{PSPACE} \subseteq \mathsf{BPP}^{\mathcal{O}}$. Furthermore, if $\mathcal{O}$ is randomized and satisfies the promise of bounded acceptance probabilities over the inputs of $\mathsf{MrKtP}[\beta, \alpha, s]$, then the corresponding algorithm in $\mathsf{BPP}^{\mathcal{O}}$ satisfies this promise over all input strings.*

We postpone the proof of these lemmas. The next lemma is well known, and can be proved by a diagonalization argument (see e.g. [29, Corollary 2]).

▶ **Lemma 14.** *Let $s_1, s_2 \colon \mathbb{N} \to \mathbb{N}$ be space-constructible functions such that $s_2(n)^2 = o(s_1(n))$, $s_2(n) = \Omega(n)$, and $s_1(n) = 2^{o(n)}$. Then there is a language in $\mathsf{DSPACE}[s_1(n)]$ that cannot be computable by circuits of size $s_2(n)$.*

We are ready to prove Theorem 11, assuming these results.

Proof of Theorem 11. Suppose toward a contradiction that $\mathsf{MrKtP}[\beta, \alpha, s]$ can be computed in $\mathsf{BPTIME}[n^{(\log n)^a}]$, for some $a > 0$. Then, by standard non-uniform derandomization, $\mathsf{MrKtP}[\beta, \alpha, s]$ can be computed by circuits of size $O(n^{(\log n)^b})$, for some $b > 0$. It follows from Lemma 12 that every language $L \in \mathsf{BPE}$ can be computed by circuits of size $O(n^{(\log n)^{c_L}})$, for some $c_L > 0$.

Let L^* be a language given by Lemma 14 for appropriate parameters $s_1(n) = 2^{n^{o(1)}}$ and $s_2(n) = n^{(\log n)^{\omega(1)}}$. In other words, $L^* \in \mathsf{DSPACE}[s_1] \setminus \mathsf{SIZE}[s_2]$. Lemma 13 and our initial assumption imply that $\mathsf{PSPACE} \subseteq \mathsf{BPTIME}[n^{\mathsf{poly}(\log n)}]$. By a standard padding argument, we get that $L^* \in \mathsf{BPE}$. But then the upper and lower bounds on the circuit complexity of L^* are in contradiction. This completes the proof of Theorem 11. ◀

We proceed with the proofs of Lemmas 12 and 13. Given a function $f \colon \{0,1\}^* \to \{0,1\}$, we consider an associated "pseudorandom" generator G_f. (Formally, the argument employs a uniform sequence of generators, one for each $n \geq 1$.) More precisely, the generator $G_f^{\mathsf{BFNW}} \colon \{0,1\}^{n^\varepsilon} \to \{0,1\}^n$ can be computed in deterministic time $\exp(O(n^\varepsilon))$ given oracle access to f on inputs of length at most n^ε, and satisfies the following crucial property.

▶ **Theorem 15** (see [6, 21]). *Let $f\colon \{0,1\}^* \to \{0,1\}$ be a function, $\varepsilon > 0$ be arbitrary, and G_f^{BFNW} be the associated sequence of functions mentioned above. Moreover, let $T \subseteq \{0,1\}^*$ be an arbitrary test. If*

$$\left| \Pr_{\boldsymbol{r} \in U_n} [\boldsymbol{r} \in T] - \Pr_{\boldsymbol{x} \in U_{n^\varepsilon}} [G_f^{\mathsf{BFNW}}(\boldsymbol{x}) \in T] \right| \geq 1/n$$

for every large enough n, then there is a sequence $\{C_n\}_{n \geq 1}$ of polynomial size oracle circuits with access to T that compute f on each input length n and query T nonadaptively.

Proof of Lemma 12. Let $L \in \mathsf{BPE}$, and $\{f_n\}_{n \geq 1}$ be the corresponding sequence of boolean functions that compute L. Recall the constants $1/2 < \alpha \leq \beta < 1$ and $\gamma > 0$ from the statements of Theorem 11 and Lemma 12. Take $\varepsilon \stackrel{\text{def}}{=} \gamma/2$, and consider the generator G_f^{BFNW} obtained from f and ε. Moreover, let $\{g_n\}_{n \geq 1}$ be a sequence of boolean functions $g_n\colon \{0,1\}^n \to \{0,1\}$ that agree with $\mathsf{MrKtP}[\beta, \alpha, s]$ over input strings in $\mathcal{YES}_n \cup \mathcal{NO}_n$. Finally, set $T \stackrel{\text{def}}{=} \bigcup_{n \geq 1} g_n^{-1}(0)$.

We claim that T distinguishes the output of G_f^{BFNW} from a random n-bit string. First, for each seed $w \in \{0,1\}^{n^\varepsilon}$, $G_f^{\mathsf{BFNW}}(w)$ can be computed in time at most $\exp(O(n^\varepsilon))$ given w and oracle access to $f_1, \ldots, f_{n^\varepsilon}$. Since each function f_i for $i \leq n^\varepsilon$ can be computed in randomized time $\exp(O(n^\varepsilon))$ and with error probability at most $\exp(-n)$ by a uniform algorithm, it follows that $\mathsf{Kt}_\beta(G_f^{\mathsf{BFNW}}(w)) \leq \mathsf{Kt}_{1-o(1)}(G_f^{\mathsf{BFNW}}(w)) \leq O(n^\varepsilon) < n^\gamma \leq s(n)$, for n sufficiently large. Therefore, $G_f^{\mathsf{BFNW}}(w) \notin T$ for every $w \in \{0,1\}^{n^\varepsilon}$. On the other hand, a typical random n-bit string $\boldsymbol{r} \in U_n$ has (standard) Kolmogorov complexity $K(\boldsymbol{r}) \geq (1 - o(1))n$. It is easy to see that if $\lambda > 1/2$, then $K(x) \leq \mathsf{Kt}_\lambda(x)$ for a string x. As a consequence, with high probability $\mathsf{Kt}_\alpha(\boldsymbol{r}) > n/2 \geq s(n)$, in which case we have $\boldsymbol{r} \in T$.

Since T distinguishes the generator from random, it follows from Theorem 15 that L can be computed by polynomial size oracle circuits that make non-adaptive queries to T, i.e., to the functions $\{g_n\}_{n \geq 1}$. ◀

In order to prove Lemma 13, we need a *uniform* version of Theorem 15. A result of this form was established in [18], and we discuss it in more detail now. For $\varepsilon > 0$ and a function $f\colon \{0,1\}^* \to \{0,1\}$, the generator $G_f^{\mathsf{IW}}\colon \{0,1\}^{n^\varepsilon} \to \{0,1\}^n$ is also computable in deterministic time $\exp(O(n^\varepsilon))$ with oracle access to f on inputs of size at most n^ε. In addition, it satisfies the following property.

▶ **Theorem 16** (see [18]). *Let $f\colon \{0,1\}^* \to \{0,1\}$ be a function that is both random self-reducible and downward self-reducible, $\varepsilon > 0$ be arbitrary, and G_f^{IW} be the associated sequence of functions mentioned above. Moreover, let $T \subseteq \{0,1\}^*$ be an arbitrary test. If*

$$\left| \Pr_{\boldsymbol{r} \in U_n} [\boldsymbol{r} \in T] - \Pr_{\boldsymbol{x} \in U_{n^\varepsilon}} [G_f^{\mathsf{IW}}(\boldsymbol{x}) \in T] \right| \geq 1/n$$

for every large enough n, then there is a randomized polynomial-time Turing machine with oracle access to T that on every input x outputs $f(x)$ with high probability.

▶ **Theorem 17** (see [33]). *There is a language $L_{\mathsf{TV}} \in \mathsf{DSPACE}[O(n)]$ that is PSPACE-hard, random self-reducible, and downward self-reducible.*

We are ready to prove Lemma 13, which completes the proof of Theorem 11.

Proof of Lemma 13 (Sketch). Let L_{TV} be the language from Theorem 17. Since this language is PSPACE-hard under polynomial-time reductions, it suffices to show that $L_{\mathsf{TV}} \in \mathsf{BPP}^{\mathsf{MrKtP}[\beta,\alpha,s]}$.

We argue as in the proof of Lemma 12. More precisely, we let $\varepsilon \stackrel{\text{def}}{=} \gamma/2$, and we instantiate the generator G_f^{IW} using the function f that computes the characteristic function of L_{TV}. If $\mathcal{O}$ is a deterministic test that agrees with $\mathsf{MrKtP}[\beta, \alpha, s]$, then a similar argument shows that every output string of the generator has randomized Kt complexity at most $s(n)$, while a random string has almost maximum complexity. The only modification here is that $f_1, \ldots, f_{n^\varepsilon}$ can all be computed in *deterministic* time $\exp(O(n^\varepsilon))$, which follows from the fact that L_{TV} is computable in linear space. Theorem 16 immediately implies that $L_{\mathsf{TV}} \in \mathsf{BPP}^{\mathcal{O}}$, as desired.

Suppose that $\mathcal{O}$ is a *randomized* procedure that accepts the positive instances of $\mathsf{MrKtP}[\beta, \alpha, s]$ with high probability, and rejects the negative instances of $\mathsf{MrKtP}[\beta, \alpha, s]$ with high probability. We make no assumptions on the acceptance probabilities of $\mathcal{O}$ over the remaining input strings. In order to establish the furthermore part in Lemma 13, it is necessary to inspect the proof of Theorem 16. The crucial observation is that the oracle $\mathcal{O}$ is only used as a distinguisher during the computation of L_{TV}, and that any procedure that distinguishes with noticeable advantage the output of the generator from a random string can be used in place of $\mathcal{O}$. (The argument sketched in the paragraph above can be used to show that the output of $\mathcal{O}$ on strings that violate the promise condition affects in a negligible way its advantage as a distinguisher.)

We also note that it is possible to reduce the analysis of the case of a randomized algorithm A as oracle to the deterministic case. By running polynomially many independent copies of A and taking a majority vote, one gets a randomized algorithm A' that is correct with probability at least $1 - 2^{-m^2}$ on every fixed string of length at most m satisfying the promise condition (think of m as n^ℓ for a large enough constant ℓ, where n is the input length of L_{TV}). By a union bound, randomly fixing the string in the random tape of A' provides w.h.p a deterministic oracle $\mathcal{O}$ that is correct on all strings of length at most m satisfying the promise condition. The analysis now reduces to the deterministic case.

This completes the proof of Lemma 13. ◀

Sketch of an alternate presentation via learning algorithms. Suppose that $\mathsf{MrKtP} \in \mathsf{BPP}$, i.e., there is a polynomial time randomized algorithm that is correct with high probability over inputs satisfying the promise condition. Then, by adapting ideas from [11], it is possible to prove that for every reasonable function $t\colon \mathbb{N} \to \mathbb{N}$, $\mathsf{SIZE}[t]$ can be learned in $\mathsf{BPTIME}[\mathsf{poly}(t)]$ in the model of learning with membership queries under the uniform distribution. The connection between learning algorithms and lower bounds (see [29]) now implies that, for any choice of $s(n) \leq n^{\mathsf{poly}(\log n)}$, $\mathsf{BPEXP} \not\subseteq \mathsf{SIZE}[s(n)]$. But this is in contradiction to Lemma 12 and the assumption that $\mathsf{MrKtP} \in \mathsf{BPP}$, which imply $\mathsf{BPEXP} \subseteq \mathsf{SIZE}[\mathsf{poly}(n)]$.

We remark that common to both approaches are elements from the theory of pseudorandomness, such as the use of pseudorandom generators based on [27], and ideas that go back to the work of [18] on connections between algorithms and lower bounds via random self-reducibility and downward self-reducibility. The use of [6] in the proof of Lemma 12 appears to be crucial in the arguments presented above.

3.2 Weakness of deterministic algorithms for MKtP and MrKtP

It is natural to conjecture that $\mathsf{BPEXP} \not\subseteq \mathsf{BPTIME}[2^{o(n)}]$ (a strong hierarchy theorem for randomized time) and $\mathsf{MrKtP} \notin \mathsf{BPTIME}[2^{o(n)}]$ (a nearly-optimal lower bound for MrKtP). However, it is unclear even how to show that $\mathsf{MrKtP} \notin \mathsf{DTIME}[2^{n^{o(1)}}]$. It is also open whether $\mathsf{MKtP} \in \mathsf{P}$. In this section, we place limits on the efficiency of deterministic algorithms solving these problems. We start with the following observation.

▶ **Proposition 18.** *Either* $\mathsf{EXP} \not\subseteq \mathsf{BPTIME}[2^{o(n)}]$ *or* $\mathsf{MrKtP} \notin \mathsf{EXP}$.

Proof. Suppose $\mathsf{MrKtP} = \mathsf{MrKtP}[\beta, \alpha, s]$ is in $\mathsf{DTIME}[2^{n^d}]$ for some constant d, where $\beta = 3/4$, $\alpha = 2/3$, and $s = n/2$. Let M_{MrKtP} be a Turing machine that witnesses this inclusion. Consider the following language:

$$L \stackrel{\text{def}}{=} \{\langle M, 1^n, w\rangle \mid M \text{ is a TM that accepts in time } \leq 2^{n^d} \text{ some } n\text{-bit string } y \text{ whose prefix is } w\}.$$

Note that $L \in \mathsf{EXP}$, i.e., L can be computed in deterministic time $2^{m^{O(1)}}$ on inputs of length m. Assume that $\mathsf{EXP} \subseteq \mathsf{BPTIME}[2^{o(n)}]$, and let M_L be a randomized Turing machine for L that witnesses this inclusion. It is easy to see that M_L can be used to find w.h.p. and in time $2^{o(n)}$ the lexicographic first string $z \in \{0,1\}^n$ accepted by $\overline{M_{\mathsf{MrKtP}}}$, the complement of machine M_{MrKtP} (observe that such string must exist). It follows that the triple $(M_L, M_{\mathsf{MrKtP}}, 1^n)$ can be used to give a shorter description of z. More precisely, $\mathsf{rKt}_{1-o(1)}(z) = o(n)$. On the other hand, since $\overline{M_{\mathsf{MrKtP}}}(z) = 1$ and M_{MrKtP} computes MrKtP, we must have $\mathsf{rKt}_\beta(z) > s$. These inequalities imply that $n/2 = s < \mathsf{rKt}_\beta(z) \leq \mathsf{rKt}_{1-o(1)}(z) \leq o(n)$, a contradiction. This completes the proof of Proposition 18. ◀

Additionally, we will use the following reductions.

▶ **Lemma 19** (see [4])**.** $\mathsf{EXP} \subseteq \mathsf{NP}^{\mathsf{MKtP}}$.

▶ **Lemma 20** (see [4])**.** *If* $\mathsf{MKtP} \in \mathsf{P}$ *then* $\mathsf{PSPACE} \subseteq \mathsf{ZPP}$.

As alluded to above, the next result shows hardness of deciding deterministic/randomized time-bounded Kolmogorov complexity using deterministic algorithms. (It should be contrasted with the inclusion $\mathsf{MrKtP} \in \mathsf{Promise\text{-}BPE}$ from Lemma 8.)

▶ **Theorem 21.** *Either* $\mathsf{MKtP} \notin \mathsf{P}$ *or* $\mathsf{MrKtP} \notin \mathsf{EXP}$.

Proof. Suppose $\mathsf{MKtP} \in \mathsf{P}$. Then $\mathsf{PSPACE} \subseteq \mathsf{ZPP}$ follows by Lemma 20. Moreover, Lemma 19 gives $\mathsf{EXP} \subseteq \mathsf{NP}$. Combining these two class inclusions, we get that $\mathsf{EXP} \subseteq \mathsf{BPP}$. But this implies that $\mathsf{MrKtP} \notin \mathsf{EXP}$ via Proposition 18, which is the desired result. ◀

4 Non-uniform versus randomized lower bounds for MrKtP

It is not hard to see that the proof of Theorem 11 carries over with $\mathsf{Gap\text{-}MrKtP}[s_1, s_2]$ in place of $\mathsf{MrKtP}[\beta, \alpha, s]$. We state the result here for completeness.

▶ **Theorem 22.** *Let* $\gamma > 0$ *be an arbitrarily small constant, and consider functions* $s_1, s_2 \colon \mathbb{N} \to \mathbb{N}$. *Suppose that* $n^\gamma \leq s_1(n) \leq s_2(n) \leq n/2$. *Then* $\mathsf{Gap\text{-}MrKtP}[s_1, s_2] \notin \mathsf{BPTIME}[n^{\mathsf{poly}(\log n)}]$.

Proof Sketch. The algorithm for $\mathsf{MrKtP}[\beta, \alpha, s]$ is only used as a distinguisher in the proof of Theorem 11. It is possible to check that an algorithm for $\mathsf{Gap\text{-}MrKtP}[s_1, s_2]$ works equally well as a distinguisher in the proofs of Lemmas 12 and 13. ◀

By a straightforward extension of results from [31, 28], one can show that weak *non-uniform* lower bounds for $\mathsf{Gap\text{-}MrKtP}[s_1, s_2]$ can be "magnified" to super-polynomial circuit lower bounds for some explicit problem. We state a version of the result for general boolean circuits, but the proof can be adapted to other boolean devices (similarly to [28]).

▶ **Theorem 23.** *There is a universal constant $d \geq 1$ for which the following holds. If there exists $\varepsilon > 0$ such that for every small enough $\beta > 0$ we have* $\mathsf{Gap\text{-}MrKtP}[n^\beta, n^\beta + d \log n] \notin \mathsf{SIZE}[n^{1+\varepsilon}]$*, then* $\mathsf{Promise\text{-}BPEXP} \not\subseteq \mathsf{SIZE}[\mathrm{poly}]$.

Proof Sketch. We verify that the relevant steps in the proof of [28, Theorem 1 Part 1, Section 3] carry over to $\mathsf{Gap\text{-}MrKtP}[n^\beta, n^\beta + d \log n]$, under minor modifications. (Note that N denotes the input length in [28], while here input length is denoted by n.) First, we observe that Claims 11 and 12 also work for rKt, since the error-correcting code routines are deterministic. Using a similar notation (i.e., $z = \mathsf{ECC}(w) \in \{0,1\}^m$, where $m = m(n) = O(n)$ and $n = |w|$), it follows that if $\mathsf{rKt}(w) \leq n^\beta$ then $\mathsf{rKt}(z) \leq n^\beta + c_0 \log n$, and that if $\mathsf{rKt}(w) > n^\beta + d \log n$ then $\mathsf{rKt}(z') > n^\beta + c_1 \log n$ for any $z' \in \{0,1\}^m$ that disagrees with z on at most a δ-fraction of coordinates, where $1 \leq c_0 < c_1 < d$ are constants, and we assume that d is large enough so that c_0 and c_1 are sufficiently far apart.

The crucial part of the argument is to replace the language $L \in \mathsf{EXP}$ from their Claim 13 by an appropriate problem $\Pi \in \mathsf{Promise\text{-}BPEXP}$. The input to Π is a string y encoding a tuple of the form $(m, 1^t, (i_1, b_1), \ldots, (i_r, b_r))$, where m is a positive integer represented in binary, t is a positive integer, $i_1, \ldots, i_r \in \{0,1\}^{\log m}$, $b_1, \ldots, b_r \in \{0,1\}$, and $r \in \mathbb{N}$. For $t = n^\beta$ and $r = n^{2\beta}$, we let

$$\Pi_n^{\mathsf{yes}} \stackrel{\text{def}}{=} \{y \mid \exists z \in \{0,1\}^m \text{ such that } \mathsf{rKt}(z) \leq t + c_0 \log n \text{ and } z_{i_1} = b_1, \ldots, z_{i_r} = b_r\}, \text{ and}$$
$$\Pi_n^{\mathsf{no}} \stackrel{\text{def}}{=} \{y \mid \nexists z \in \{0,1\}^m \text{ such that } \mathsf{rKt}(z) \leq t + c_1 \log n \text{ and } z_{i_1} = b_1, \ldots, z_{i_r} = b_r\}.$$

Note that these sets are disjoint, and that $\Pi \in \mathsf{Promise\text{-}BPEXP}$ by an argument analogous to the proof of Lemma 10, using that the gap between constants c_0 and c_1 is sufficiently large.

It remains to check that their Claim 14 still holds in our context. For part (a), note that if $\mathsf{rKt}(w) \leq n^\beta$ then by the discussion above $\mathsf{rKt}(z) \leq n^\beta + c_0 \log n$. Consequently, the corresponding input y generated by the randomized reduction is in Π_n^{yes} with probability 1. Similarly, for part (b) we rely on the claim that if $\mathsf{rKt}(w) > n^\beta + d \log n$ then $\mathsf{rKt}(z') > n^\beta + c_1 \log n$ for any $z' \in \{0,1\}^m$ that disagrees with z on at most a δ-fraction of coordinates. The same union bound over exponentially small probabilities shows that $y \in \Pi_n^{\mathsf{no}}$ with probability at least $\geq 1/2$.

The rest of the construction remains unaffected. ◀

5 On the relation between rKt and Kt

First, we observe that the worst-case gap between rKt and Kt over strings of length n is closely related to the derandomization of exponential time computations.[7]

▶ **Theorem 24.** *The following implications hold.*

(i) *If* $\mathsf{Promise\text{-}BPE} \subseteq \mathsf{Promise\text{-}E}$*, then* $\mathsf{Kt}(w) = O(\mathsf{rKt}(w))$ *for every string w.*

(ii) *If* $\mathsf{Kt}(w) = O(\mathsf{rKt}(w))$ *for every string w, then* $\mathsf{BPE} \subseteq \mathsf{E}/O(n)$.

In particular, rKt *and* Kt *are linearly related if* E *requires exponential size boolean circuits.*

[7] Recall that if $\mathsf{BPP} \subseteq \mathsf{P}$ then $\mathsf{BPEXP} \subseteq \mathsf{EXP}$ by translation. Consequently, derandomizing exponential time computations is not harder than derandomizing polynomial time computations. Indeed, it is not hard to prove that the derandomization of exponential time is equivalent to the derandomization of sparse languages in BPP.

Proof. We start with a proof of (i). Let $w \in \{0,1\}^n$, and suppose M and t are such that $\Pr[\boldsymbol{M}_{\leq t} = w] \geq 2/3$ and $|M| + \log t = \mathsf{rKt}(w)$. We would like to use M and the assumption that $\mathsf{Promise\text{-}BPE} \subseteq \mathsf{Promise\text{-}E}$ to upper bound the Kt complexity of w. A potential difficulty here is that the latter inclusion offers an asymptotic upper bound, while M and w are fixed objects of finite size. In order to handle this issue, we adopt a more general perspective.

Let U be a randomized universal Turing machine that simulates computations with a polynomial overhead. In other words, given the description of a randomized machine M', a time bound t' specified as a binary string, and an input string x', $U(M', t', x')$ uses its internal randomness to simulate $M'(x')$ for at most t' steps, and outputs whatever M' outputs on x'. We assume that the computation of $U(M', t', x')$ takes time at most $c(|M'| + t' + |x'|)^c$, where $c = c(U) \geq 1$ is a universal constant.

We consider a promise problem Π, defined as follows. The $\mathcal{YES}$ instances consist of tuples $(M', t', 1^{c \cdot \log t'}, i)$, where M' is the description of a randomized Turing machine, t' and i are positive integers represented in binary, and $\Pr[\text{The } i\text{-th bit of } \boldsymbol{M}'_{\leq t'} \text{ is } 1] \geq 2/3$. On the other hand, the set $\mathcal{NO}$ of negative instances of Π is defined by the condition $\Pr[\text{The } i\text{-th bit of } \boldsymbol{M}'_{\leq t'} \text{ is } 0] \geq 2/3$. Clearly, $\mathcal{YES} \cap \mathcal{NO} = \emptyset$. We claim that $\Pi \in \mathsf{Promise\text{-}BPE}$. In order to see this, given a valid input $(M', t', 1^{c \cdot \log t'}, i)$ of Π, run the randomized universal machine U on (M', t', ϵ) for t' steps, where ϵ is the empty string, and output 1 if and only if the i-th bit in the output of M' is 1. This defines a randomized machine that runs in time $O((|M'| + t')^c + i)$, which is at most exponential in its total input length. Since the randomness of U is used to simulate the randomness of M', every string in $\mathcal{YES}$ is accepted with probability at least $2/3$, while every string in $\mathcal{NO}$ is rejected with probability at least $2/3$. This shows that $\Pi \in \mathsf{Promise\text{-}BPE}$.

Under the hypothesis of (i), we obtain that Π is computed by a deterministic machine A_Π that runs in time at most 2^{Cm} on inputs of length m, where C is fixed. Now given the pair (M, t) witnessing the rKt complexity of w (a string of length n), we can use A_Π, M, t, and n to upper bound its Kt complexity. Indeed, w can be generated by the deterministic machine that runs A_Π on $(M, t, 1^{c \cdot \log t}, i)$ for each $i \in [n]$. Note that each input to A_Π satisfies the promise condition of Π, and that A_Π runs in time at most $2^{C(|M| + \log t + c \log t + \log i)}$. Therefore, $\mathsf{rKt}(w) \leq O(|M| + |A_\Pi| + \log t + \log n) + \log(O(n \cdot 2^{C(|M| + \log t + c \log t + \log n)})) = O(|M| + \log t + \log n) = O(\mathsf{rKt}(w) + \log |w|) = O(\mathsf{rKt}(w))$, where the last inequality uses that $\mathsf{rKt}(w) \geq \log |w|$ since any machine that prints w runs in time at least $|w|$.

To prove (ii), let $L \in \mathsf{BPE}$, and let M be a machine for L that runs in randomized exponential time. Define a sequence $\{w_n\}_{n \geq 1}$ of strings $w_n \in \{0,1\}^{2^{n+1}}$, where w_n encodes the output of L on all strings of length at most n. Given n as an input, by amplifying the success probability of M, we can print w_n with high probability in time $2^{O(n)}$. Consequently, $\mathsf{rKt}(w_n) = O(n)$, which is logarithmic in $|w_n|$. Under the assumption that $\mathsf{Kt}(w) = O(\mathsf{rKt}(w))$ for every string w, it follows that for every n, $\mathsf{Kt}(w_n) = O(n)$. In particular, some deterministic machine M_n with $|M_n| = O(n)$ decides L on inputs of length at most n in time $2^{O(n)}$. Now using the sequence $\{M_n\}_{n \geq 1}$ as advice and computing in the obvious way, it follows that $L \in \mathsf{E}/O(n)$. This completes the proof of (ii).

Finally, under the assumption that there is a language in E that requires circuits of size $2^{\Omega(n)}$ on every large input length, there are quick pseudorandom generators of logarithmic seed length (cf. [34]). Such generators can be used to derandomize not only $\mathsf{BPTIME}[t]$ but also $\mathsf{Promise\text{-}BPTIME}[t]$, hence it follows from (i) that rKt and Kt are linearly related. ◀

We now relate the deterministic complexity of MKtP to the gap between rKt and Kt.

▶ **Theorem 25.** *If* $\mathsf{MKtP} \in \mathsf{P}$ *then there is a sequence* $\{w_n\}_{n\geq 1}$ *with* $w_n \in \{0,1\}^n$ *such that* $\mathsf{rKt}(w_n) = O(\log n)$ *and* $\mathsf{Kt}(w_n) = \Omega(n)$.

Proof. The proof is inspired by a related idea of Schuichi Hirahara (private communication). Let $\{D_n\}_{n\geq 1}$ be a P-uniform sequence of polynomial size circuits computing MKtP_t, for a Kt complexity threshold parameter $t(n) = n/2$. The existence of such circuits follows from the hypothesis of the theorem. Now Lemma 20 implies that there is a randomized algorithm running in time polynomial in n that solves the circuit satisfiability problem for circuits of size $\mathsf{poly}(n)$ over n input variables. We can use this algorithm and self-reduction to find with high probability the lexicographic first string w_n accepted by the complement of D_n. Then, by construction, we get that $\mathsf{rKt}(w_n) = O(\log n)$ and $\mathsf{Kt}(w_n) = \Omega(n)$. ◀

References

1. Eric Allender. Applications of time-bounded Kolmogorov complexity in complexity theory. In *Kolmogorov complexity and computational complexity*, pages 4–22. Springer, 1992.
2. Eric Allender. When Worlds Collide: Derandomization, Lower Bounds, and Kolmogorov Complexity. In *Conference on Foundations of Software Technology and Theoretical Computer Science* (FSTTCS), pages 1–15, 2001. doi:10.1007/3-540-45294-X_1.
3. Eric Allender. The complexity of complexity. In *Computability and Complexity*, pages 79–94. Springer, 2017.
4. Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from Random Strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006. doi:10.1137/050628994.
5. Eric Allender, Michal Koucký, Detlef Ronneburger, and Sambuddha Roy. The pervasive reach of resource-bounded Kolmogorov complexity in computational complexity theory. *J. Comput. Syst. Sci.*, 77(1):14–40, 2011. doi:10.1016/j.jcss.2010.06.004.
6. László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993. doi:10.1007/BF01275486.
7. Boaz Barak. A Probabilistic-Time Hierarchy Theorem for "Slightly Non-uniform" Algorithms. In *International Workshop on Randomization and Approximation Techniques* (RANDOM), pages 194–208, 2002. doi:10.1007/3-540-45726-7_16.
8. André Berthiaume, Wim van Dam, and Sophie Laplante. Quantum Kolmogorov Complexity. *J. Comput. Syst. Sci.*, 63(2):201–221, 2001. doi:10.1006/jcss.2001.1765.
9. Harry Buhrman, Lance Fortnow, and Rahul Santhanam. Unconditional Lower Bounds against Advice. In *International Colloquium on Automata, Languages and Programming* (ICALP), pages 195–209, 2009. doi:10.1007/978-3-642-02927-1_18.
10. Harry Buhrman, Lance Fortnow, and Thomas Thierauf. Nonrelativizing Separations. In *Conference on Computational Complexity* (CCC), pages 8–12, 1998. doi:10.1109/CCC.1998.694585.
11. Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning Algorithms from Natural Proofs. In *Conference on Computational Complexity* (CCC), pages 10:1–10:24, 2016. doi:10.4230/LIPIcs.CCC.2016.10.
12. Gregory J. Chaitin. Information-Theoretic Limitations of Formal Systems. *J. ACM*, 21(3):403–424, 1974. doi:10.1145/321832.321839.
13. Lance Fortnow. Kolmogorov complexity and computational complexity. *Complexity of Computations and Proofs. Quaderni di Matematica*, 13, 2004.
14. Lance Fortnow, Rahul Santhanam, and Luca Trevisan. Hierarchies for semantic classes. In *Symposium on Theory of Computing* (STOC), pages 348–355, 2005. doi:10.1145/1060590.1060642.

15 Eran Gat and Shafi Goldwasser. Probabilistic Search Algorithms with Unique Answers and Their Cryptographic Applications. *Electronic Colloquium on Computational Complexity* (ECCC), 18:136, 2011.

16 Shuichi Hirahara. Non-Black-Box Worst-Case to Average-Case Reductions within NP. In *Symposium on Foundations of Computer Science* (FOCS), pages 247–258, 2018. doi:10.1109/FOCS.2018.00032.

17 Shuichi Hirahara and Rahul Santhanam. On the Average-Case Complexity of MCSP and Its Variants. In *Computational Complexity Conference* (CCC), pages 7:1–7:20, 2017. doi:10.4230/LIPIcs.CCC.2017.7.

18 Russell Impagliazzo and Avi Wigderson. Randomness vs Time: Derandomization under a Uniform Assumption. *J. Comput. Syst. Sci.*, 63(4):672–688, 2001. doi:10.1006/jcss.2001.1780.

19 Marek Karpinski and Rutger Verbeek. On the Monte Carlo Space Constructible Functions and Seperation Results for Probabilistic Complexity Classes. *Inf. Comput.*, 75(2):178–189, 1987. doi:10.1016/0890-5401(87)90057-5.

20 Makoto Kikuchi. Kolmogorov complexity and the second incompleteness theorem. *Archive for Mathematical Logic*, 36(6):437–443, 1997.

21 Adam R. Klivans and Dieter van Melkebeek. Graph Nonisomorphism Has Subexponential Size Proofs Unless the Polynomial-Time Hierarchy Collapses. *SIAM J. Comput.*, 31(5):1501–1526, 2002. doi:10.1137/S0097539700389652.

22 Shira Kritchman and Ran Raz. The surprise examination paradox and the second incompleteness theorem. *Notices of the AMS*, 57(11):1454–1458, 2010.

23 Leonid A. Levin. Universal sequential search problems. *Problemy Peredachi Informatsii*, 9(3):115–116, 1973.

24 Leonid A. Levin. Randomness Conservation Inequalities; Information and Independence in Mathematical Theories. *Information and Control*, 61(1):15–37, 1984. doi:10.1016/S0019-9958(84)80060-1.

25 Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Texts in Computer Science. Springer, 2008. doi:10.1007/978-0-387-49820-1.

26 Caterina E. Mora and Hans J. Briegel. Algorithmic Complexity and Entanglement of Quantum States. *Phys. Rev. Lett.*, 95:200503, 2005. doi:10.1103/PhysRevLett.95.200503.

27 Noam Nisan and Avi Wigderson. Hardness vs Randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994. doi:10.1016/S0022-0000(05)80043-1.

28 Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. *Electronic Colloquium on Computational Complexity* (ECCC), 25:158, 2018. URL: https://eccc.weizmann.ac.il/report/2018/158.

29 Igor Carboni Oliveira and Rahul Santhanam. Conspiracies Between Learning Algorithms, Circuit Lower Bounds, and Pseudorandomness. In *Computational Complexity Conference* (CCC), pages 18:1–18:49, 2017. doi:10.4230/LIPIcs.CCC.2017.18.

30 Igor Carboni Oliveira and Rahul Santhanam. Pseudodeterministic constructions in subexponential time. In *Symposium on Theory of Computing* (STOC), pages 665–677, 2017. doi:10.1145/3055399.3055500.

31 Igor Carboni Oliveira and Rahul Santhanam. Hardness Magnification for Natural Problems. In *Symposium on Foundations of Computer Science* (FOCS), pages 65–76, 2018. doi:10.1109/FOCS.2018.00016.

32 Karl Svozil. Quantum Algorithmic Information Theory. *J. UCS*, 2(5):311–346, 1996. doi:10.3217/jucs-002-05-0311.

33 Luca Trevisan and Salil P. Vadhan. Pseudorandomness and Average-Case Complexity Via Uniform Reductions. *Computational Complexity*, 16(4):331–364, 2007. doi:10.1007/s00037-007-0233-x.

34 Christopher Umans. Pseudo-random generators for all hardnesses. *J. Comput. Syst. Sci.*, 67(2):419–440, 2003. doi:10.1016/S0022-0000(03)00046-1.

35 Paul M. B. Vitányi. Quantum Kolmogorov complexity based on classical descriptions. *IEEE Trans. Information Theory*, 47(6):2464–2479, 2001. doi:10.1109/18.945258.

The Power of Block-Encoded Matrix Powers: Improved Regression Techniques via Faster Hamiltonian Simulation

Shantanav Chakraborty
QuIC, Université libre de Bruxelles, Belgium
shchakra@ulb.ac.be

András Gilyén
QuSoft/CWI, The Netherlands
gilyen@cwi.nl

Stacey Jeffery
QuSoft/CWI, The Netherlands
jeffery@cwi.nl

Abstract

We apply the framework of block-encodings, introduced by Low and Chuang (under the name standard-form), to the study of quantum machine learning algorithms and derive general results that are applicable to a variety of input models, including sparse matrix oracles and matrices stored in a data structure. We develop several tools within the block-encoding framework, such as singular value estimation of a block-encoded matrix, and quantum linear system solvers using block-encodings. The presented results give new techniques for Hamiltonian simulation of non-sparse matrices, which could be relevant for certain quantum chemistry applications, and which in turn imply an exponential improvement in the dependence on precision in quantum linear systems solvers for non-sparse matrices.

In addition, we develop a technique of variable-time amplitude *estimation*, based on Ambainis' variable-time amplitude amplification technique, which we are also able to apply within the framework.

As applications, we design the following algorithms: (1) a quantum algorithm for the quantum weighted least squares problem, exhibiting a 6-th power improvement in the dependence on the condition number and an exponential improvement in the dependence on the precision over the previous best algorithm of Kerenidis and Prakash; (2) the first quantum algorithm for the quantum generalized least squares problem; and (3) quantum algorithms for estimating electrical-network quantities, including effective resistance and dissipated power, improving upon previous work.

2012 ACM Subject Classification Theory of computation → Quantum computation theory

Keywords and phrases Quantum algorithms, Hamiltonian simulation, Quantum machine learning

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.33

Category Track A: Algorithms, Complexity and Games

Related Version Full version of this submission is available at: https://arxiv.org/abs/1804.01973.

Acknowledgements The authors are grateful to Iordanis Kerendis, Anupam Prakash and Michael Walter for useful discussions. SC is supported by the Belgian Fonds de la Recherche Scientifique - FNRS under grants no F.4515.16 (QUICTIME) and R.50.05.18.F (QuantAlgo). AG is supported by ERC Consolidator Grant QPROGRESS. SJ is supported by an NWO WISE Grant and NWO Veni Innovational Research Grant under project number 639.021.752.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 33; pp. 33:1–33:14

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

1 Introduction

A rapidly growing and important class of quantum algorithms are those that use Hamiltonian simulation subroutines to solve linear algebraic problems, many with potential applications to machine learning. This subfield began with the HHL algorithm, due to Harrow, Hassidim and Lloyd [18], which solves the *quantum linear system problem* (QLS problem). In this problem, the input consists of a matrix $A \in \mathbb{R}^{N\times N}$ and a vector $\vec{b} \in \mathbb{R}^N$, in some specified format, and the algorithm should output a quantum state proportional to $\sum_{i=1}^{N} x_i|i\rangle$, where $\vec{x} = A^{-1}\vec{b}$.

The format in which the input is presented is of crucial importance. For a sparse A, given an efficient algorithm to query the i-th non-zero entry of the j-th row of A, the HHL algorithm and its subsequent improvements [2, 14] can solve the QLS problem in complexity that depends poly-logarithmically on N. Here, if A were given naively as a list of all its entries, it would generally take time proportionally to N^2 just to read the input. We will refer to the model of accessing A, in which we can query the i-th non-zero entry of the j-th row, as the *sparse-access input model*.[1]

In [19] and [20], Kerenidis and Prakash consider several linear algebraic problems in a different input model. They assume that data has been collected and stored in some carefully chosen data structure in advance. If the data is described by an arbitrary $N \times N$ matrix, then of course, this collection will take time at least N^2 (or, if the matrix is sparse, at least the number of non-zero entries). However, processing the data, given such a data structure, is significantly cheaper, depending only poly-logarithmically on N. Kerenidis and Prakash describe a data structure that, when stored in quantum-random-access read-only memory (QROM)[2], allows for the preparation of a superposition over N data points in complexity poly-logarithmic in N. We call this the *quantum data structure input model* and discuss it more in Section 2.2. Although in some applications it might be too much to ask for the data to be presented in such a structure, one advantage of this input model is that it is not restricted to sparse matrices. This result can potentially also be useful for some quantum chemistry applications, since a recent proposal of Babbush et al. [4] uses a database of all Hamiltonian terms in order to simulate the electronic structure.

The HHL algorithm and its variants and several other applications are based on techniques from Hamiltonian simulation. Given a Hermitian matrix H and an input state $|\psi\rangle$, the Hamiltonian simulation problem is to simulate the unitary e^{iH} on $|\psi\rangle$ for some time t. Most work in this area has considered the sparse-access input model [22, 1, 6, 7, 5, 8, 12, 13, 15, 26, 31, 9, 25, 10], but recent work of Low and Chuang [24] has considered a different model, which we call *the block-encoding framework*[3].

The block-encoding framework. A block-encoding of a matrix $A \in \mathbb{C}^{N\times N}$ is a unitary U such that the top left block of U is equal to A/α for some normalizing constant $\alpha \geq \|A\|$:

$$U = \begin{pmatrix} A/\alpha & \cdot \\ \cdot & \cdot \end{pmatrix},$$

i.e. $(\langle 0|^{\otimes a} \otimes I)U(|0\rangle^{\otimes a} \otimes = A/\alpha$. In other words, for some a, for any state $|\psi\rangle$ of appropriate dimension, $\alpha(\langle 0|^{\otimes a} \otimes I)U(|0\rangle^{\otimes a} \otimes |\psi\rangle) = A|\psi\rangle$.

[1] If the matrix is not symmetric (or Hermitian) we also assume access to its transpose in a similar fashion.
[2] This refers to memory that is only required to store classical (non-superposition) data, but can be addressed in superposition.
[3] Low and Chuang call this input model *standard form*.

Such an encoding is useful if U can be implemented efficiently. In that case, U, combined with amplitude amplification, can be used to generate the state $A|\psi\rangle/\||A|\psi\rangle\|$ given a circuit for generating $|\psi\rangle$. The main motivation for using block-encodings is that Low and Chuang showed [24] how to perform optimal Hamiltonian simulation given a block-encoded Hamiltonian A.

In Ref. [19], Kerenidis and Prakash implicitly prove that if an $N \times N$ matrix A is given as a quantum data structure, then there is an ε-approximate block-encoding of A that can be implemented in complexity $\text{polylog}(N/\varepsilon)$. This implies that all results about block-encodings – including Low and Chuang's Hamiltonian simulation when the input is given as a block-encoding [24], and other techniques we develop in this paper – also apply to input presented in the quantum data structure model. This observation is the essential idea behind our applications. Implicit in work by Childs [13] is the fact that, given A in the sparse-access input model, there is an ε-approximate block-encoding of A that can be implemented in complexity $\text{polylog}(N/\varepsilon)$, so our results also apply to the sparse-access input model. In fact, the block-encoding framework unifies a number of possible input models, and also enables one to work with hybrid input models, where some matrices may come from purifications of density operators, whereas other input matrices may be accessed through sparse oracles or a quantum data structure. For a very recent overview of these general techniques see e.g. [16].

We demonstrate the elegance of the block-encoding framework by showing how to combine and modify block-encodings to build up new block-encodings, similar to building new algorithms from existing subroutines. For example, given block-encodings of A and B, their product yields a block-encoding of AB. Given a block-encoding of a Hermitian A, it is possible to construct a block-encoding of e^{iA}, using which one can implement a block-encoding of A^{-1}. We present these techniques in Section 3.

To illustrate the elegance of the block-encoding framework, consider one of our applications: generalized least squares. This problem, defined in Section 4, requires that given inputs $X \in \mathbb{R}^{M\times N}$, $\Omega \in \mathbb{R}^{M\times M}$ and $\vec{y} \in \mathbb{R}^M$, we output a quantum state proportional to

$$\vec{\beta} = (X^T\Omega^{-1}X)^{-1}X^T\Omega^{-1}\vec{y}.$$

Given block-encodings of X and Ω, it is simple to combine them to get a block-encoding of $(X^T\Omega^{-1}X)^{-1}X^T\Omega^{-1}$, which can then be applied to a quantum state proportional to $\vec{y}$.

Variable-time amplitude estimation. A variable-stopping-time quantum algorithm is a quantum algorithm $\mathcal{A}$ consisting of m stages $\mathcal{A} = \mathcal{A}_m \dots \mathcal{A}_1$, where $\mathcal{A}_j\mathcal{A}_{j-1}\dots\mathcal{A}_1$ has complexity t_j, for $t_m > \dots > t_1 > 0$. At each stage, a certain flag register, which we can think of as being initialized to a neutral symbol, may be marked as "good" in some branches of the superposition, or "bad" in some branches of the superposition, or left neutral. Each subsequent stage only acts non-trivially on those branches of the superposition in which the flag is not yet set to "good" or "bad".

At the end of the algorithm, we would like to project onto that part of the final state in which the flag register is set to "good". This is straightforward using amplitude amplification, however this approach may be vastly sub-optimal. If the algorithm terminates with amplitude $\sqrt{p_{succ}}$ on the "good" part of the state, then standard amplitude amplification requires that we run $1/\sqrt{p_{succ}}$ rounds, each of which requires us to run the full algorithm $\mathcal{A}$ to generate its final state, costing $t_m/\sqrt{p_{succ}}$.

To see why this might be sub-optimal, suppose that after $\mathcal{A}_1$, the amplitude on the part of the state in which the flag register is set to "bad" is already very high. Using amplitude amplification at this stage is very cheap, because we only have to incur the cost

t_1 of $\mathcal{A}_1$ at each round, rather than running all of $\mathcal{A}$. In [2], Ambainis showed that given a variable-stopping-time quantum algorithm, there exists an algorithm that approximates the "good" part of the algorithm's final state in cost $\widetilde{\mathcal{O}}\left(t_m + \sqrt{\sum_{j=1}^{m} \frac{p_j}{p_{succ}} t_j^2}\right)$[4], where p_j is the amplitude on the part of the state that is moved from neutral to "good" or "bad" during application of $\mathcal{A}_j$ (intuitively, the probability that the algorithm stops at stage j).

While amplitude amplification can easily be modified to not only project a state onto its "good" part, but also return an estimate of p_{succ} (i.e. the probability of measuring "good" given the output of $\mathcal{A}$), this is not immediate in variable-time amplitude amplification. The main difficulty is that a variable-time amplification algorithm applies a lot of subsequent amplification phases, where in each amplification phase the precise amount of amplification is a priori unknown. We overcome this difficulty by separately estimating the amount of amplification in each phase with some additional precision and finally combining the separate estimates in order to get a multiplicative estimate of p_{succ}.

We prove rigorously in the full version of this paper [11], how to estimate the success probability of a variable-stopping-time quantum algorithm to within a multiplicative error of ε in complexity

$$\widetilde{\mathcal{O}}\left(\frac{1}{\varepsilon}\left(t_m + \sqrt{\sum_{j=1}^{m} \frac{p_j}{p_{succ}} t_j^2}\right)\right).$$

Meanwhile we also derive some logarithmic improvements to the complexity of variable-time amplitude amplification.

Applications. We give several applications of the block-encoding framework and variable-time amplitude estimation.

We first present a quantum weighted least squares solver (WLS solver), which outputs a quantum state proportional to the optimal solution to a weighted least squares problem, when the input is given either in the quantum data structure model of Kerenidis and Prakash, or the sparse-access input model. We remark that the sparse-access input model is perhaps less appropriate to the setting of data analysis, where we cannot usually assume any special structure on the input data, however, since our algorithm is designed in the block-encoding framework, it works for either input model. Our quantum WLS solver improves the dependence on the condition number from κ^6 in [20][5] to κ, and the dependence on ε from $1/\varepsilon$ to $\mathrm{polylog}(1/\varepsilon)$.

We next present the first quantum generalized least squares solver (GLS solver), which outputs a quantum state proportional to the optimal solution to a generalized least squares problem. We again assume that the input is given in either the quantum data structure model or the sparse-access model. The complexity is again polynomial in $\log(1/\varepsilon)$ and in the condition numbers of the input matrices. We describe our WLS and GLS solvers in Section 4.

We build on the algorithms of Wang [29] to estimate effective resistance between two nodes of an electrical network and the power dissipated across a network when the input is given as a quantum data structure or in the sparse-access model. We estimate the norm of the output state of a certain linear system by applying the variable-time amplitude estimation algorithm. In the sparse-access model, we find that our algorithm outperforms Wang's linear-system-based algorithm. In the quantum data structure model, our algorithms offer

[4] We use the notation $\widetilde{\mathcal{O}}(f(x))$ to indicate $\mathcal{O}(f(x)\mathrm{polylog}(f(x)))$.
[5] In the paper of Kerenidis and Prakash their κ corresponds to our κ^2.

a speedup whenever the maximum degree of an electrical network of n nodes is $\Omega(n^{1/3})$. Our algorithms also have a speedup over the quantum walk based algorithm by Wang in certain regimes.

Throughout the article, the theorems, lemmas, and corollaries that are provided without a reference, are all rigorously proven in the full version of this paper [11].

Related Work. Independently of this work, recently, Wang and Wossnig [28] have also considered Hamiltonian simulation of a Hamiltonian given in the quantum data structure model, using quantum-walk based techniques from earlier work on Hamiltonian simulation [9]. Their algorithm's complexity scales as $\|A\|_1$ (which they upper bound by $\sqrt{N}$); whereas our Hamiltonian simulation results (Theorem 8), which follow from Low and Chuang's block-Hamiltonian simulation result, have a complexity that depends poly-logarithmically on the dimension, N. Instead, our complexity depends on the parameter μ, described below, which is also at most $\sqrt{N}$.

2 Preliminaries

For $A \in \mathbb{C}^{M\times N}$, define $\overline{A} \in \mathbb{C}^{(M+N)\times(M+N)}$ by

$$\overline{A} = \begin{bmatrix} 0 & A \\ A^\dagger & 0 \end{bmatrix}. \tag{1}$$

For many applications where we want to simulate A, or a function of A, it suffices to simulate $\overline{A}$.

For $A \in \mathbb{C}^{N\times N}$, we will let $\|A\|$ denote the spectral norm and $\|A\|_F$ the Frobenius norm. For $A \in \mathbb{C}^{M\times N}$, let $A_{i,\cdot}$ denote the i-th row of A, and define the following:

- For $q \in [0,1]$, $s_q(A) = \max_{i\in M} \|A_{i,\cdot}\|_q^q$
- For $p \in [0,1]$, $\mu_p(A) = \sqrt{s_{2p}(A)s_{2(1-p)}(A^T)}$
- $\sigma_{\min}(A) = \min\{\||A|u\rangle\| : |u\rangle \in \mathrm{row}(A), \||u\rangle\| = 1\}$ (the smallest non-zero singular value)
- $\sigma_{\max}(A) = \max\{\||A|u\rangle\| : \||u\rangle\| = 1\}$ (the largest singular value)
- $\|A\| = \left\|\overline{A}\right\| = \sigma_{\max}(A)$

For $A \in \mathbb{C}^{M\times N}$ with singular value decomposition $A = \sum_i \sigma_i |u_i\rangle\langle v_i|$, we define the Moore-Penrose pseudoinverse of A by $A^+ = \sum_i \sigma_i^{-1}|v_i\rangle\langle u_i|$. For a matrix A, we let $A^{(p)}$ be defined $A^{(p)}_{i,j} = (A_{i,j})^p$.

2.1 Block-encodings

Following [16] we use the following definition:

▶ **Definition 1** (Block-encoding). *Suppose that A is an s-qubit operator, $\alpha, \varepsilon \in \mathbb{R}_+$ and $a \in \mathbb{N}$. Then we say that the $(s+a)$-qubit unitary U is an (α, a, ε)-block-encoding[6] of A, if*

$$\left\|A - \alpha(\langle 0|^{\otimes a} \otimes I)U(|0\rangle^{\otimes a} \otimes I)\right\| \leq \varepsilon.$$

Block-encodings are really intuitive to work with. For example, one can easily take the product of two block-encoded matrices by keeping their ancilla qubits separately. The following lemma shows that the errors during such a multiplication simply add up as one would expect, and the block-encoding does not introduce any additional errors.

[6] Note that since $\|U\| = 1$ we necessarily have $\|A\| \leq \alpha + \varepsilon$.

▶ **Lemma 2.** *If U is an (α, a, δ)-block-encoding of an s-qubit operator A, and V is a (β, b, ε)-block-encoding of an s-qubit operator B then*[7] *$(I_b \otimes U)(I_a \otimes V)$ is an $(\alpha\beta, a+b, \alpha\varepsilon + \beta\delta)$-block-encoding of AB.*

Proof.

$$\begin{aligned}
&\left\|AB - \alpha\beta(\langle 0|^{\otimes a+b} \otimes I)(I_b \otimes U)(I_a \otimes V)(|0\rangle^{\otimes a+b} \otimes I)\right\| \\
=&\left\|AB - \underbrace{\alpha(\langle 0|^{\otimes a} \otimes I)U(|0\rangle^{\otimes a} \otimes I)}_{\tilde{A}}\underbrace{\beta(\langle 0|^{\otimes b} \otimes I)V(|0\rangle^{\otimes b} \otimes I)}_{\tilde{B}}\right\| \\
=&\left\|AB - \tilde{A}B + \tilde{A}B - \tilde{A}\tilde{B}\right\| = \left\|(A-\tilde{A})B + \tilde{A}(B-\tilde{B})\right\| = \left\|A-\tilde{A}\right\|\beta + \alpha\left\|B-\tilde{B}\right\| \\
\leq&\alpha\varepsilon + \beta\delta.
\end{aligned}$$

◀

The above lemma can be made more efficient in some cases when both A and B are significantly subnormalized.

The following theorem about block-Hamiltonian simulation is a corollary of the results of [24, Theorem 1], which also includes bounds on the propagation of errors.

▶ **Theorem 3.** *Suppose that U is an $(\alpha, a, \varepsilon/|2t|)$-block-encoding of the Hamiltonian H. Then we can implement an ε-precise Hamiltonian simulation unitary V which is an $(1, a+2, \varepsilon)$-block-encoding of e^{itH}, with $\mathcal{O}(|\alpha t| + \log(1/\varepsilon))$ uses of controlled-U or its inverse and with $\mathcal{O}(a|\alpha t| + a\log(1/\varepsilon))$ two-qubit gates.*

2.2 Data structures and sparse access

We will consider the following data structure, studied in [19]. We will refer to this data structure as a *quantum-accessible* data structure, because it is a classical data structure, which, if stored in QROM, is addressable in superposition, but needn't be able to store a quantum state, facilitates the implementation of certain useful quantum operations. In our complexity analysis, we consider the cost of accessing a QROM of size N to be polylog(N). Although this operation requires order N gates [17, 3], but the gates can be arranged in parallel such that the depth of the circuit indeed remains polylog(N).

The following is proven in [19].

▶ **Theorem 4** (Implementing quantum operators using an efficient data structure [19])**.** *Let $A \in \mathbb{R}^{M\times N}$ be a matrix with $A_{ij} \in \mathbb{R}$ being the entry of the i-th row and the j-th column. If w is the number of non-zero entries of A, then there exists a data structure of size*[8] *$\mathcal{O}\left(w\log^2(MN)\right)$ that, given the entries (i, j, A_{ij}) in an arbitrary order, stores them such that time*[8] *taken to store each entry of A is $\mathcal{O}(\log(MN))$. Once this data structure has been initiated with all non-zero entries of A, there exists a quantum algorithm that can perform the following maps with ε-precision in $\mathcal{O}(polylog(MN/\varepsilon))$ time:*

$$\widetilde{U} : |i\rangle|0\rangle \mapsto |i\rangle \frac{1}{\|A_{i,\cdot}\|}\sum_{j=1}^{N} A_{i,j}|j\rangle = |i, A_i\rangle,$$

[7] In the expression $(I_b \otimes U)(I_a \otimes V)$, the identity operator I_b should be seen as acting on the ancilla qubits of V, and I_a on those of U.

[8] Here, for simplicity we assume that we can store a real number in 1 data register, however more realistically we should actually count the number of bits, incurring logarithmic overheads. Also in this theorem we assign unit cost for classical arithmetic operations.

$$\widetilde{V} : |0\rangle|j\rangle \mapsto \frac{1}{\|A\|_F} \sum_{i=1}^{M} \|A_{i,\cdot}\| \, |i\rangle|j\rangle = |\widetilde{A}, j\rangle,$$

where $|A_{i,\cdot}\rangle$ *is the normalized quantum state corresponding to the* i*-th row of* A *and* $|\widetilde{A}\rangle$ *is a normalized quantum state such that* $\langle i|\widetilde{A}\rangle = \|A_{i,\cdot}\|$*, i.e. the norm of the* i*-th row of* A*.*

In particular, given a vector $\vec{v} \in \mathbb{R}^{M\times 1}$ *stored in this data structure, we can generate an* ε*-approximation of the superposition* $\sum_{i=1}^{M} v_i|i\rangle / \|\vec{v}\|$ *in complexity* $\mathrm{polylog}(M/\varepsilon)$*.*

As a corollary, we have the following, which allows us to generate alternative quantum state representations of the rows of A, as long as we have stored A appropriately beforehand:

▶ **Corollary 5.** *If* $A^{(p)}$ *is stored in a quantum data structure, then there exists a quantum algorithm that can perform the following map with* ε*-precision in* $\mathrm{polylog}(MN/\varepsilon)$ *time:*

$$|i\rangle|0\rangle \mapsto |i\rangle \frac{1}{s_{2p}(A)} \sum_{j=1}^{N} A_{i,j}^{p} |j\rangle.$$

The following was proven in [20], although not in the language of block-encodings.

▶ **Lemma 6** (Implementing block-encodings from quantum data structures)**.** *Let* $A \in \mathbb{C}^{M\times N}$*.*

1. *Fix* $p \in [0,1]$*. If* $A \in \mathbb{C}^{M\times N}$*, and* $A^{(p)}$ *and* $(A^{(1-p)})^\dagger$ *are both stored in quantum-accessible data structures*[9]*, then there exist unitaries* U_R *and* U_L *that can be implemented in time* $\mathcal{O}\left(\mathrm{polylog}(MN/\varepsilon)\right)$ *such that* $U_R^\dagger U_L$ *is a* $(\mu_p(A), \lceil \log(N+M+1)\rceil, \varepsilon)$*-block-encoding of* $\overline{A}$*.*
2. *On the other hand, if* A *is stored in a quantum-accessible data structure*[9]*, then there exist unitaries* U_R *and* U_L *that can be implemented in time* $\mathcal{O}\left(\mathrm{polylog}(MN)/\varepsilon\right)$ *such that* $U_R^\dagger U_L$ *is a* $(\|A\|_F, \lceil \log(M+N)\rceil, \varepsilon)$*-block-encoding of* $\overline{A}$*.*

This allows us to apply our block-encoding results in the quantum data structure setting, including Hamiltonian simulation (Section 3.1), quantum linear system solvers (Section 3.3) and implementing negative powers of a Hamiltonian (Section 3.3).

In contrast, in the *sparse-access model* we assume that the input matrix $A \in \mathbb{C}^{M\times N}$ has s_r-sparse rows and s_c-sparse columns, such that the matrix elements can be queried via an oracle

$$\mathrm{O}_A : |i\rangle|j\rangle|0\rangle^{\otimes b} \mapsto |i\rangle|j\rangle|a_{ij}\rangle \qquad \forall i \in [M], j \in [N].$$

Moreover, the indices of non-zero elements of each row can be queried via an oracle

$$\mathrm{O}_r : |i\rangle|k\rangle \mapsto |i\rangle|r_{ik}\rangle \qquad \forall i \in [N], k \in [s_r], \text{ where}$$

r_{ij} is the index for the j-th non-zero entry of the i-th row of A, or if there are less than i non-zero entries, then it is $j+N$. If A is not symmetric (or Hermitian) then we also assume the analogous oracle for columns. It is not difficult to prove [13] that a block-encoding of A can be efficiently implemented in the sparse-access input model, see [16, Lemma 48] for a direct proof.

▶ **Lemma 7** (Constructing block-encodings for sparse-access matrices [16, Lemma 48])**.** *Let* $A \in \mathbb{C}^{M\times N}$ *be an* s^r, s^c *row and column-sparse matrix given in the sparse-access input model. Then for any* $\varepsilon \in (0,1)$*, we can implement a* $(\sqrt{s^r s^c}, \mathrm{polylog}(MN/\varepsilon), \varepsilon)$*-block-encoding of* A *with* $\mathcal{O}(1)$ *queries and* $\mathrm{polylog}(MN/\varepsilon)$ *elementary gates.*

Thus, our block-encoding results also apply to the sparse access model.

[9] Here we assume that the data structure stores the matrices with sufficient precision.

3 Techniques for block-encodings

We develop several tools within the block-encoding framework that are crucial to our applications, but also likely of independent interest. Since an input given either in the sparse-access model or as a quantum data structure can be made into a block-encoding, our block-encoding results imply analogous results in each of the sparse-access and quantum data structure models.

Throughout this section, let $\mu(A)$, and "μ-encoding of A" be one of:

(1) $\mu(A) = \|A\|_F$, the Frobenius norm of A, in which case μ-encoding refers to a quantum data structure encoding A;

(2) for some $p \in [0,1]$, $\mu(A) = \sqrt{s_{2p}(A)s_{2(1-p)}(A)}$, where $s_p(A) = \max_j \|A_{j,\cdot}\|_p^p$, in which case μ-encoding refers to quantum data structures encoding both $A^{(p)}$ and $(A^{(1-p)})^T$, defined by $A^{(q)}_{i,j} := (A_{i,j})^q$; or

(3) $\mu(A) = \sqrt{s^r s^c}$, where s^r and s^c are the row and column sparsities of A, in which case, μ-encoding refers to having sparse access to A.

3.1 Hamiltonian simulation from quantum data structure

We first have the following important building block for our other results:

▶ **Theorem 8.** *For any $t \in \mathbb{R}$ and $\varepsilon \in (0, 1/2)$, let $H \in \mathbb{C}^{N\times N}$ be a Hermitian matrix that is μ-encoded, with $\|H\| \leq 1$. Then we can implement a unitary $\widetilde{U}$ that is a $(1, n+3, \varepsilon)$-block-encoding of e^{itH} in time $\widetilde{\mathcal{O}}\left(t\mu(A)\mathrm{polylog}(N/\varepsilon)\right)$.*

This follows from the quantum Hamiltonian simulation algorithm of Low and Chuang that expects the input as a block-encoding, and Lemmas 6 and 7. Independently, Wang and Wossnig have proven a similar result, with $\|A\|_1 \leq \sqrt{N}$ in place of $\mu(A)$ [28].

3.2 Quantum singular value estimation

Given access to a matrix $A \in \mathbb{R}^{M\times N}$ with singular value decomposition $A = \sum_j \sigma_j |u_j\rangle\langle v_j|$, and given some input state, the quantum singular value estimation (QSVE) problem requires estimating the singular values of A up to some precision with a high probability. We present a quantum algorithm for singular value estimation of a matrix A given as a block-encoding. In particular, using our algorithm it is possible to obtain an estimate of σ_j, $\tilde{\sigma}_j$ such that with probability $1-\varepsilon$, $|\sigma_j - \tilde{\sigma}_j| \leq \Delta$. We give a precise description of quantum singular value estimation, and prove the following theorem:

▶ **Theorem 9.** *Let $\varepsilon, \Delta \in (0,1)$, and $\varepsilon' = \frac{\varepsilon\Delta}{4\log^2(1/\Delta)}$. Let U be an $(\alpha, a, \varepsilon')$-block-encoding of a matrix A that can be implemented in cost T_U. Then we can implement a quantum algorithm that solves QSVE of A in complexity*

$$\mathcal{O}\left(\frac{\alpha}{\Delta}(a+T_U)\mathrm{polylog}(1/\varepsilon)\right).$$

In the special case when the block-encoding is implemented by a quantum data structure, we recover the complexity of the quantum algorithm for singular value estimation by Kerenidis and Prakash [19].

3.3 Quantum linear system solver

The quantum linear system problem (QLS problem) is the following. Given access to an $N \times N$ matrix A, and a procedure for computing a quantum state $|b\rangle$ in the image of A, prepare a state that is within ε of $A^+|b\rangle / \|A^+|b\rangle\|$. Given a block-encoding of A, we can use Theorem 3 to get a block-encoding of e^{itA}, from which we can implement a $(2\kappa, a, \varepsilon)$-block-encoding of A^{-1} (for some a and ε), where κ is an upper bound on[10] $\left\|A^{-1}\right\|$. Such a block-encoding can be applied to a state $|b\rangle$ to get $\frac{1}{2\kappa}|0\rangle^{\otimes a}(A^{-1}|b\rangle) + |0^\perp\rangle$ for some unnormalized state $|0^\perp\rangle$ orthogonal to every state with $|0\rangle$ in the first a registers. Performing amplitude amplification on this procedure, we can approximate the state $A^{-1}|b\rangle / \left\|A^{-1}|b\rangle\right\|$. However, this gives quadratic dependence on κ, whereas only linear dependence is needed for quantum linear systems solvers in the sparse-access input model, thanks to the technique of variable-time amplitude amplification. Using this technique, we are able to show the following:

▶ **Theorem 10.** *Let $\kappa \geq 2$, and A be an $N \times N$ Hermitian matrix[11] such that the non-zero eigenvalues of H lie in the range $[-1, -1/\kappa] \bigcup [1/\kappa, 1]$. Suppose that for $\delta = o\left(\varepsilon/(\kappa^2 \log^3 \frac{\kappa}{\varepsilon})\right)$ we have a unitary U that is a (α, a, δ)-block-encoding of A that can be implemented using T_U elementary gates. Also suppose that we can prepare an input state $|\psi\rangle$ which spans the eigenvectors of A in time T_ψ. Then there exists a variable time amplitude amplification based quantum algorithm that outputs a state that is ε-close to $A^{-1}|\psi\rangle / \left\|A^{-1}|\psi\rangle\right\|$ at a cost*

$$\mathcal{O}\left(\kappa\left(\alpha\big(T_U + a\big)\log^2\left(\frac{\kappa}{\varepsilon}\right) + T_\psi\right)\log(\kappa)\right).$$

From this we get the following theorem:

▶ **Theorem 11.** *Let $\varepsilon \in (0, 1/2)$, suppose that $A \in \mathbb{C}^{M \times N}$ such that $\|A\| \leq 1$, $\left\|A^{-1}\right\| \leq \kappa$, and A is μ-encoded. Also assume that there is a unitary U which acts on $\mathrm{polylog}(MN/\varepsilon)$ qubits and prepares the state $|b\rangle$ with complexity T_b. Then*

- **(i)** *The QLS problem can be solved in time $\widetilde{\mathcal{O}}\left(\kappa\left(\mu(A) + T_b\right)\mathrm{polylog}(MN/\varepsilon)\right)$.*
- **(ii)** *If $\varepsilon \in (0, 1)$, then an ε-multiplicative approximation of $\|A^+|b\rangle\|$ can be obtained in time $\widetilde{\mathcal{O}}\left(\frac{\kappa}{\varepsilon}\left(\mu(A) + T_b\right)\mathrm{polylog}(MN)\right)$*

For (ii), we use our new technique of variable time amplitude estimation.

Finally, we generalize our QLS solver to apply A^{-c} for any $c \in (0, \infty)$. Using variable-time amplification techniques we show the following:

▶ **Theorem 12.** *Let $\kappa \geq 2$, $c \in (0, \infty)$, $q = \max(1, c)$, and A be an $N \times N$ Hermitian matrix such that the eigenvalues of A lie in the range $[-1, -1/\kappa] \bigcup [1/\kappa, 1]$. Suppose that for $\delta = o\left(\varepsilon/\left(\kappa^q q \log^3 \frac{\kappa^q}{\varepsilon}\right)\right)$ we have a unitary U that is a (α, a, δ)-block-encoding of A which can be implemented using T_U elementary gates. Also suppose that we can prepare an input state $|\psi\rangle$ that is spanned by the eigenvectors of A in time T_ψ. Then there exists a variable time amplitude amplification based quantum algorithm that outputs a state that is ε-close to $A^{-c}|\psi\rangle / \|A^{-c}|\psi\rangle\|$ with a cost of*

$$\mathcal{O}\left(\left(\alpha\kappa^q\big(T_U + a\big)q\log^2\left(\frac{\kappa^q}{\varepsilon}\right) + \kappa^c T_\psi\right)\log\left(\kappa\right)\right).$$

[10] In the special case when $\|A\| = 1$, κ is an upper bound on the condition number of A, justifying the notation.

[11] Since for any matrix $C \in \mathbb{C}^{M' \times N'}$ we have that $\overline{C} \in \mathbb{C}^{(M'+N') \times (M'+N')}$ is Hermitian, and the eigenvalues of $\overline{C}$ are ± 1 times the singular values of C, this statement and its corollaries also apply to non-symmetric matrices.

Also, there exists a variable time amplitude amplification based quantum algorithm that outputs a number Γ *such that*

$$1-\varepsilon \leq \frac{\Gamma}{\|A^{-c}|\psi\rangle\|} \leq 1+\varepsilon,$$

with success probability at least $1-\delta$*, at a cost*

$$\mathcal{O}\left(\frac{1}{\varepsilon}\left(\alpha\kappa^q\left(T_U+a\right)q\log^2\left(\frac{\kappa^q}{\varepsilon}\right)+\kappa^c T_\psi\right)\log^3(\kappa)\log\left(\frac{\log(\kappa)}{\delta}\right)\right).$$

4 Application to least squares

The problem of *ordinary least squares (OLS)* is the following. Given data points $\{(\vec{x}^{(i)}, y^{(i)})\}_{i=1}^M$ for $\vec{x}^{(1)},\ldots,\vec{x}^{(M)} \in \mathbb{R}^N$ and $y^{(1)},\ldots,y^{(M)} \in \mathbb{R}$, find $\vec{\beta} \in \mathbb{R}^N$ that minimizes:

$$\sum_{i=1}^{M}(y^{(i)} - \vec{\beta}^T\vec{x}^{(i)})^2. \tag{2}$$

The motivation for this task is the assumption that the samples are obtained from some process such that at every sample i, $y^{(i)}$ depends linearly on $\vec{x}^{(i)}$, up to some random noise, so $y^{(i)}$ is drawn from a random variable $\vec{\beta}^T\vec{x}^{(i)} + E_i$, where E_i is a random variable with mean 0, for example, a Gaussian. The vector $\vec{\beta}$ that minimizes (2) represents a good estimate of the underlying linear function. We assume $M \geq N$ so that it is feasible to recover this linear function.

We can generalize this task to settings in which certain samples are thought to be of higher quality than others, for example, because the random variables E_i are not identical. We express this belief by assigning a positive weight w_i to each sample, and minimizing

$$\sum_{i=1}^{M} w_i(y^{(i)} - \vec{\beta}^T\vec{x}^{(i)})^2. \tag{3}$$

Let $X \in \mathbb{R}^{M\times N}$ be the matrix such that its i^{th} row is $\vec{x}^{(i)T}$. Finding $\vec{\beta}$ given X, $\vec{w}$ and $\vec{y}$ is the problem of *weighted least squares (WLS)*.

We can further generalize to settings in which the random variables E_i for sample i are correlated. In the problem of *generalized least squares (GLS)*, the presumed correlations in error between pairs of samples are given in a symmetric non-singular covariance matrix Ω. We then want to find the vector $\vec{\beta}$ that minimizes

$$\sum_{i,j=1}^{M} \Omega^{-1}_{i,j}(y^{(i)} - \vec{\beta}^T\vec{x}^{(i)})(y^{(j)} - \vec{\beta}^T\vec{x}^{(j)}). \tag{4}$$

We will consider solving *quantum* versions of these problems. Specifically, a *quantum WLS solver* (resp. *quantum GLS solver*) is given access to $\vec{y} \in \mathbb{R}^M$, $X \in \mathbb{R}^{M\times N}$, and positive weights $w_1,\ldots,w_M$ (resp. Ω), in some specified manner, and outputs an ε-approximation of a quantum state $\sum_i \beta_i|i\rangle / \left\|\vec{\beta}\right\|$, where $\vec{\beta}$ minimizes the expression in (3) (resp. (4)).

Quantum algorithms for least squares fitting were first considered in [32]. They considered query access to X, and a procedure for outputting $|y\rangle = \sum_i y_i|i\rangle / \|\vec{y}\|$, which we refer to as the sparse-access input model. They present a *quantum* OLS solver, outputting a state

proportional to a solution $\vec{\beta}$, that runs in time $\widetilde{\mathcal{O}}\left(\min\{\log(M)s^3\kappa^6/\varepsilon, \log(M)s\kappa^6/\varepsilon^2\}\right)$, where s is the sparsity of X, and κ the condition number. To compute a state proportional to $\vec{\beta}$, they first apply X^T to $|y\rangle$ to get a state proportional to $X^T\vec{y}$, using techniques similar to [18]. They then apply $(X^TX)^{-1}$ using the quantum linear system solving algorithm of [18], giving a final state proportional to $(X^TX)^{-1}X^T\vec{y} = X^+\vec{y}$.

The approach of [32] was later improved upon by [21], who also give a quantum OLS solver in the sparse-access input model. Unlike [32], they apply X^+ directly, by using Hamiltonian simulation of X and phase estimation to estimate the singular values of X, and then apply a rotation depending on the inverse singular value if it's larger than 0, and using amplitude amplification to de-amplify the singular-value-zero parts of the state. This results in an algorithm with complexity $\widetilde{\mathcal{O}}\left(s\kappa^3\log(M+N)/\varepsilon^2\right)$.

Several works have also considered quantum algorithms for least squares problems with a classical output. The first, due to Wang [30], outputs the vector $\vec{\beta}$ in a classical form. The input model should be compared with the sparse-access model – although $\vec{y}$ is given in classical random access memory, an assumption about the regularity of $\vec{y}$ means the quantum state $|y\rangle$ can be efficiently prepared. The algorithm also requires a regularity condition on the matrix X. The algorithm's complexity is $\mathrm{poly}(\log M, N, \kappa, \frac{1}{\varepsilon})$. Like [21], Wang's algorithm uses techniques from quantum linear system solving to apply X^+ directly to $|y\rangle$. To do this, Hamiltonian simulation of X is accomplished via what we would call a block-encoding of X. This outputs a state proportional to $X^+\vec{y}$, whose amplitudes can be estimated one-by-one to recover $\vec{\beta}$.

A second algorithm to consider least squares with a classical output is [27], which does not output $\vec{\beta}$, but rather, given an input $\vec{x}$, outputs $\vec{x}^T\vec{\beta}$, thus predicting a new data point. This algorithm requires that $\vec{x}$, $\vec{y}$, and even X be given as quantum states, and assumes that X has low approximate rank. The algorithm uses techniques from quantum principal component analysis [23], and runs in time $\mathcal{O}\left(\log(N)\kappa^2/\varepsilon^3\right)$.

Recently, Kerenidis and Prakash introduced the quantum data structure input model [19]. This input model fits data analysis tasks, because unlike in more abstract problems such as Hamiltonian simulation, where the input matrix may be assumed to be sparse and well-structured so that we can hope to have implemented efficient subroutines to find the non-zero entries of the rows and columns, the input to least squares is generally noisy data for which we may not assume any such structure. In Ref. [20], utilizing this data structure, they solve the quantum version of the *weighted* least squares problem. Their algorithm assumes access to quantum data structures storing X, or some closely related matrix (see Section 2.2), $W = \mathrm{diag}(\vec{w})$, and $\vec{y}$, and have running time $\widetilde{\mathcal{O}}\left(\frac{\kappa^6\mu}{\varepsilon}\mathrm{polylog}(MN)\right)$, where κ is the condition number of $X^T\sqrt{W}$, and μ is some prior choice of $\left\|X^T\sqrt{W}\right\|_F$ or $\sqrt{s_{2p}(X^T\sqrt{W})s_{2(1-p)}(X^T\sqrt{W})}$ for some $p \in [0,1]$[12]. Note that the choice of μ impacts the way X must be encoded, leading to a family of algorithms requiring slightly different encodings of the input.

[12] We stress that our algorithms do not achieve the minimum possible μ, but rather, we need to store the input in QROM with a particular μ in mind. We might more accurately describe the quantum data structure input model as a *family* of input models, parametrized by μ.

4.1 Our results

We give quantum WLS and GLS solvers in the model where the input is given as a block-encoding. As a special case, we get quantum WLS and GLS solvers in the quantum data structure input model of Kerenidis and Prakash. First, we give the following WLS solver:

▶ **Theorem 13.** *Let $A = \sqrt{W}X$ such that $\|A^+\| \leq \kappa_A$. Suppose $\sqrt{W}\vec{y}$ is stored in a quantum-accessible data structure, and A is μ-encoded. Suppose the data points have residual error $\overline{SS}^W_{\text{res}}$ satisfying $\overline{SS}^W_{\text{res}} \leq \eta$. Then we can implement a quantum WLS solver with error ε in complexity:*

$$\widetilde{\mathcal{O}}\left(\frac{\kappa_A \mu(A)}{\sqrt{1-\eta}}\text{polylog}\left(MN/\varepsilon\right)\right).$$

The *residual error* is a measure of how well the data can be linearly approximated. It is reasonable to assume that it's close to 0, as otherwise, linear regression is inappropriate. Indeed, previous work seems to implicitly assume it's bounded by a constant below 1.

Theorem 13 is a 6-th power improvement in the dependence on κ, and an exponential improvement in the dependent on $1/\varepsilon$ as compared with the quantum WLS solver of [20]. As a special case we get a quantum OLS solver, which compares favourably to previous quantum OLS solvers in the sparse-access model [32, 21] in having a linear dependence on κ, and a polylog$(1/\varepsilon)$ dependence on the precision. However, these previous results rely on QLS solver subroutines which have since been improved, so their complexity can also likely be improved.

In addition, we give the first quantum GLS solver. We first show how to implement a GLS solver when the inputs are given as block-encodings. We prove the following general Theorem:

▶ **Theorem 14.** *Suppose that we have a unitary U_y preparing a quantum state proportional to $\vec{y}$ in complexity T_y. Suppose $X \in \mathbb{R}^{M\times N}$, $\Omega \in \mathbb{R}^{M\times M}$ are such that $\|X\| \leq 1$, $\|\Omega\| \leq 1$ and $\Omega \succ 0$ is positive definite. Suppose that we have access to U_X that is an $(\alpha_X, a_X, 0)$-block-encoding of X which has complexity $T_X \geq a_X$, and similarly we have access U_Ω that is an $(\alpha_\Omega, a_\Omega, 0)$-block-encoding of $\Omega^{-\frac{1}{2}}$ which has complexity $T_\Omega \geq a_\Omega$. Let $A := \Omega^{-\frac{1}{2}}X$, and suppose we have the following upper bounds: $\|A^+\| \leq \kappa_A$, $\left\|\Omega^{-1}\right\| \leq \kappa_\Omega$, and $\overline{SS}^\Omega_{\text{res}} \leq \eta$. Then we can implement a quantum GLS-solver with error ε in complexity*

$$\mathcal{O}\left(\frac{\kappa_A \log\left(\kappa_A\right)}{\sqrt{1-\eta}}\left(\left(\sqrt{\kappa_\Omega}\alpha_X T_X + \alpha_\Omega T_\Omega\right)\log^3\left(\frac{\kappa_A}{\varepsilon}\right) + \sqrt{\kappa_\Omega}T_y\right)\right).$$

As a special case, we get the following:

▶ **Corollary 15.** *Suppose $\vec{y}$ is stored in a quantum-accessible data structure, and X, Ω are such that $\|X\| \leq 1$, $\|\Omega\| \leq 1$ and Ω is positive definite. Further assume they are μ-encoded and $\|X^+\| \leq \kappa_X$, $\|\Omega^+\| \leq \kappa_\Omega$. Then we can implement a quantum GLS-solver with error ε in complexity*

$$\widetilde{\mathcal{O}}\left(\frac{\kappa_X\sqrt{\kappa_\Omega}}{\sqrt{1-\eta}}\left(\mu(X) + \mu(\Omega)\kappa_\Omega\right)\text{polylog}\left(MN/\varepsilon\right)\right).$$

Note that for Theorem 14 and Corollary 15, the parameter κ_X is not exactly the condition number of X. In fact, what we require is that the product of the upper bounds on $\|X\|$ and $\|X^+\|$ respectively, be upper bounded by κ_X. Thus without loss of generality it suffices to consider that $\|X\| \leq 1$ and $\|X^+\| \leq \kappa_X$ (same holds for Ω).

References

1 Dorit Aharonov and Amnon Ta-Schma. Adiabatic quantum state generation and statistical zero-knowledge. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC 2003)*, pages 20–29, 2003. doi:10.1145/780542.780546.

2 Andris Ambainis. Variable time amplitude amplification and quantum algorithms for linear algebra problems. In *Symposium on Theoretical Aspects of Computer Science STACS*, pages 636–647, 2012. doi:10.4230/LIPIcs.STACS.2012.636.

3 Srinivasan Arunachalam, Vlad Gheorghiu, Tomas Jochym-O'Connor, Michele Mosca, and Priyaa Varshinee Srinivasan. On the robustness of bucket brigade quantum RAM. *New Journal of Physics*, 17(12):123010, 2015. arXiv:1502.03450.

4 Ryan Babbush, Dominic W Berry, Ian D Kivlichan, Annie Y Wei, Peter J Love, and Alán Aspuru-Guzik. Exponentially more precise quantum simulation of fermions in second quantization. *New Journal of Physics*, 18(3):033032, 2016. arXiv:1506.01020.

5 D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma. Exponential improvement in precision for simulating sparse Hamiltonians. In *Symposium on Theory of Computing, STOC 2014*, pages 283–292, 2014. doi:10.1145/2591796.2591854.

6 Dominic W. Berry, Graeme Ahokas, Richard Cleve, and Barry C. Sanders. Efficient quantum algorithms for simulating sparse Hamiltonians. *Communications in Mathematical Physics*, 270(2):359–371, 2007. doi:10.1007/s00220-006-0150-x.

7 Dominic W. Berry and Andrew M. Childs. Black-box Hamiltonian simulation and unitary implementation. *Quantum Information and Computation*, 12(1–2):29–62, 2012. arXiv:0910.4157.

8 Dominic W. Berry, Andrew M. Childs, Richard Cleve, Robin Kothari, and Rolando D. Somma. Simulating Hamiltonian Dynamics with a Truncated Taylor Series. *Phys. Rev. Lett.*, 114:090502, 2015. doi:10.1103/PhysRevLett.114.090502.

9 Dominic W. Berry, Andrew M. Childs, and Robin Kothari. Hamiltonian Simulation with Nearly Optimal Dependence on all Parameters. In *FOCS*, pages 792–809, 2015. doi:10.1109/FOCS.2015.54.

10 Dominic W. Berry and Leonardo Novo. Corrected quantum walk for optimal Hamiltonian simulation. *Quantum Information and Computation*, 16(15–16):1295–1317, 2016. doi:10.26421/QIC16.15-16.

11 Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. The power of block-encoded matrix powers: improved regression techniques via faster Hamiltonian simulation, 2018. arXiv:1804.01973.

12 Andrew M. Childs. *Quantum information processing in continuous time*. PhD thesis, Massachusetts Institute of Technology, 2004. URL: https://dspace.mit.edu/handle/1721.1/16663.

13 Andrew M. Childs. On the relationship between continuous- and discrete-time quantum walk. *Communications in Mathematical Physics*, 294(2):581–603, 2010. doi:10.1007/s00220-009-0930-1.

14 Andrew M. Childs, Robin Kothari, and Rolando D. Somma. Quantum Algorithm for Systems of Linear Equations with Exponentially Improved Dependence on Precision. *SIAM Journal on Computing*, 46(6):1920–1950, 2017. doi:10.1137/16M1087072.

15 Andrew M. Childs and Nathan Wiebe. Hamiltonian simulation using linear combinations of unitaries. *Quantum Information and Computation*, 12(11–12):901–924, 2012. arXiv:1202.5822.

16 András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st ACM Symposium on Theory of Computing (STOC)*, 2019. (to appear). arXiv:1806.01838.

17 Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum Random Access Memory. *Phys. Rev. Lett.*, 100:160501, April 2008. doi:10.1103/PhysRevLett.100.160501.

18 Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009. doi:10.1103/PhysRevLett.103.150502.

19 Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. *CoRR*, 2016. arXiv:1603.08675.

20 Iordanis Kerenidis and Anupam Prakash. Quantum gradient descent for linear systems and least squares, 2017. arXiv:1704.04992.

21 Yang Liu and Shengyu Zhang. Fast Quantum Algorithms for Least Squares Regression and Statistic Leverage Scores. *Theor. Comput. Sci.*, 657(PA):38–47, 2017. doi:10.1016/j.tcs.2016.05.044.

22 Seth Lloyd. Universal quantum simulators. *Science*, 273(5278):1073–1078, 1996. doi:10.1126/science.273.5278.1073.

23 Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(631), 2014. doi:10.1038/nphys3029.

24 Guang Hao Low and Isaac L. Chuang. Hamiltonian Simulation by Qubitization, 2016. arXiv:1610.06546.

25 Leonardo Novo and Dominic W. Berry. Improved Hamiltonian simulation via a truncated Taylor series and corrections. *Quantum Information and Computation*, 17(7–8):0623–0635, 2016. doi:10.26421/QIC17.7-8.

26 David Poulin, Angie Qarry, Rolando Somma, and Frank Verstraete. Quantum Simulation of Time-Dependent Hamiltonians and the Convenient Illusion of Hilbert Space. *Phys. Rev. Lett.*, 106:170501, April 2011. doi:10.1103/PhysRevLett.106.170501.

27 Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. Prediction by linear regression on a quantum computer. *Physical Review A*, 94:022342, 2016. doi:10.1103/PhysRevA.94.022342.

28 Chunhao Wang and Leonard Wossnig. A quantum algorithm for simulating non-sparse Hamiltonian, 2018. arXiv:1803.08273.

29 Guoming Wang. Efficient Quantum Algorithms for Analyzing Large Sparse Electrical Networks. *Quantum Information and Computation*, 11 and 12:987–1026, 2017. doi:10.26421/QIC17.11-12.

30 Guoming Wang. Quantum algorithm for linear regression. *Physical Review A*, 96:012335, 2017. doi:10.1103/PhysRevA.96.012335.

31 Nathan Wiebe, Dominic W. Berry, Peter Høyer, and Barry C. Sanders. Simulating Hamiltonian dynamics on a quantum computer. *Journal of Physics A*, 44(44):445308, 2011. arXiv:1011.3489.

32 Nathan Wiebe, Daniel Braun, and Seth Lloyd. Quantum algorithm for data fitting. *Physical review letters*, 109(5):050505, 2012. doi:10.1103/PhysRevLett.109.050505.

Unlabeled Sample Compression Schemes and Corner Peelings for Ample and Maximum Classes

Jérémie Chalopin
CNRS, Aix-Marseille Université, Université de Toulon, LIS, Marseille, France
jeremie.chalopin@lis-lab.fr

Victor Chepoi
Aix-Marseille Université, CNRS, Université de Toulon, LIS, Marseille, France
victor.chepoi@lis-lab.fr

Shay Moran
Department of Computer Science, Princeton University, Princeton, USA
shaym@cs.princeton.edu

Manfred K. Warmuth
Computer Science Department, University of California, Santa Cruz, USA
manfred@ucsc.edu

Abstract

We examine connections between combinatorial notions that arise in machine learning and topological notions in cubical/simplicial geometry. These connections enable to export results from geometry to machine learning. Our first main result is based on a geometric construction by H. Tracy Hall (2004) of a partial shelling of the cross-polytope which can not be extended. We use it to derive a maximum class of VC dimension 3 that has no corners. This refutes several previous works in machine learning from the past 11 years. In particular, it implies that the previous constructions of optimal unlabeled compression schemes for maximum classes are erroneous.

On the positive side we present a new construction of an optimal unlabeled compression scheme for maximum classes. We leave as open whether our unlabeled compression scheme extends to ample (a.k.a. lopsided or extremal) classes, which represent a natural and far-reaching generalization of maximum classes. Towards resolving this question, we provide a geometric characterization in terms of unique sink orientations of the 1-skeletons of associated cubical complexes.

2012 ACM Subject Classification Mathematics of computing → Combinatorics; Theory of computation → Machine learning theory; Theory of computation → Computational geometry

Keywords and phrases VC-dimension, sample compression, Sauer-Shelah-Perles lemma, Sandwich lemma, maximum class, ample/extremal class, corner peeling, unique sink orientation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.34

Category Track A: Algorithms, Complexity and Games

Related Version A full version of this paper is available at `http://arxiv.org/abs/1812.02099`.

Funding The research of J.C. and V.C. is supported by the ANR project DISTANCIA (ANR-17-CE40-0015). The research of S.M. is supported by the Simons Foundation and the NSF; part of this project was carried while S.M. was at the Institute for Advanced Study and was supported by the National Science Foundation under agreement No. CCF-1412958. The research of M.W. is supported by the NSF grant IIS-1619271.

Acknowledgements The authors are grateful to Olivier Bousquet for insightful discussions and to the anonymous referees for useful remarks that helped improving the presentation of this work.

1 Introduction

The Sauer-Shelah-Perles Lemma [28, 30, 32] is arguably the most basic fact in VC theory; it asserts that any class $C \subseteq \{0,1\}^n$ satisfies $|C| \leq \binom{n}{\leq d}$, where $d = \text{VC-dim}(C)$. A beautiful generalization of Sauer-Shelah-Perles's inequality asserts that $|C| \leq |\overline{X}(C)|$, where $\overline{X}(C)$ is the family of subsets that are shattered by C.[1] The latter inequality is a part of the Sandwich Lemma [1, 4, 7, 23], which also provides a lower bound for $|C|$ (and thus "sandwiches" $|C|$) in terms of the number of its *strongly shattered subsets* (see Section 2). A class C is called *maximum/ample* if the Sauer-Shelah-Perles/Sandwich upper bounds are tight (respectively). Every maximum class is ample, but not vice versa.

Maximum classes were studied mostly in discrete geometry and machine learning, e.g. [34, 11, 9, 10, 14]. The history of ample classes is more interesting as they were discovered independently by several works in disparate contexts [1, 15, 4, 19, 3, 7, 35]. Consequently, they received different names such as lopsided classes [15], extremal classes [4, 19], and ample classes [3, 7]. Lawrence [15] was the first to define them for the investigation of the possible sign patterns realized by points of a convex set of $\mathbb{R}^d$. Interestingly, Lawrence's definition of these classes does not use the notion of shattering nor the Sandwich Lemma. In this context, these classes were discovered by Bollobás and Radcliffe [4] and Bandelt et al. [3], and the equivalence between the two definitions appears in [3]. Ample classes admit a multitude of combinatorial and geometric characterizations [3, 4, 15] and comprise many natural examples arising from discrete geometry, combinatorics, graph theory, and geometry of groups [3, 15].

Main Results

Corner Peelings. A *corner* in an ample class C is any concept $c \in C$ that belongs to a unique maximal cube of C (equivalently, c is a corner if $C \setminus \{c\}$ is also ample, see Claim 6). A sequence of corner removals leading to a single concept is called a *corner peeling*. Wiedemann [35] and independently Chepoi (unpublished, 1996) asked whether every ample class has a corner. The machine learning community studied this question independently in the context of *sample compression schemes* for maximum classes: Rubinstein and Rubinstein [25] showed that corner peelings lead to optimal *unlabeled sample compression schemes (USCS)*.

In Theorem 9 we refute this conjecture. The crux of the proof is an equivalence between corner peelings and partial shellings of the cross-polytope. This equivalence translates the question whether corners always exist to the question whether partial shellings can always be extended. The latter was an open question in Ziegler's book on polytopes [38], and was resolved in H. Tracy Hall's PhD thesis where she presented an interesting counterexample [13]. The ample class resulting from Hall's construction yields a maximum class without corners.

Sample Compression. Sample compression is a powerful technique to derive generalization bounds in statistical learning. Littlestone and Warmuth [16] introduced it and asked if every class of VC-dimension $d < \infty$ has a sample compression scheme of a finite size. This question was later precised by Floyd and Warmuth [10, 33] to *whether a sample compression scheme of size $O(d)$ exists*. The first question was recently resolved by [21] who exhibited an $\exp(d)$ sample compression. The second question however remains one of the oldest open problems in machine learning (for more background we refer the reader to [20] and the books [29, 36]).

[1] Note that this inequality indeed implies the Sauer-Shelah-Perles Lemma, since $|\overline{X}(C)| \leq \binom{n}{\leq d}$.

Rubinstein and Rubinstein [25, Theorem 16] showed that the existence of a corner peeling for a maximum class C implies a *representation map* for C (see Section 3 for a definition), which is known to yield an optimal unlabeled sample compression scheme of size VC-dim(C) [14].[2] They claim, using an interesting topological approach, that maximum classes admit corner peelings. Unfortunately, our Theorem 9 shows that this does not hold.

While our Theorem 9 rules out the program of deriving representation maps from corner peelings, in Theorem 10 we provide an alternative derivation of representation maps for maximum classes and therefore also of an unlabeled sample compression scheme for them.

Sample Compression and Unique Sink Orientations. We next turn to construction of representation maps for ample classes. In Theorem 15 we present geometric characterizations of such maps via *unique sink orientations*: an orientation of the edges of a cube B is a *unique sink orientation (USO)* if any subcube $B' \subseteq B$ has a unique sink. Szabó and Welzl [31] showed that any USO of B leads to a representation map for B. We extend this bijection to ample classes C by proving that representation maps are equivalent to orientations o of C such that (i) o is a USO on each subcube $B \subseteq C$, and (ii) for each $c \in C$ the edges outgoing from c belong to a subcube $B \subseteq C$. We further show that any ample class admits orientations satisfying each one of those conditions. However, *the question whether all ample classes admit representation maps remains open.*

Implications on Previous Works. Our Theorem 9 establishes the existence of maximum classes without any corners, thus countering several previous results in machine learning:

- Rubinstein and Rubinstein [25, Theorem 32] showed that any maximum class can be represented by a simple arrangement of piecewise-linear hyperplanes. In [25, Theorem 39], they claim that sweeping such an arrangement leads to a corner peeling of the corresponding maximum class. This is unfortunately false, as witnessed by Theorem 9.
- Kuzmin and Warmuth [14] constructed unlabeled sample compression schemes for maximum classes based on the presumed uniqueness of a certain matching (their Theorem 10). This theorem is wrong as it implies the existence of corners. However their conclusion is correct: in our Theorem 10 we show that such unlabeled compression schemes exist.
- Theorem 3 by Samei, Yang, and Zilles [27] is built on a generalization of Theorem 10 from [14] to the multiclass case which is also incorrect.
- Theorem 26 by Doliwa et al. [6] uses the result by [25] to show that the Recursive Teaching Dimension (RTD) of maximum classes equals to their VC dimension. However the VC dimension 3 maximum class from Theorem 9 has RTD at least 4. It remains open whether the RTD of every maximum class C is bounded by $O(\text{VC-dim}(C))$.

Organization. Section 2 presents the main definitions and notations. Section 3 reviews characterizations of ample/maximum classes and presents characteristic examples. Section 4 demonstrates the existence of the maximum class C_H without corners. Section 5 establishes the existence of representation maps for maximum classes. Section 6 establishes a bijection between representation maps and unique sink orientations for ample classes. Due to space limitations, some proofs are omitted and can be found in the full version of this paper [5].

[2] Pálvölgyi and Tardos [24] recently exhibited a (non-ample) class C with no USCS of size VC-dim(C).

2 Preliminaries

A *concept class* C is a set of subsets (concepts) of a finite ground set U which is called the *domain* of C and denoted $\mathrm{dom}(C)$. We sometimes treat the concepts as characteristic functions rather than subsets. The *support* (or *dimension set*) $\mathrm{supp}(C)$ of C is the set $\{x \in U : x \in c' \setminus c'' \text{ for some } c', c'' \in C\}$. $C^* := 2^U \setminus C$ is the *complement* of C. The *restriction* of C on $Y \subseteq U$ is the class $C|Y = \{c \cap Y : c \in C\}$ whose domain is Y. We use C_Y as shorthand for $C|(U \setminus Y)$; in particular, we write C_x for $C_{\{x\}}$, and c_x for $c|(U \setminus \{x\})$ for $c \in C$ (note that $c_x \in C_x$). A class $B \subseteq 2^U$ is a *cube* if there exists $Y \subseteq U$ such that $B|Y = 2^Y$ and B_Y contains a single concept (denoted by $\mathrm{tag}(B)$). Note that $\mathrm{supp}(B) = Y$ and therefore we say that B is a *Y-cube*; $|Y|$ is called the *dimension* $\dim(B)$ of B. Two cubes B, B' with the same support are called *parallel cubes*. A cube B is *maximal* if there is no cube B' such that $B \subsetneq B'$.

Let Q_n denote the n-dimensional cube where $n = |U|$; $c, c' \in Q_n$ are called adjacent if the symmetric difference $c \Delta c'$ is of size 1. The *1-inclusion graph* of C is the subgraph $G(C)$ of Q_n induced by the vertex-set C when the concepts of C are identified with the corresponding vertices of Q_n. Any cube $B \subseteq C$ is called *a cube of* C. The *cube complex* of C is the set $Q(C) = \{B : B \text{ is a cube of } C\}$. The *dimension* of $Q(C)$ is $\dim(Q(C)) := \max_{B \in Q(C)} \dim(B)$. A concept $c \in C$ is called a *corner* of C if c belongs to a unique maximal cube of C. The *reduction* C^Y of C to $Y \subseteq U$ is the class $C^Y := \{\mathrm{tag}(B) : B \in Q(C) \text{ and } \mathrm{supp}(B) = Y\}$ whose domain is $U \setminus Y$. When $x \in U$ we denote $C^{\{x\}}$ by C^x and call it *the x-hyperplane of* C. Note that a concept c belongs to C^x if and only if c and $c \cup \{x\}$ both belong to C. The union of all cubes of C having x in their support is called the *carrier* of C^x and is denoted by $N_x(C)$. If $c \in N_x(C)$, we also denote $c|U \setminus \{x\}$ by c^x (note that $c^x \in C^x$).

A class C is *connected* if the graph $G(C)$ is connected. Let $d_{G(C)}(c, c')$ denote the distance between c and c' in $G(C)$. Note that $d_{Q_n}(c, c') =: d(c, c')$ coincides with the Hamming distance $|c \Delta c'|$. Let $B(c, c') = \{t \subseteq U : d(c, t) + d(t, c') = d(c, c')\}$ be the *interval* between c and c' in Q_n. A class C is called *isometric* if $d(c, c') = d_{G(C)}(c, c')$ for any $c, c' \in C$ and *weakly isometric* if $d(c, c') = d_{G(C)}(c, c')$ if $d(c, c') \leq 2$. Any path connecting two concepts $\mathrm{tag}(B)$ and $\mathrm{tag}(B')$ of C^Y inside C^Y can be lifted to a path of Y-cubes connecting B and B' in C; such a path of cubes is called a *gallery*.

A class C *shatters* $Y \subseteq U$ if $C|Y = 2^Y$. C *strongly shatters* Y if C contains a Y-cube. Let $\overline{X}(C)$, $\underline{X}(C)$ denote the simplicial complexes $\overline{X}(C) = \{Y : C \text{ shatters } Y\}$, $\underline{X}(C) = \{Y : C \text{ strongly shatters } Y\}$. Note that $\underline{X}(C) \subseteq \overline{X}(C)$. The *VC-dimension* $\text{VC-dim}(C)$ of C is the size of a largest set C shatters. The *Sandwich Lemma* asserts that $|\underline{X}(C)| \leq |C| \leq |\overline{X}(C)|$.

A *labeled sample* is a set $s = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, where $x_i \in U$ and $y_i \in \{0, 1\}$. An *unlabeled sample* is a set $\{x_1, \ldots, x_m\}$, where $x_i \in U$. Given a labeled sample $s = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, the unlabeled sample $\{x_1, \ldots, x_m\}$ is the domain of s and is denoted by $\mathrm{dom}(s)$. A sample s is *realizable* by a concept $c : U \to \{0, 1\}$ if $c(x_i) = y_i$ for every i, and s is realizable by a concept class C if it is realizable by some $c \in C$.

A *sample compression scheme* for a concept class C is best viewed as a protocol between a *compressor* and a *reconstructor*. The compressor gets a realizable sample s from which it picks a small subsample s'. The compressor sends s' to the reconstructor. Based on s', the reconstructor outputs a concept c that needs to be consistent with the entire input sample s. A sample compression scheme has size k if for every realizable input sample s the size of the compressed subsample s' is at most k. An *unlabeled (sample) compression scheme* (USCS) is a sample compression scheme in which the compressed subsample s' is unlabeled. So, the compressor removes the labels before sending the subsample to the reconstructor.

3 Ample and Maximum Classes

We briefly review the main characterizations and basic geometric examples of ample and maximum classes. The next theorem summarizes the main characterizations of ample classes:

▶ **Theorem 1** ([3, 4, 15]). *The following are equivalent for a class C: (1) C is ample; (2) C^* is ample; (3) $\underline{X}(C) = \overline{X}(C)$; (4) $|\underline{X}(C)| = |C|$; (5) $|\overline{X}(C)| = |C|$; (6) $C \cap B$ is ample for any cube B; (7) $(C^Y)_Z = (C_Z)^Y$ for all partitions $U = Y \cup Z$; (8) for all partitions $U = Y \cup Z$, either $Y \in \underline{X}(C)$ or $Z \in \underline{X}(C^*)$.*

Condition (3) leads to a simple definition of ampleness: C is ample if whenever $Y \subseteq U$ is shattered by C, then there is a Y-subcube of C. Thus, if C is ample we will write $X(C)$ instead of $\underline{X}(C) = \overline{X}(C)$. A *representation map* for an ample class C is a bijection $r : C \to X(C)$ satisfying the *non-clashing condition*: $c|(r(c) \cup r(c')) \neq c'|(r(c) \cup r(c'))$, for all $c, c' \in C, c \neq c'$. We continue with metric and recursive characterizations of ample classes:

▶ **Theorem 2** ([3]). *The following are equivalent for a class C: (1) C is ample; (2) C^Y is connected for all $Y \subseteq U$; (3) C^Y is isometric for all $Y \subseteq U$; (4) C is isometric, and both C_x and C^x are ample for all $x \in U$; (5) C is connected and all hyperplanes C^x are ample.*

▶ **Corollary 3.** *Two maximal cubes of an ample class C have different supports.*

Indeed, if B and B' are two d-cubes with the same support, by Theorem 2(2) B and B' can be connected in C by a gallery, and thus B is contained in a $d+1$-cube. Therefore, B and B' cannot be maximal.

The Sandwich Lemma and Theorem 1(5) imply that maximum classes are ample. Basic examples of maximum classes are concept classes derived from hyperplane arrangements in $\mathbb{R}^n$, ball arrangements in $\mathbb{R}^n$, and unions of n intervals in $\mathbb{R}$. The following theorem summarizes some characterizations of maximum classes provided in [11, 9, 10, 34]:

▶ **Theorem 4.** *The following are equivalent for a class C: (1) C is maximum; (2) C_Y is maximum for all $Y \subseteq U$; (3) C_x and C^x are maximum for all $x \in U$; (4) C^* is maximum.*

We continue with some important geometric examples of ample classes.

1. Simplicial Complexes. Every simplicial complex S (viewed as a set system closed under taking subsets) is ample since $\overline{X}(S) = \underline{X}(S)$.

2. Realizable Ample Classes. Let $K \subseteq \mathbb{R}^n$ be a convex set. Let $C(K) := \{\text{sign}(v) : v \in K, v_i \neq 0 \, \forall i \leq n\}$, where $\text{sign}(v) \in \{\pm 1\}^n$ is the sign pattern of v. Lawrence [15] showed that $C(K)$ is ample, and called ample classes representable in this manner *realizable*.

3. Median Classes. A class C is called *median* if for every three concepts c_1, c_2, c_3 of C their *median* $m(c_1, c_2, c_3) := (c_1 \cap c_2) \cup (c_1 \cap c_3) \cup (c_2 \cap c_3)$ also belongs to C. Median classes are ample by [3, Proposition 2]. Due to their relationships with other discrete structures, median classes are one of the most important examples of ample classes. Median classes are equivalent to finite median graphs (a well-studied class in metric graph theory, see [2]), to CAT(0) cube complexes, i.e., cube complexes of global nonpositive curvature (central objects in geometric group theory, see [12, 26]), and to the domains of event structures (a basic model in concurrency theory [22, 37]).

4. Convex Geometries and Conditional Antimatroids. Let C be a class such that (i) $\varnothing \in C$ and (ii) $c, c' \in C$ implies that $c \cap c' \in C$. Call $x \in c \in C$ *extremal* if $c \setminus \{x\} \in C$. We say that $c \in C$ is *generated* by $s \subseteq c$ if c is the smallest member of C containing s. A class C satisfying (i) and (ii) with the additional property that every member c of C is generated by its extremal points is called a *conditional antimatroid* [3, Section 3]. If $U \in C$, then we obtain the well-known structure of a *convex geometry* (called also an *antimatroid*) [8]. By [3, Proposition 1], conditional antimatroids C are ample since $\underline{X}(C)$ coincides with the sets of extremal points and $\overline{X}(C)$ coincides with the set of all minimal generating sets of sets from C. Convex geometries comprise many examples from geometry, ordered sets, and graphs; see the foundational paper [8]. For example, a *realizable convex geometry* is a convex geometry $C \subseteq U$ such that U can be realized as a set of $\mathbb{R}^n$ and $c \in C$ if and only if c is the intersection of a convex set of $\mathbb{R}^n$ with U.

4 Corner Peelings and Partial Shellings

In this section, we prove that corner peelings of ample classes are equivalent to isometric orderings of C as well as to partial shellings of the cross-polytope. This equivalence, combined with a result by Hall [13] yields a maximum class with VC dimension 3 without corners (Theorem 9 below). Let $C_< := (c_1, \ldots, c_m)$ be an ordering of the concepts in C. For any $1 \leq i \leq m$, let $C_i := \{c_1, \ldots, c_i\}$ denote the *i'th level set*. The ordering $C_<$ is called:

- an *ample* ordering if every level set C_i is ample;
- a *corner peeling* if every c_i is a *corner* of C_i;
- an *isometric* ordering if every level set C_i is isometric;
- a *weakly isometric* ordering if every level set C_i is weakly isometric.

▶ **Proposition 5.** *The following are equivalent for an ordering $C_<$ of an isometric class C: (1) $C_<$ is ample; (2) $C_<$ is a corner peeling; (3) $C_<$ is isometric; (4) $C_<$ is weakly isometric.*

Proof. Clearly, (3)⇒(4). Conversely, suppose $C_<$ is weakly isometric but one of its levels is not isometric. Hence, there exists $i < j$ such that any shortest (c_i, c_j)-path in C contains some c_k with $k > j$. Additionally, assume that c_i, c_j minimizes the distance $d(c_i, c_j)$ among all such pairs. Since C_j is weakly isometric, necessarily $d(c_i, c_j) \geq 3$. Let c_r be the first concept among $\{c_{j+1}, \ldots, c_m\}$ lying in $B(c_i, c_j) \cap C$. If $d(c_i, c_r) \geq 3$ or $d(c_r, c_j) \geq 3$ (say the first), then one can replace c_i, c_j by c_i, c_r, which contradicts the choice of c_i, c_j. Thus, $d(c_i, c_r), d(c_r, c_j) \leq 2$, and at least one of them equals 2 (say $d(c_i, c_r) = 2$). Now, weak isometricity implies that c_i and c_r have a common neighbor c_ℓ with $\ell < \max\{i, r\} = r$. If $\ell < j$ then c_ℓ, c_j contradicts the minimality of c_i, c_j, and if $j < \ell < r$ then c_ℓ contradicts the minimality of c_r. This shows (4)⇒(3).

If c_i is a corner of C_i, then any two neighbors of c_i in C_i have a second common neighbor in C_i, and therefore $d_{G(C_{i-1})}$ is the restriction of $d_{G(C_i)}$ on C_{i-1}. Since $C_m = C$ is isometric, this proves (2)⇒(3). We now prove (3)⇒(1)⇒(2) using the next lemma. For $t \notin C$, let $F[t]$ be the smallest cube of Q_n containing t and all neighbors of t in $G(C)$. Note that the dimension of $F[t]$ is the number of neighbors of t in $G(C)$.

▷ Claim 6. Let C be ample. Then: *(i)* If $t \notin C$ then $F[t] \subseteq C \cup \{t\}$. *(ii)* If c is a corner of C then $C \setminus \{c\}$ is ample. *(iii)* If $t \notin C$ and $C' := C \cup \{t\}$ is isometric then C' is ample and t is a corner of C'.

Proof. Item (i): Suppose $F[t] \setminus C \neq \{t\}$. Pick $s \neq t$ that is closest to t in $F[t] \setminus C$ (with respect to the Hamming distance of Q_n). Then t and s are not adjacent (by the definition of $F[t]$). By the choice of s, $B(s,t) \setminus \{s,t\} \subseteq C$, i.e., $B(s,t) \cap C^* = \{t,s\}$, contrary to the ampleness of C^*.

Item (ii): If $c \in C$ is a corner then there is a unique maximal cube $F \subseteq C$ containing it. Combined with Corollary 3, this implies that $\underline{X}(C \setminus \{c\}) = \underline{X}(C) \setminus \{\mathrm{supp}(F)\}$. Next, since $|C| = |\underline{X}(C)|$, we get that $|C \setminus \{c\}| = |\underline{X}(C \setminus \{c\})|$, and by Theorem 1 $C \setminus \{c\}$ is ample.

Item (iii): To prove that C' is ample, we use Theorem 2(2). First note that by item (i), $F[t] \subseteq C'$. Let $F' \neq F''$ be parallel cubes of C'. If $t \notin F' \cup F''$, then a gallery connecting F' and F'' in C is a gallery in C'. So, assume $t \in F'$. If F' is a proper face of $F[t]$, then F' is parallel to a face F of $F[t]$ not containing t. Since F' and F are connected in $F[t]$ by a gallery and F and F'' are connected in C by a gallery, we obtain a gallery between F' and F'' in C'. Finally, let $F' = F[t]$. In this case, we assert that F'' does not exist. Otherwise, let π be the parallelism map between F' and F'' (π maps each concept in F' to its unique closest concept in F''). Note that for any $r \in F'$: $d(t,\pi(t)) = d(r,\pi(r)) = d(F',F'')$. Since C' is isometric, t and $\pi(t)$ can be connected in C' by a path P of length $d(t,\pi(t))$. Let s be the neighbor of t in P. Since $s \in C$ it follows that $s \in F[t] = F'$. So, s is a concept in F' that is closer to $\pi(t)$ than t; this contradicts that $d(t,t'') = d(F',F'')$. ◁

To show (1)⇒(2), let $C_<$ be an ample order of C. We assert that each c_i is a corner of C_i. Indeed, since C_{i-1} is ample and $c_i \notin C_{i-1}$, by Item (i) in Claim 6 the cube $F[c_i]$, defined with respect to C_{i-1}, is included in C_i. Thus, c_i belongs to a unique maximal cube $F[c_i]$ of C_i, i.e., c_i is a corner of C_i. To prove (3)⇒(1), let $C_<$ be isometric. The ampleness of each C_i follows by induction from Item (iii) of Claim 6. ◀

A concept class C is *dismantlable* if it admits an ordering satisfying any of the equivalent conditions (1)-(4) in Proposition 5. Isometric orderings of Q_n are closely related to shellings of its dual, the *cross-polytope* O_n (which we define next). Define $\pm U := \{\pm x_1, \ldots, \pm x_n\}$; so, $|\pm U| = 2n$, and we call $-x_i, +x_i$ *antipodal*. The n-dimensional *cross-polytope* is the pure simplicial complex of dimension n whose facets are all $\sigma \subseteq \pm U$ that contain exactly one element in each antipodal pair. Thus, O_n has 2^n facets and each facet σ of O_n corresponds to a vertex c of Q_n ($+x_i \in \sigma$ if and only if $x_i \in c$). Observe that $x_i \in c' \Delta c''$ if and only if $\{+x_i, -x_i\} \subseteq \sigma' \Delta \sigma''$ where σ' correspond to c' and σ'' corresponds to c''.

Let X be a pure simplicial complex (PSC) of dimension d, i.e., a simplicial complex in which all facets have size d. Two facets σ, σ' are adjacent if $|\sigma \Delta \sigma'| = 2$. A *shelling* of X is an ordering $\sigma_1, \ldots, \sigma_p$ of all of its facets such that $2^{\sigma_j} \bigcap (\bigcup_{i<j} 2^{\sigma_i})$ is a PSC of dimension $d-1$ for every $j \leq p$ [38, Lecture 8]. A *partial shelling* is an ordering of some facets that satisfies the above condition. Note that $\sigma_1, \ldots, \sigma_m$ is a partial shelling if and only if for every $i < j$ there exists $k < j$ such that $\sigma_i \cap \sigma_j \subseteq \sigma_k \cap \sigma_j$, and $\sigma_k \cap \sigma_j$ is a facet of both σ_j and σ_k. X is *extendably shellable* if every partial shelling can be extended to a shelling. We next establish a relationship between partial shellings and isometric orderings.

▶ **Proposition 7.** *Every partial shelling of the cross-polytope O_n defines an isometric ordering of the corresponding vertices of the cube Q_n. Conversely, if C is an isometric class of Q_n, then any isometric ordering of C defines a partial shelling of O_n.*

Proof. Let $\sigma_1, \ldots, \sigma_m$ be a partial shelling of O_n and $c_1, \ldots, c_m$ be the ordering of the corresponding vertices of Q_n. We need to prove that each level set $C_j = \{c_1, \ldots, c_j\}$ is isometric. It suffices to show that for every $i < j$ there is $k < j$ such that $d(c_k, c_j) = 1$ and $c_k \in B(c_i, c_j)$. Equivalently, that $|\sigma_k \Delta \sigma_j| = 2$ and $\sigma_i \cap \sigma_j \subseteq \sigma_k \subseteq \sigma_i \cup \sigma_j$: since

$\sigma_1, \ldots, \sigma_m$ is a partial shelling, there is a facet σ_k with $k < j$ such that $|\sigma_k \cap \sigma_j| = n-1$ and $\sigma_i \cap \sigma_j \subseteq \sigma_k \cap \sigma_j$. We claim that σ_k is the desired facet. It remains to show that (i) $|\sigma_j \Delta \sigma_k| = 2$ and (ii) $\sigma_k \subseteq \sigma_i \cup \sigma_j$. Item (i) follows since $|\sigma_j| = |\sigma_k| = n$, and $|\sigma_k \cap \sigma_j| = n-1$. For Item (ii), let $\sigma_j \setminus \sigma_k = \{+x\}$ and $\sigma_k \setminus \sigma_j = \{-x\}$. We need to show that $-x \in \sigma_i$, or equivalently that $+x \notin \sigma_i$. The latter follows since $+x \in \sigma_j \setminus \sigma_k$ and $\sigma_j \cap \sigma_i \subseteq \sigma_k$.

Conversely, let $c_1, \ldots, c_m$ be an isometric ordering and $\sigma_1, \ldots, \sigma_m$ be the ordering of the corresponding facets of O_n. We assert that this is a partial shelling. Let $i < j$. It suffices to exhibit $k < j$ such that $|\sigma_k \cap \sigma_j| = n-1$ and $\sigma_i \cap \sigma_j \subseteq \sigma_k \cap \sigma_j$. Since C_j is isometric, c_j has a neighbor $c_k \in B(c_i, c_j) \cap C_j$. Since $d(c_j, c_k) = 1$ it follows that $|\sigma_k \cap \sigma_j| = n-1$. Since $c_k \in B(c_i, c_j)$ it follows that $\sigma_i \cap \sigma_j \subset \sigma_k \subset \sigma_i \cup \sigma_j$ and hence that $\sigma_i \cap \sigma_j \subseteq \sigma_k \cap \sigma_j$. ◀

▶ **Corollary 8.** *If all ample classes are dismantlable, then O_n is extendably shellable.*

Proof. Let $\sigma_1, \ldots, \sigma_m$ be a partial shelling of O_n and let $C = \{c_1, \ldots, c_m\}$ be the corresponding vertices of Q_n. By Proposition 7, the level sets are isometric, thus C is ample by Proposition 5. The complement C^* is also ample, thus dismantlable. Thus C^* contains a concept t such that $C^* \setminus \{t\}$ is ample. Consequently, $C' := C \cup \{t\}$ is ample. Let τ be the facet of O_n corresponding to t. Since $c_1, \ldots, c_m, t$ is an isometric ordering of C', by Proposition 7, $\sigma_1, \ldots, \sigma_m, \tau$ is a partial shelling of O_n. ◀

It was asked in [38] if any cross-polytope O_n is extendably shellable. In the PhD thesis of H. Tracy Hall from 2004, a nice counterexample to this question is given [13]. Hall's counterexample arises from the 299 regions of an arrangement of 12 pseudo-hyperplanes. These regions are encoded as facets of the cross-polytope O_{12} and it is shown in [13] that the subcomplex of O_{12} consisting of all other facets admits a shelling which cannot be extended. By the proof of Corollary 8, the ample concept class C_H defined by those 299 simplices does not have any corner.[3] A counting shows that C_H is a maximum class of VC-dimension 3. This completes the proof of our first main result:

▶ **Theorem 9.** *There exists a maximum class C_H of VC-dimension 3 without any corner.*

However, conditional antimatroids and 2-dimensional ample classes are dismantlable. The 2-dimensional case was proved in [25, Theorem 34] for maximum classes and in [18] for ample classes. The proof for conditional antimatroids appears in the full version of this paper [5], as well as a different proof for the 2-dimensional case that is based on a local characterization of convex sets of ample classes.

5 Representation Maps for Maximum Classes

In this section, we prove that maximum classes admit representation maps and therefore by [14, Lemma 1], they admit optimal unlabeled compression schemes.

▶ **Theorem 10.** *Any maximum class $C \subseteq 2^U$ of VC-dimension d admits a representation map, and consequently, an unlabeled compression scheme of size d.*

The crux of the proof of Theorem 10 is the following proposition. Let C be a d-dimensional maximum class and let $D \subseteq C$ be a $(d-1)$-dimensional maximum subclass. A *missed simplex* for the pair (C, D) is a simplex $\sigma \in X(C) \setminus X(D)$. Note that any missed simplex has size d.

[3] For the interested reader, a file containing the 299 concepts of C_H represented as elements of $\{0,1\}^{12}$ is available at `https://arxiv.org/src/1812.02099/anc/CH.txt`.

An *incomplete cube* Q for (C, D) is a cube of C such that $\mathrm{supp}(Q)$ is a missed simplex. For any incomplete cube Q with $\sigma = \mathrm{supp}(Q)$, $C|\sigma$ and $D|\sigma$ are maximum classes of dimensions d and $d-1$, respectively. Since $|\sigma| = d$, we have $|C|\sigma| = \binom{d}{\leq d} = \binom{d}{\leq d-1} + 1 = |D|\sigma| + 1$. Since $Q|\sigma = C|\sigma$, there exists a unique concept $c \in Q$ such that $c|\sigma \notin D|\sigma$. We denote c by $s(Q)$, and call c the *source* of Q. In fact, the source map is a bijection between missed simplices for (C, D) and concepts of $C \setminus D$:

▶ **Proposition 11.** *Each $c \in C \setminus D$ is the source of a unique incomplete cube. Moreover, if $r' : D \to X(D)$ is a representation map for D and $r : C \to X(C)$ extends r' by setting $r(c) = \mathrm{supp}(s^{-1}(c))$ for each $c \in C \setminus D$, then r is a representation map for C.*

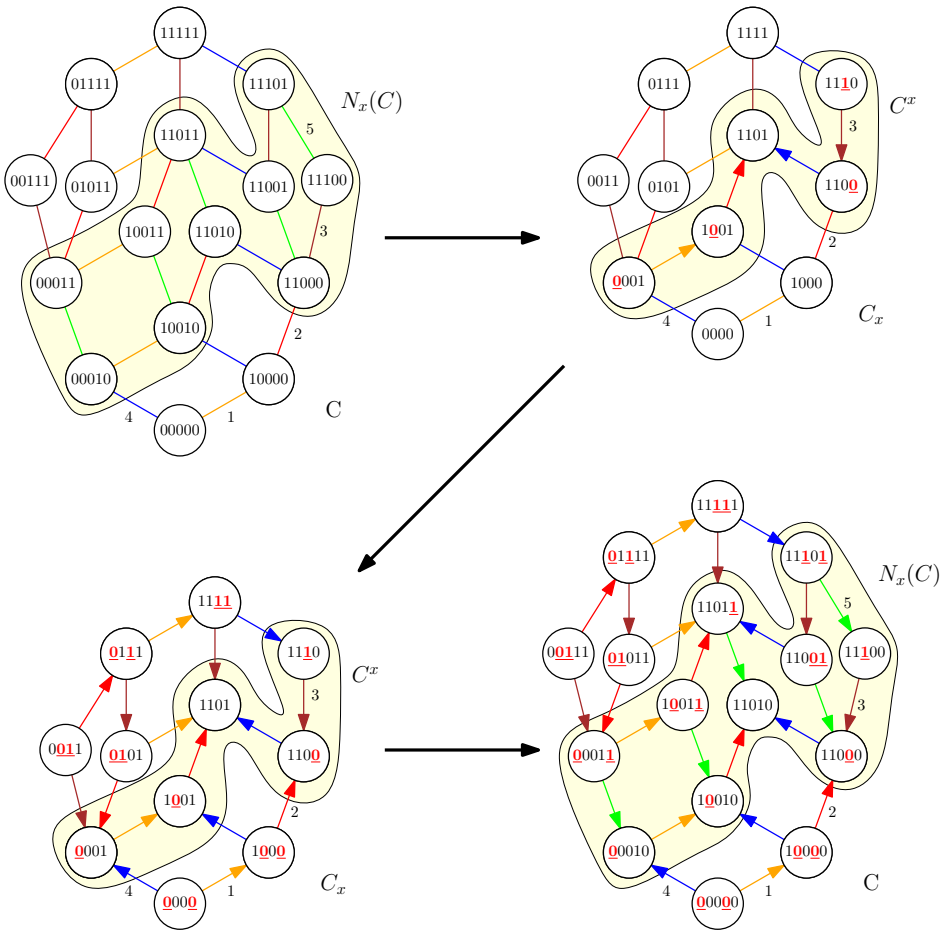


Figure 1 Illustrating the proof of Theorem 10 (when $x = 5$): to construct a representation map for C, we inductively construct a representation map r^x for C^x, extend it to a representation map r_x for C_x using Proposition 11 with $D = C^x$, and finally extend it to a representation map r for C. The representation maps r^x, r_x, and r are defined by the orientation as in Theorem 15 and by the coordinates of the underlined bits.

Proof of Theorem 10. Following the general idea of [14], we derive a representation map for C by induction on $|U|$. For the induction step (see Fig. 1), pick $x \in U$ and consider the maximum classes C_x and $C^x \subset C_x$ with domain $U \setminus \{x\}$. By induction, C^x has a representation map r^x. Use Proposition 11 to extend r^x to a representation map r_x of C_x. Define a map r on C as follows:

- $r(c) = r_x(c_x)$ if $c_x \notin C^x$ or $x \notin c$,
- $r(c) = r_x(c_x) \cup \{x\}$ if $c_x \in C^x$ and $x \in c$.

It is easy to verify that r is non-clashing: indeed, if $c' \neq c'' \in C$ satisfy $c'_x \neq c''_x$ then $c'_x | r_x(c'_x) \cup r_x(c''_x) \neq c''_x | r_x(c'_x) \cup r_x(c''_x)$. Since $r_x(c'_x) \subseteq r(c'), r_x(c''_x) \subseteq r(c'')$, it follows that also c', c'' disagree on $r(c') \cup r(c'')$. Else, $c'_x = c''_x \in C^x$ and $c'(x) \neq c''(x)$. In this case, $x \in r(c') \cup r(c'')$ and therefore c', c'' disagree on $r(c') \cup r(c'')$.

It remains to show that r is a bijection between C and $X(C) = \binom{U}{\leq d}$. It is easy to verify that r is injective. So, it remains to show that $|r(c)| \leq d$, for every $c \in C$. This is clear when $c_x \notin C^x$ or $x \notin c$. If $c_x \in C^x$ and $x \in c$, then $r(c) = r^x(c_x) \cup \{x\}$ and $|r^x(c_x)| \leq d-1$ (since C^x is $(d-1)$-dimensional). Hence, $|r(c)| \leq d$ as required, concluding the proof. ◀

Proof of Proposition 11. Call a maximal cube of C a *chamber* and a facet of a chamber a *panel* (a σ'-*panel* if its support is σ'). Any σ'-panel in C satisfies $|\sigma'| = d-1$ and $\sigma' \in X(D)$. Recall that a gallery between two parallel cubes Q', Q'' (say, two σ'-cubes) is any simple path of σ'-cubes $(Q_0 := Q', Q_1, \ldots, Q_k := Q'')$, where $Q_i \cup Q_{i+1}$ is a d-cube. By Theorem 2(3), any two parallel cubes of C are connected by a gallery in C. Since D is a maximum class, any panel of C is parallel to a panel that is a maximal cube of D. Also for any maximal simplex $\sigma' \in X(D)$, the class $C^{\sigma'}$ is a maximum class of dimension 1 and $D^{\sigma'}$ is a maximum class of dimension 0 (single concept). Thus $C^{\sigma'}$ is a tree (e.g. [11, Lemma 7]) which contains the unique concept $c \in D^{\sigma'}$. We call c *the root* of $C^{\sigma'}$ and denote the σ'-panel P such that $P^{\sigma'} = c$ by $P(\sigma')$.

▷ Claim 12. Let Q be an incomplete cube for (C, D) with source s and support σ, and let $x, y \in U$ such that $x \notin \sigma$ and $y \in \sigma$. Then, the following holds:

(i) Q_x is an incomplete cube for (C_x, D_x) whose source is s_x.

(ii) Q^y is an incomplete cube for (C^y, D^y) whose source is s^y.

Proof. Item (i): C_x and D_x are maximum classes on $U \setminus \{x\}$ of VC-dimensions d and $d-1$, and $\mathrm{supp}(Q_x) = \sigma$. Therefore, Q_x is an incomplete cube for (C^x, D^x). By definition, s is the unique concept $c \in Q$ such that $c|\sigma \notin D|\sigma$. Since $x \notin \sigma$, $D|\sigma = D_x|\sigma$ and s_x is the unique concept c of Q_x so that $c|\sigma \notin D_x|\sigma$, i.e., s_x is the source of Q_x.

Item (ii): C^y and D^y are maximum classes on $U \setminus \{y\}$ of VC-dimensions $d-1$ and $d-2$. Since $y \in \mathrm{supp}(Q)$, $\dim(Q^y) = d-1$ and Q^y is an incomplete cube for (C^y, D^y). Let $\sigma' = \sigma \setminus \{y\}$. It remains to show that $s^y|\sigma' \notin D^y|\sigma'$. Indeed, otherwise both extensions of s^y in σ, namely $s, s\Delta\{y\}$, are in $D|\sigma$ which contradicts that $s = s(Q)$. ◁

Next we prove that each concept of $C \setminus D$ is the source of a unique incomplete cube. Assume the contrary and let (C, D) be a counterexample minimizing the size of U. First, if a concept $c \in C \setminus D$ is the source of two incomplete cubes Q_1, Q_2, then $\mathrm{dom}(C) = \mathrm{supp}(Q_1) \cup \mathrm{supp}(Q_2)$. Indeed, let $\sigma_1 = \mathrm{supp}(Q_1)$ and $\sigma_2 = \mathrm{supp}(Q_2)$. By Claim 12(i) and minimality of (C, D), $\mathrm{dom}(C) = \sigma_1 \cup \sigma_2$. By Claim 12(ii) and minimality of (C, D), $\sigma_1 \cap \sigma_2 = \varnothing$. Indeed, if there exists x in $\sigma_1 \cap \sigma_2$, c^x is the source of the incomplete cubes Q_1^x and Q_2^x for (C^x, D^x), contrary to minimality of (C, D).

Next we assert that any $c \in C \setminus D$ is the source of at most 2 incomplete cubes. Indeed, let c be the source of incomplete cubes Q_1, Q_2, Q_3. Then $\mathrm{dom}(C) = \mathrm{supp}(Q_1) \uplus \mathrm{supp}(Q_2)$, i.e., $\mathrm{supp}(Q_2) = \mathrm{dom}(C) \setminus \mathrm{supp}(Q_1)$. For similar reasons, $\mathrm{supp}(Q_3) = \mathrm{dom}(C) \setminus \mathrm{supp}(Q_1) = \mathrm{supp}(Q_2)$. Thus, by Corollary 3, $Q_2 = Q_3$.

▷ Claim 13. Let $c', c'' \in C \setminus D$ be neighbors and let $c'\Delta c'' = \{x\}$. Then, c' is the source of 2 incomplete cubes if and only if c'' is the source of 0 incomplete cubes. Consequently, every connected component in $G(C \setminus D)$ either contains only concepts c with $|s^{-1}(c)| \in \{0, 2\}$, or only concepts c with $|s^{-1}(c)| = 1$.

Proof. By minimality of (C, D), $(c')^x = (c'')^x$ is the source of a unique incomplete cube for (C^x, D^x) and $c'_x = c''_x$ is the source of a unique incomplete cube for (C_x, D_x). Let Q_1 be the incomplete cube for (C, D) such that $(c')^x$ is the source of Q_1^x. Let Q_2 be the incomplete cube for (C, D) such that c'_x is the source of $(Q_2)_x$. By Claim 12, items (i) and (ii), both $s(Q_1), s(Q_2)$ are in $\{c', c''\}$. Consequently, c' is the source of 2 incomplete cubes (Q_1 and Q_2) if and only if c'' is the source of 0 incomplete cubes. ◁

Pick $c \in C \setminus D$ that is the source of two incomplete cubes for (C, D) and an incomplete cube Q such that $c = s(Q)$. Let $\sigma = \mathrm{supp}(Q)$, $x \in \sigma$, and $\sigma' = \sigma \setminus \{x\}$. The concept c belongs to a unique σ'-panel P. Let $L = (P_0 = P(\sigma'), P_1, \ldots, P_{m-1}, P_m = P)$ be the unique gallery between the root $P(\sigma')$ of the tree C^σ and P. For $i = 1, \ldots, m$, denote the chamber $P_{i-1} \cup P_i$ by Q_i. Since $P_i \cap D$ and $Q_i \cap D$ are ample for $i \geq 0$, and P_i is not contained in D for $i > 0$, it follows that the complements $P_i \setminus D$ and $Q_i \setminus D$ are nonempty ample classes. Hence $P_i \setminus D$ and $Q_i \setminus D$ induce nonempty connected subgraphs of $G(C \setminus D)$. Therefore, it follows that c and each concept $c' \in Q_i \setminus D$ are connected in $G(C \setminus D)$ by a path for $i > 0$, and by Claim 13 it follows that

For every $i > 0$, each $c' \in Q_i \setminus D$ is the source of either 0 or 2 incomplete cubes. (5.1)

Consider the chamber $Q_1 = P_0 \cup P_1$ and its source $s = s(Q_1)$. By the definition of the source, necessarily $s \in P_1$ and $s \notin D$. Therefore, Equation (5.1) implies that there must exist another cube Q' such that $s = s(Q')$. Let s' be the neighbor of s in $P_0 = P(\sigma')$; note that $s' \in D$. Since $\mathrm{supp}(Q_1) \cap \mathrm{supp}(Q') = \varnothing$, it follows that $s|\mathrm{supp}(Q') = s'|\mathrm{supp}(Q') \in D|\mathrm{supp}(Q')$, contradicting that $s = s(Q')$. This establishes the first assertion of Proposition 11.

We prove now that the map r defined in Proposition 11 is a representation map for C. It is easy to verify that it is a bijection between C and $X(C)$, so it remain to establish the non-clashing property: $c|(r(c) \cup r(c')) \neq c'|(r(c) \cup r(c'))$ for all distinct pairs $c, c' \in C$. This holds when $c, c' \in D$ because r' is a representation map. Next, if $c \in C \setminus D$ and $c' \in D$, this holds because $c|r(c) \notin D|r(c)$ by the properties of s. Thus, it remains to show that every distinct $c, c' \in C \setminus D$ satisfy $c|(\mathrm{supp}(Q) \cup \mathrm{supp}(Q')) \neq c'|(\mathrm{supp}(Q) \cup \mathrm{supp}(Q'))$, where $Q = s^{-1}(c), Q' = s^{-1}(c')$. Assume towards contradiction that this does not hold and consider a counterexample with minimal domain size $|U|$. By minimality, $\mathrm{supp}(Q') \cup \mathrm{supp}(Q) = U$ (or else (C_x, D_x), for some $x \notin \mathrm{supp}(Q') \cup \mathrm{supp}(Q)$ would be a smaller counterexample). Therefore, since c, c' are distinct, there must be $x \in U = \mathrm{supp}(Q') \cup \mathrm{supp}(Q)$ such that $c(x) \neq c'(x)$, which is a contradiction. This ends the proof of Proposition 11. ◀

6 Representation Maps for Ample Classes

In this section, we provide combinatorial and geometric characterizations of representation maps of ample classes (which lead to optimal unlabeled compression schemes).

▶ **Theorem 14.** *For an ample class C and a map $r : C \to X(C)$,* (i)-(iii) *are equivalent:*

- **(i)** *r is a representation map;*
- **(ii)** *$c'|r(c')\Delta r(c'') \neq c''|r(c')\Delta r(c'')$ for all $c', c'' \in C, c' \neq c''$;*
- **(iii)** *r is a bijection and for every realizable sample s of C, there is a unique $c \in C$ that is consistent with s and $r(c) \subseteq \mathrm{dom}(s)$.*

This theorem implies that for any representation map $r : C \to X(C)$ and any x-edge cc', $r(c)\Delta r(c') = \{x\}$. Hence, r defines an orientation o_r of $G(C)$: an x-edge cc' is oriented from c to c' iff $x \in r(c) \setminus r(c')$. Moreover, as a corollary of Theorem 14, we can show that:

- **(C1)** for any $c \in C$, all outgoing neighbors of c belong to a cube of C;
- **(C2)** o_r is a USO on each cube of C.

An orientation o of the edges of $G(C)$ is a *unique sink orientation (USO)* if o satisfies (C1) and (C2). The *out-map* r_o of an orientation o associates to each $c \in C$ the coordinate set of the edges outgoing from c. We continue with a characterization of representation maps of ample classes as out-maps of USOs, extending a similar result of Szabó and Welzl [31] for cubes. This characterization is "local-to-global", since (C1) and (C2) are conditions on the *stars* $\mathrm{St}(c)$ of all concepts $c \in C$ ($\mathrm{St}(c)$ is the set of all faces of the cubes containing c).

▶ **Theorem 15.** *For an ample class C and a map $r : C \to 2^U$,* (i)-(iii) *are equivalent:*
- **(i)** *r is a representation map;*
- **(ii)** *r is the out-map of a USO;*
- **(iii)** *$r(c) \in X(C)$ for any $c \in C$ and o_r satisfies (C2).*

Proof. The implication (i)⇒(ii) is a consequence of Theorem 14. Now, we prove (ii)⇒(i). Clearly, property (C1) implies that $r(c) \in X(C)$ for any $c \in C$, whence r is a map from C to $X(C)$. Let C be an ample class of smallest size admitting a non-representation map $r : C \to X(C)$ satisfying (C1) and (C2). Hence there exist $u_0, v_0 \in C$ such that $u_0|(r(u_0)\Delta r(v_0)) = v_0|(r(u_0)\Delta r(v_0))$, i.e., $(u_0\Delta v_0) \cap (r(u_0)\Delta r(v_0)) = \varnothing$; (u_0, v_0) is called a *clashing pair*.

▷ Claim 16. If (u_0, v_0) is a clashing pair, then $C = C \cap B(u_0, v_0)$ and $r(u_0) = r(v_0) = \varnothing$.

Proof. Since $C \cap B(u_0, v_0)$ is ample and $(u_0\Delta v_0) \cap (r(u_0)\Delta r(v_0)) = \varnothing$, (u_0, v_0) is a clashing pair for $C \cap B(u_0, v_0)$ and the restriction r_B of r to $\mathrm{supp}(B(u_0, v_0))$. Since r_B and $C \cap B(u_0, v_0)$ satisfy (C1) and (C2), by minimality of C, $C = C \cap B(u_0, v_0)$. Moreover, if $r(u_0) \neq r(v_0)$, then there is $x \in r(u_0)\Delta r(v_0)$ and $x \in u_0\Delta v_0$, contradicting that (u_0, v_0) is a clashing pair.

Suppose $r(u_0) \neq \varnothing$ and pick $x \in r(u_0) = r(v_0)$. Consider the carrier $N_x(C)$ of C^x. Note that $r(u_0) \subseteq \mathrm{supp}(N_x(C))$. Indeed, let $y \in r(u_0)$. By (C1), u_0 belongs to an $\{x, y\}$-square of C, whence $y \in \mathrm{supp}(N_x(C))$. Analogously, $r(v_0) \subseteq \mathrm{supp}(N_x(C))$, thus (u_0, v_0) is a clashing pair for $N_x(C)$ and the restriction of r to $N_x(C)$. $N_x(C)$ is ample as the product of C^x by an x-edge. By minimality of C, $C = N_x(C)$. Define $r^x : C^x \mapsto X(C^x)$ by
- $r^x(c) = r(c) \setminus \{x\}$ if $x \in r(c)$,
- $r^x(c) = r(cx) \setminus \{x\}$ otherwise.

Consequently, for an x-edge of C between c and cx, $r^x(c)$ is the label of the origin of this edge minus x; we call r^x the *x-out-map* of r. We assert that r^x satisfies (C1) and (C2). Condition (C1) is trivial because it holds for cubes of C. To establish condition (C2), suppose that there exists a cube B' of C^x and $u', v' \in B'$ such that $r^x(u') \cap \mathrm{supp}(B') = r^x(v') \cap \mathrm{supp}(B')$. The cube $B := B' \times \{x\}$ is included in C since B' is a cube of C^x. Then among the four pairs $(u', v'), (u', v'x), (u'x, v'), (u'x, v'x)$ of B one can select a pair (u, v) such that $r(u) = r^x(u') \cup \{x\} = r^x(v') \cup \{x\} = r(v)$, a contradiction with condition (C2) for C and r. This shows that r^x satisfies (C1) and (C2). Recall that $x \in r(u_0) = r(v_0)$, suppose wlog that $x \in v_0 \setminus u_0$, and let $u_0' = u_0$ and $v_0' = v_0 \setminus \{x\}$. Then $r^x(u_0') = r(u_0) \setminus \{x\} = r(v_0) \setminus \{x\} = r^x(v_0')$, and consequently (u_0', v_0') is a clashing pair for the restriction of r on $C^x \cap B(u_0', v_0')$. Since $C^x \cap B(u_0', v_0')$ is ample and smaller than C, this contradicts the minimality of C. ◁

▷ Claim 17. C is a cube minus a vertex.

Proof. By (C2), C is not a cube. If C is not a cube minus a vertex, since the complement $C^* = 2^U \setminus C$ is also ample (thus $G(C^*)$ is connected), $G(C^*)$ contains an x-edge ww' with $x \notin w$ and $x \in w'$. Consider C_x and define the map $r_x : C_x \mapsto X(C_x)$ by
- $r_x(c) = r(c)$ if $c \in C$ and $x \notin r(c)$,
- $r_x(c) = r(cx)$ otherwise.

Hence $r_x(c \setminus \{x\}) = r(c)$ for each $c \in C$ with $x \notin r(c)$. We call r_x the *x-in-map* of r; r_x satisfies (C1), because r satisfies (C1). Suppose that r_x violates (C2). Then there exists a cube B' of C_x and $u', v' \in B'$ such that $(u'\Delta v') \cap (r_x(u')\Delta r_x(v')) = \varnothing$. Let $u \in \{u', u'x\}$ such that $r(u) = r(u')$ and let $v \in \{v', v'x\}$ such that $r(v) = r(v')$. The restriction C'_x of the ample class $C' := C \cap B(u, v)$ is the cube B'. Since $w, w' \notin C$ and ww' is an x-edge, $w \notin C'_x$. Thus there exists $y \in \mathrm{supp}(C)$ such that C' and the edge ww' of C^* belong to different y-half-spaces $C^- = \{c \in C : y \notin c\}$ and $C^+ = \{c \in C : y \in c\}$ of the cube 2^U. Since $y \in \mathrm{supp}(C)$, the half-space containing ww' also contains a concept of C. Hence, C' is a proper ample subset of C. Since $u \subseteq u' \cup \{x\}$, $v \subseteq v' \cup \{x\}$, $x \notin r(u) = r_x(u')$, $x \notin r(v) = r_x(v')$, we deduce that $u \cap (r(u)\Delta r(v)) = u' \cap (r_x(u')\Delta r_x(v'))$ and $v' \cap (r_x(u')\Delta r_x(v')) = v \cap (r(u)\Delta r(v))$. Since $(u'\Delta v') \cap (r_x(u')\Delta r_x(v')) = \varnothing$, (u, v) is a clashing pair for the restriction of r on C', contrary to the minimality of C. Hence by minimality of C, r_x is a representation map for C_x.

Consider a clashing pair (u_0, v_0) for C and r, and let $u'_0 = u_0 \setminus \{x\}$ and $v'_0 = v_0 \setminus \{x\}$. Observe that $r_x(u'_0) = r(u_0) = r(v_0) = r_x(v'_0) = \varnothing$. Since r_x is a representation map for C_x, necessarily $u'_0 = v'_0$. Consequently, $u_0\Delta v_0 = \{x\}$, i.e., u_0v_0 is an x-edge of $G(C)$. This is impossible since C satisfies (C2). Therefore, C is necessarily a cube minus a vertex. ◁

Now, we complete the proof of the implication (ii)⇒(i). By Claim 16, $r(u_0) = r(v_0) = \varnothing$. By condition (C1), $r(c) \neq U$ for any $c \in C$. Thus there exists a set $s \in X(C) = 2^U \setminus \{U, \varnothing\}$ such that $s \neq r(c)$ for any $c \in C$. Every s-cube B of C contains a source $p(B)$ for o_{r_B} (i.e., $s \subseteq r(p(B))$). For each s-cube B of C, let $t(B) = r(p(B)) \setminus s$. Notice that $\varnothing \subsetneq t(B) \subsetneq U \setminus s$ since $s \subsetneq r(p(B)) \subsetneq U$. Consequently, there are $2^{|U|-|s|} - 2$ choices for $t(B)$ and since C is a cube minus one vertex by Claim 17, there are $2^{|U|-|s|} - 1$ s-cubes in C. Consequently, there exist two s-cubes B, B' such that $t(B) = t(B')$. Thus $\varnothing \subsetneq s \subsetneq r(p(B)) = r(p(B'))$ and $(p(B), p(B'))$ is a clashing pair for C and r, contradicting Claim 16.

The implication (ii)⇒(iii) is trivial. To prove (iii)⇒(ii), we show by induction on $|U|$ that a map $r : C \to X(C)$ satisfying (C2) also satisfies (C1). For any $x \in U$, let r^x denote the x-out-map defined in Claim 16. Recall that if cc' is an x-edge directed from c to c', then $x \in r(c)$ and r^x maps $c^x = (c')^x \in C^x$ to $r(c) \setminus \{x\} \in X(C^x)$. Thus r^x maps C^x to $X(C^x)$. Moreover, each cube B^x of C^x is contained in a unique cube B of C such that $\mathrm{supp}(B) = \mathrm{supp}(B^x) \cup \{x\}$. If there exist $c^x_1, c^x_2 \in B^x$ such that $r^x(c^x_1) = r(c^x_2)$, then there exist $c_1, c_2 \in B$ such that $r(c_1) = r^x(c^x_1) \cup \{x\} = r^x(c^x_2) \cup \{x\} = r(c_2)$, contradicting (C2). Consequently, o_{r_x} satisfies (C2). By induction hypothesis, o_{r_x} satisfies (C1) for any $x \in U$.

For any concept $c \in C$, pick $x \in r(c)$. Since r^x satisfies (C1), c^x belongs to a σ'-cube in C^x with $\sigma' = r^x(c^x) = r(c) \setminus \{x\}$. This implies that c belongs to a σ-cube in C with $\sigma = \sigma' \cup \{x\} = r(c)$. Thus o_r satisfies (C1), concluding the proof of Theorem 15. ◀

We conclude with some remarks regarding Theorems 14 and 15. First, corner peelings correspond *exactly* to acyclic USOs. Second, given a representation map for C one can derive representations maps for intersections of C with cubes, reductions C^Y, and restrictions C_Y. Third, there exist a bijection $r' : C \to X(C)$ satisfying (C1) and an injection $r'' : C \to 2^U$ satisfying (C2). Nevertheless, we were not able to find a map satisfying (C1) and (C2). It is surprising that, while each d-cube has at least $d^{\Omega(2^d)}$ USOs [17], it is so difficult to find a single USO for ample classes. One can try to find such maps by extending the approach for maximum classes: given ample classes C and D with $D \subset C$, a representation map r for C is called *D-entering* if all edges cd with $c \in C \setminus D$ and $d \in D$ are directed by o_r from c to d. The representation map defined in Proposition 11 is D-entering. Given $x \in \mathrm{dom}(C)$, suppose that r_x is a C^x-entering representation map for C_x. We can extend the orientation o_{r_x} to an orientation o of $G(C)$ as follows. Each x-edge cc' of $G(C)$ is directed arbitrarily, while each

other edge cc' is directed as the edge $c_x c'_x$ is directed by o_{r_x}. Since o_{r_x} satisfies (C1), (C2) and r_x is C^x-entering, o also satisfies (C1), (C2), thus the map r_o is a representation map for C. So, ample classes would admit representation maps, if *for any ample classes $D \subseteq C$, any representation map r' of D extends to a D-entering representation map r of C.*

References

1 Richard P. Anstee, Lajos Rónyai, and Attila Sali. Shattering News. *Graphs Combin.*, 18(1):59–73, 2002. doi:10.1007/s003730200003.

2 Hans-Jürgen Bandelt and Victor Chepoi. Metric Graph Theory and Geometry: a Survey. In J. E. Goodman, J. Pach, and R. Pollack, editors, *Surveys on Discrete and Computational Geometry: Twenty Years Later*, volume 453 of *Contemp. Math.*, pages 49–86. Amer. Math. Soc., Providence, RI, 2008. doi:10.1090/conm/453/08795.

3 Hans-Jürgen Bandelt, Victor Chepoi, Andreas W. M. Dress, and Jack H. Koolen. Combinatorics of Lopsided Sets. *European J. Combin.*, 27(5):669–689, 2006. doi:10.1016/j.ejc.2005.03.001.

4 Béla Bollobás and Andrew J. Radcliffe. Defect Sauer Results. *J. Combin. Theory Ser. A*, 72(2):189–208, 1995. doi:10.1016/0097-3165(95)90060-8.

5 Jérémie Chalopin, Victor Chepoi, Shay Moran, and Manfred K. Warmuth. Unlabeled Sample Compression Schemes and Corner Peelings for Ample and Maximum Classes. *arXiv preprint*, 1812.02099, 2018. arXiv:1812.02099.

6 Thorsten Doliwa, Gaojian Fan, Hans Ulrich Simon, and Sandra Zilles. Recursive Teaching Dimension, VC-dimension and Sample Compression. *J. Mach. Learn. Res.*, 15(1):3107–3131, 2014. URL: http://jmlr.org/papers/v15/doliwa14a.html.

7 Andreas W. M. Dress. Towards a Theory of Holistic Clustering. In *Mathematical Hierarchies and Biology*, volume 37 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 271–290. DIMACS, Amer. Math. Soc., 1996. doi:10.1090/dimacs/037/19.

8 Paul H. Edelman and Robert E. Jamison. The Theory of Convex Geometries. *Geom. Dedicata*, 19(3):247–270, 1985. doi:10.1007/BF00149365.

9 Sally Floyd. *On Space Bounded Learning and the Vapnik-Chervonenkis Dimension*. PhD thesis, International Computer Science Institut, Berkeley, CA, 1989. URL: https://www.icsi.berkeley.edu/icsi/node/2289.

10 Sally Floyd and Manfred K. Warmuth. Sample Compression, Learnability, and the Vapnik-Chervonenkis Dimension. *Mach. Learn.*, 21(3):269–304, 1995. doi:10.1007/BF00993593.

11 Bernd Gärtner and Emo Welzl. Vapnik-Chervonenkis dimension and (pseudo-)hyperplane arrangements. *Discrete Comput. Geom.*, 12(4):399–432, 1994. doi:10.1007/BF02574389.

12 Mikhaïl Gromov. Hyperbolic Groups. In S. M. Gersten, editor, *Essays in Group Theory*, volume 8 of *Math. Sci. Res. Inst. Publ.*, pages 75–263. Springer, New York, 1987. doi:10.1007/978-1-4613-9586-7_3.

13 H. Tracy Hall. *Counterexamples in Discrete Geometry*. PhD thesis, University of California, 2004.

14 Dima Kuzmin and Manfred K. Warmuth. Unlabeled Compression Schemes for Maximum Classes. *J. Mach. Learn. Res.*, 8:2047–2081, 2007. URL: http://www.jmlr.org/papers/v8/kuzmin07a.html.

15 James F. Lawrence. Lopsided Sets and Orthant-intersection of Convex Sets. *Pacific J. Math.*, 104(1):155–173, 1983. doi:10.2140/pjm.1983.104.155.

16 Nick Littlestone and Manfred K. Warmuth. Relating Data Compression and Learnability. Technical report, Department of Computer and Information Sciences, Santa Cruz, CA, 1986.

17 Jiří Matoušek. The Number Of Unique-Sink Orientations of the Hypercube. *Combinatorica*, 26(1):91–99, 2006. doi:10.1007/s00493-006-0007-0.

18 Tamás Mészáros and Lajos Rónyai. Shattering-Extremal Set Systems of VC Dimension at most 2. *Electron. J. Combin.*, 21(4):P4.30, 2014. URL: http://www.combinatorics.org/ojs/index.php/eljc/article/view/v21i4p30.

19 Shay Moran. Shattering-extremal Systems. *arXiv preprint*, 1211.2980, 2012. arXiv:1211.2980.

20 Shay Moran and Manfred K. Warmuth. Labeled Compression Schemes for Extremal Classes. In *ALT 2016*, volume 9925 of *Lecture Notes in Comput. Sci.*, pages 34–49. Springer, 2016. doi:10.1007/978-3-319-46379-7_3.

21 Shay Moran and Amir Yehudayoff. Sample Compression Schemes for VC Classes. *J. ACM*, 63(3):21:1–21:10, 2016. doi:10.1145/2890490.

22 Mogens Nielsen, Gordon D. Plotkin, and Glynn Winskel. Petri nets, event structures and domains, part I. *Theoret. Comput. Sci.*, 13(1):85–108, 1981. doi:10.1016/0304-3975(81)90112-2.

23 Alain Pajor. *Sous-espaces ℓ_1^n des Espaces de Banach*. Travaux en Cours. Hermann, Paris, 1985.

24 Dömötör Pálvölgyi and Gábor Tardos. Unlabeled Compression Schemes Exceeding the VC-dimension. *arXiv preprint*, 1811.12471, 2018. arXiv:1811.12471.

25 Benjamin I. P. Rubinstein and J. Hyam Rubinstein. A Geometric Approach to Sample Compression. *J. Mach. Learn. Res.*, 13:1221–1261, 2012. URL: http://www.jmlr.org/papers/v13/rubinstein12a.html.

26 Michah Sageev. CAT(0) cube complexes and groups. In M. Bestvina, M. Sageev, and K. Vogtmann, editors, *Geometric Group Theory*, volume 21 of *IAS/Park City Math. Ser.*, pages 6–53. Amer. Math. Soc., Inst. Adv. Study, 2012. doi:10.1090/pcms/021/02.

27 Rahim Samei, Boting Yang, and Sandra Zilles. Generalizing Labeled and Unlabeled Sample Compression to Multi-label Concept Classes. In *ALT 2014*, volume 8776 of *Lecture Notes in Comput. Sci.*, pages 275–290. Springer, 2014. doi:10.1007/978-3-319-11662-4_20.

28 Norbert Sauer. On the Density of Families of Sets. *J. Combin. Theory Ser. A*, 13(1):145–147, 1972. doi:10.1016/0097-3165(72)90019-2.

29 Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge Univ. Press, 2014. doi:10.1017/CBO9781107298019.

30 Saharon Shelah. A Combinatorial Problem, Stability and Order for Models and Theories in Infinitary Languages. *Pacific J. Math.*, 41(1):247–261, 1972. doi:10.2140/pjm.1972.41.247.

31 Tibor Szabó and Emo Welzl. Unique Sink Orientations of Cubes. In *FOCS 2001*, pages 547–555. IEEE Computer Society, 2001. doi:10.1109/SFCS.2001.959931.

32 Vladimir N. Vapnik and Alexey Y. Chervonenkis. On the Uniform Convergence of Relative Frequencies of Events to their Probabilities. *Theory Probab. Appl.*, 16(2):264–280, 1971. doi:10.1137/1116025.

33 Manfred K. Warmuth. Compressing to VC Dimension Many Points. In *COLT/Kernel 2003*, volume 2777 of *Lecture Notes in Comput. Sci.*, pages 743–744. Springer, 2003. doi:10.1007/978-3-540-45167-9_60.

34 Emo Welzl. Complete Range Spaces, 1987. Unpublished notes.

35 Douglas H. Wiedemann. *Hamming Geometry*. PhD thesis, University of Waterloo, 1986. re-typeset July, 2006.

36 Avi Wigderson. *Mathematics and Computation*. Princeton Univ. Press, 2019.

37 Glynn Winskel. *Events in Computation*. PhD thesis, Edinburgh University, 1980.

38 Günter M. Ziegler. *Lectures on Polytopes*, volume 152 of *Grad. Texts in Math.* Springer–Verlag, New York, 1995. doi:10.1007/978-1-4613-8431-1.

Query-To-Communication Lifting for BPP Using Inner Product

Arkadev Chattopadhyay
School of Technology and Computer Science, Tata Institute of Fundamental Research, Mumbai, India
http://www.tcs.tifr.res.in/~arkadev/
arkadev@tifr.res.in

Yuval Filmus
Department of Computer Science, Technion Israel Institute of Technology, Haifa, Israel
https://filmus.net.technion.ac.il/
yuvalfi@cs.technion.ac.il

Sajin Koroth
Department of Computer Science, University of Haifa, Haifa, Israel
https://sites.google.com/csweb.haifa.ac.il/sajin
sajin@csweb.haifa.ac.il

Or Meir
Department of Computer Science, University of Haifa, Haifa, Israel
http://cs.haifa.ac.il/~ormeir/
ormeir@cs.haifa.ac.il

Toniann Pitassi
Department of Computer Science, University of Toronto, Canada
https://www.cs.toronto.edu/~toni/
toni@cs.toronto.edu

Abstract

We prove a new query-to-communication lifting for randomized protocols, with inner product as gadget. This allows us to use a much smaller gadget, leading to a more efficient lifting. Prior to this work, such a theorem was known only for deterministic protocols, due to Chattopadhyay et al. [4] and Wu et al. [22]. The only query-to-communication lifting result for randomized protocols, due to Göös, Pitassi and Watson [13], used the much larger indexing gadget.

Our proof also provides a unified treatment of randomized and deterministic lifting. Most existing proofs of deterministic lifting theorems use a measure of information known as *thickness*. In contrast, Göös, Pitassi and Watson [13] used blockwise min-entropy as a measure of information. Our proof uses the blockwise min-entropy framework to prove lifting theorems in both settings in a unified way.

2012 ACM Subject Classification Theory of computation → Communication complexity; Theory of computation → Oracles and decision trees

Keywords and phrases lifting theorems, inner product, BPP Lifting, Deterministic Lifting

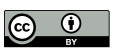
Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.35

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at [3], https://arxiv.org/abs/1904.13056.

Funding *Yuval Filmus*: Taub Fellow – supported by the Taub Foundations. The research was funded by ISF grant 1337/16.
Sajin Koroth: Supported by the Israel Science Foundation (grant No. 1445/16)
Or Meir: Partially supported by ISF grant by the Israel Science Foundation (grant No. 1445/16).

Acknowledgements We thank Daniel Kane for some very enlightening conversations and suggestions. This work was done (in part) while the authors were visiting the Simons Institute for the Theory of Computing.



46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 35; pp. 35:1–35:15

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

1 Introduction

In this work, we prove new lifting theorems that use the inner-product function as a gadget. Let $f\colon \{0,1\}^n \to \{0,1\}^m$ and $g\colon \{0,1\}^b \times \{0,1\}^b \to \{0,1\}$ be functions (where g is referred to as a *gadget*). The block-composed function $f \circ g^n$ is the function that takes n instances $(x_1, y_1), \ldots, (x_n, y_n)$ of inputs for g and computes $f \circ g^n$ as,

$$f \circ g^n((x_1, y_1), \ldots, (x_n, y_n)) = f(g(x_1, y_1), g(x_2, y_2), \ldots, g(x_n, y_n)).$$

Lifting theorems are theorems that relate the communication complexity of $f \circ g^n$ to the query complexity of f and the communication complexity of g.

More specifically, consider the following communication problem: Alice gets $x_1, \ldots, x_n$, Bob gets $y_1, \ldots, y_n$, and they wish to compute the output of $f \circ g^n$ on their inputs. The natural protocol for doing so is the following: Alice and Bob jointly *simulate* a decision tree of optimal height for solving f. Any time the tree queries the i-th bit, they compute g on the i-th instance by invoking the best possible communication protocol for g. A *lifting theorem* is a theorem that says that this natural protocol is optimal.

Lifting theorems are interesting because they create a connection between query complexity and communication complexity. This connection, besides being interesting in its own right, allows us to transfer lower bounds and separations from the from query complexity (which is a relatively simple model) to a communication complexity (which is a significantly richer model).

In particular, the first result of this form, due to Raz and McKenzie [19], proved a lifting theorem from *deterministic* query complexity to *deterministic* communication complexity when g is the index function. They then used it to prove new lower bounds on communication complexity by lifting query-complexity lower-bounds. More recently, Göös, Pitassi and Watson [12] applied that theorem to separate the logarithm of the partition number and the deterministic communication complexity of a function, resolving a long-standing open problem. This too was done by proving such a separation in the setting of query complexity and lifting it to the setting of communication complexity. This result stimulated a flurry of work on lifting theorems of various kinds, such as: round-preserving lifting theorems with applications to time-space trade-offs for proof complexity [6], deterministic lifting theorems with other gadgets [4, 22], lifting theorems from randomized query complexity to randomized communication complexity [13], lifting theorems for DAG-like protocols [8] with applications to monotone circuit lower bounds, lifting theorems for asymmetric communication problems [5] with applications to data-structures, and a lifting theorem [18] for the EQUALITY gadget. There are also lifting theorems which lifts more analytic properties of the function like approximate degree due to Sherstov [20] and independently due to Shi and Zhu [21], that enabled several important later developments. Although such lifting theorems lift analytical properties of functions, several later works [11] showed how analytical arguments can be made to work for lifting relations.

Viewed from another angle, lifting theorems are natural generalizations of classic theorems such as direct-sum theorems and XOR lemmas [23, 15, 7, 16, 1, 2]: in particular, if we set f to be the identity function or the parity function, we get a direct sum theorem or an XOR lemma for g, respectively. This point of view motivates the work of Hatami et al. [14] that made progress towards proving a lifting theorem with a constant-size gadget.

In almost all known lifting theorems, the function f can be arbitrary (and may also be a general search problem) while g is usually a specific function (e.g., the index function). This raises the following natural question: for which choices of g can we prove lifting theorems?

This question is interesting both because many applications depend on the choice of g, and because if we view lifting theorems as generalizations of direct-sum theorems, we would like them to work for as many choices of g as possible.

In particular, applications of lifting theorems often depend on the size of the gadget, which is the length of the input to g. Both the deterministic lifting theorem of Raz and McKenzie [19] and the randomized lifting theorem of Göös et al. [13] use the indexing function INDEX, which has very large size (polynomial in n). Reducing the gadget size to a constant would have many interesting applications like reproducing tight randomized lower bounds for important functions such as set-disjointness etc. We would like point out that, although we reduce the gadget size to logarithmic in n in this work, it is not enough to obtain the interesting applications a constant sized gadget would have yielded.

In the deterministic setting, the gadget size was recently improved to logarithmic by the independent works of [4] and [22], who chose the gadget g to be the inner product function. Moreover, [4, 17] showed the lifting to work for a large class of gadgets. However, the randomized lifting theorem of Göös et al. [13], until our work, seemed to work only with INDEX as gadget.

In this work, we prove a randomized lifting theorem using an inner product gadget of logarithmic size. This has the immediate application that any lower bound on the outer function f can now be lifted to a much stronger lower bound on the composed function $f \circ g^n$, since hardness is measured as a function of the input length. This allows us, for example, to simplify the lower bounds of Göös, and Jayram [9] on AND-OR trees and MAJORITY trees, since we can now obtain them directly from the randomized query complexity lower bounds rather than going through conical juntas.

We now turn to state our main result more formally. Let $n \in \mathbb{N}$ be such that $n \geq 2$ and let $b \stackrel{\text{def}}{=} 40,000 \cdot \log n$. Let $\Lambda \stackrel{\text{def}}{=} \{0,1\}^b$, and let $g \colon \Lambda \times \Lambda \to \{0,1\}$ denote the inner product (mod 2) gadget. We prove lifting theorems for various lifted versions of $G \stackrel{\text{def}}{=} g^n$. That is, $G \colon \Lambda^n \times \Lambda^n \to \{0,1\}^n$ is the function that takes n independent instances of g and computes g on all of them. Here is our main result:

▶ **Theorem 1** (Randomized lifting). *Let $S \colon \{0,1\}^n \to \Sigma$ be any search problem and let Π be a bounded-error randomized communication protocol that solves $S \circ G$ with complexity c and error probability ε. Then, there exists a randomized decision tree T that solves S with complexity $O(\frac{c}{b})$ and bounded error probability.*

Using essentially the same proof method, we also prove a similar result in the deterministic setting:

▶ **Theorem 2** (Deterministic lifting). *Let S be any search problem that takes inputs from $\{0,1\}^n$, and let Π be a deterministic communication protocol that solves $S \circ G$ with complexity c. Then, there exists a deterministic decision tree T that solves S with complexity $O(\frac{c}{b})$.*

Most existing proofs of deterministic lifting theorems employ an information measure known as *thickness*, borrowed from earlier work on the KRW conjecture. The one deviation from this is the recent beautiful work of Garg et al. [8] who prove a deterministic lifting theorem in the dag-like setting. Curiously, their result does not use the thickness measure of information, but rather uses the blockwise min-entropy measure of information that was used by Göös, Pitassi and Watson [13] in order to prove a randomized lifting theorem. A natural direction of further research is to investigate if these disparate techniques can be unified. Indeed, a related question was asked in the first work to employ the measures of min-entropy for lifting by Göös et al. [10]: they asked if min-entropy and density based techniques could be used to prove (or simplify the existing proof of) Raz–McKenzie style deterministic lifting theorems.

Our unified proof answers this question by showing that the same information measure (blockwise min entropy) can in fact be used in both the deterministic and randomized settings. The main difference between the two proofs is the way in which we decide the next bit of the communication protocol: in the deterministic setting, we make a greedy choice, and in the randomized setting, we make a (non-uniform) random choice. Whereas in the randomized setting, our information measure guarantees that we are able to estimate the distribution of the next bit of the protocol, in the deterministic setting it guarantees *richness*, that is, when the protocol ends, there is some input consistent with answers of all queries made by the decision tree.

Organization of the paper. In Section 2 we set up the machinery that is used in both the deterministic and the randomized lifting theorems. We prove the deterministic lifting theorem in Section 3, and the randomized lifting theorem in Section 4. Both proofs use a Fourier-theoretic lemma, proved in Section 5.

2 Common Machinery

In this paper we consider lifting theorems for the most general case of search problems. A search problem $\mathcal{S}$ is defined by a relation $\mathcal{I} \times \mathcal{O}$ where $\mathcal{I}$ is a finite set of inputs and $\mathcal{O}$ is a finite set of outputs. The goal of the search problem, given an input $x \in \mathcal{I}$ is to find at least one output $o \in \mathcal{O}$ such that $(x, o) \in \mathcal{S}$. Like in the statement of the main theorem, let S be any search problem that takes inputs from $\{0,1\}^n$, and let Π be a bounded-error randomized communication protocol that solves $S \circ G$ with complexity c and error probability ε. We prove the randomized and deterministic lifting theorems, by building deterministic and randomized decision trees of cost $O(c/b)$ based on respective protocols of cost c. Intuitively, in both theorems, on input $z \in \{0,1\}^n$, the tree T will simulate the action of the protocol Π on inputs $(x, y) \in G^{-1}(z)$. More specifically, the tree will simulate the protocol bit by bit, and maintain a rectangle $\mathcal{X} \times \mathcal{Y}$ that is consistent with the protocol so far such that all the strings in $G(\mathcal{X} \times \mathcal{Y})$ are consistent with the queries made so far. To this end, we consider random variables X and Y that are distributed uniformly over $\mathcal{X}$ and $\mathcal{Y}$ respectively. We now state a few useful definitions and results about such random variables

The first such definition ensures that the random variables we consider have enough blockwise min-entropy.

▶ **Definition 3.** *Let X be a random variable taking values in Λ^n. We say that X is* δ-dense *if for every $I \subseteq [n]$ it holds that $H_\infty(X_I) \geq \delta \cdot b \cdot |I|$.*

We would like these random variables to be consistent with the query answers obtained by the decision tree thus far in the simulation. To this end, we also define the following notion of restrictions.

▶ **Definition 4.** *Given a restriction $\rho \in \{0,1,*\}^n$, we denote by* $\mathrm{fix}(\rho)$ *and* $\mathrm{free}(\rho)$ *the set of fixed and free coordinates of ρ respectively.*

Intuitively, $\mathrm{fix}(\rho)$ represents the query answers obtained thus far, and $\mathrm{free}(\rho)$ represents the yet unqueried coordinates. With these definitions, we define the property that we would like to maintain for X and Y during the simulation.

▶ **Definition 5** (following [13])**.** *Let X, Y be random variables taking values in Λ^n, and let $\rho \in \{0,1,*\}^n$ be a restriction. We say that X and Y are ρ-structured if $X_{\mathrm{free}(\rho)}$ and $Y_{\mathrm{free}(\rho)}$ are 0.9-dense, and $g^{\mathrm{fix}(\rho)}\left(X_{\mathrm{fix}(\rho)}, Y_{\mathrm{fix}(\rho)}\right) = \rho_{\mathrm{fix}(\rho)}$.*

In both lifting theorems, the decision tree T starts by setting X and Y to be uniform over Λ^n, and maintains throughout the simulation the invariant that, if ρ is the restriction that represents the current "state of knowledge" regarding the input z, then X and Y are ρ-structured. In order to maintain this invariant, we use the following Fourier-analytic result, which is proved in Section 5.

▶ **Definition 6.** *Let $\alpha \in \Lambda^n$ and let Y be a random variable taking values in Λ^n. We say that α is η-*bad *for Y if there exists a set $I \subset [n]$ and a string $\sigma \in \{0,1\}^I$ such that the random variable*

$$Y_{[n]-I} \left| g^I(\alpha_I, Y_I) = \sigma_I \right.$$

is not η-dense or

$$\Pr\left[g^I(\alpha_I, Y_I) = \sigma_I\right] < 2^{-|I|-1}.$$

▶ **Theorem 7** (Main Technical Tool)**.** *Let $n \in \mathbb{N}$ and let $b \in \mathbb{N}$ such that $b \geq 40000 \cdot \log(n)$. Let X and Y be random variables taking values in Λ^n that are δ_X-dense and δ_Y-dense respectively. Suppose that $\delta_X + \delta_Y \geq 1.3$ and $\delta_Y \geq 0.1$. Then, the probability that X takes a value that is $\frac{\delta_Y}{2.01}$-bad for Y is at most $2^{-0.01 \cdot b}$.*

We also use the following analogue of the "uniform marginals lemma" of [13] for the inner product gadget.

▶ **Lemma 8** (Uniform marginals lemma)**.** *Let X, Y be random variables uniformly distributed over sets $\mathcal{X}, \mathcal{Y} \subseteq \Lambda^n$, and suppose they are ρ-structured. Then, for any $z \in \{0,1\}^n$ that is consistent with ρ, the uniform distribution over $G^{-1}(z) \cap (\mathcal{X} \times \mathcal{Y})$ has its marginal distributions $\frac{1}{n^3}$-close to X and Y respectively.*

In order to prove Lemma 8, we use the following definition and lemma from Göös et al. [10].

▶ **Definition 9.** *Let $\varepsilon > 0$ and let V be a random variable taking values from a set $\mathcal{V}$. We say that V is* ε-pointwise close to uniform *if for every $v \in \mathcal{V}$ it holds that $\Pr[V = v] \in (1 \pm \varepsilon) \cdot \frac{1}{|\mathcal{V}|}$.*

▶ **Lemma 10.** *Let A, B be 0.6-dense random variables taking values from Λ^m. Then $g^m(A, B)$ is $2^{-\frac{b}{20}}$-uniform.*

The proof of this lemma, which is similar to the proof of the uniform marginals lemma in [13], appears in the full version [3] of the paper.

We use the following simple folklore fact about density.

▶ **Proposition 11.** *Let X be a random variable over Λ^J, and let $I \subseteq J$ be maximal subset of coordinates such that $H_\infty(X_I) < \delta \cdot b \cdot |I|$. Let $\alpha \in \Lambda^I$ be a value such that*

$$\Pr[X_I = \alpha] > 2^{-\delta \cdot b \cdot |I|}.$$

Then, the random variable $X_{J-I} | X_I = \alpha$ is δ-dense.

We also use the following decomposition result from Göös et al. [13], which extends the last proposition.

▶ **Lemma 12** (Density-restoring partition)**.** *Let X be a random variable over $\mathcal{X} \subseteq \Lambda^J$. Then, there exists a partition*

$$\mathcal{X} \stackrel{\text{def}}{=} \mathcal{X}^1 \cup \cdots \cup \mathcal{X}^r$$

such that every $\mathcal{X}^i$ is associated with a set $I_i \subseteq J$, a value $\alpha_i \in \Lambda^{I_i}$, and a probability $p_{\geq i} \stackrel{\text{def}}{=} \Pr\left[X \in \mathcal{X}^i \cup \ldots \cup \mathcal{X}^r\right]$ that satisfy the following properties: Denote by X^i the random variable X conditioned on $X \in \mathcal{X}^i$.

- $X^i_{I_i}$ *is fixed to* α_i.
- $X^i_{J-I_i}$ *is* 0.9*-dense.*
- $H_\infty(X^i) \geq H_\infty(X) - 0.9 \cdot b \cdot |I_i| - \log \frac{1}{p_{\geq i}}$.

3 The deterministic lifting theorem

In this section, we prove the deterministic lifting theorem, restated from the Introduction.

▶ **Theorem 13** (Restatement of Theorem 2)**.** *Let S be any search problem that takes inputs from $\{0,1\}^n$, and let Π be a deterministic communication protocol that solves $S \circ G$ with complexity c. Then, there exists a decision tree T that solves S with complexity $O(\frac{c}{b})$.*

As noted earlier, the decision tree T we construct would simulate the protocol Π. Throughout the simulation, the tree keeps track of random variables X, Y, which represent the inputs to the protocol, and maintains the invariant that they are ρ-structured. When the protocol Π ends, the decision tree T ends as well and outputs the output of Π. In order to complete the proof of Theorem 2, we need to show three things:

- How to simulate a single bit of the protocol while maintaining the above invariant.
- After the decision tree ends, its output is a correct output of S on z.
- The total number of queries made by the decision tree T during the lifting is $O(\frac{c}{b})$.

Due to space constraints, we will only briefly describe the simulation, relegating its analysis to the full version [3] of the paper.

Consider a given step in the simulation where the tree is at a particular node of the protocol Π. Let $\mathcal{X}, \mathcal{Y}$ be the current set of inputs that are being maintained which are consistent with this node, and let X, Y be random variables uniformly distributed over $\mathcal{X}, \mathcal{Y}$. Let $\rho \in \{0,1,*\}^n$ denote the restriction that represents the queries that have been made so far and their answers, i.e., coordinates that were queried are fixed to the answers that were received, and coordinates that were not queried are free. By the invariant we maintain, the variables X, Y are ρ-structured.

We would like to simulate the next bit of the protocol. Suppose without loss of generality that it is Alice's turn to speak. The tree T chooses the next bit to be the bit that has the highest probability of being sent by Alice, if the inputs are chosen according to X. The tree then updates the set $\mathcal{X}$ to be consistent with the new bit, and updates the random variable X accordingly. Now, if the ρ-structure property of X, Y has been violated, then it must be because $X_{\text{free}(\rho)}$ is no longer 0.9-dense, since the new bit did not affect Y. The tree now modifies the sets $\mathcal{X}, \mathcal{Y}$ and the restriction ρ to restore the structuredness of X, Y. In order to do so, the tree T repeats the following steps iteratively until X and Y are ρ-structured:

1. Condition $X_{\text{free}(\rho)}$ on not taking a value that is 0.4-bad for $Y_{\text{free}(\rho)}$, and update $\mathcal{X}$ accordingly.
2. If $X_{\text{free}(\rho)}$ is now 0.9-dense, then we are done – the structuredness has been restored. Otherwise continue.
3. Let $I \subseteq \text{free}(\rho)$ be a maximal set that violates the density of $X_{\text{free}(\rho)}$ (i.e., $H_\infty(X_I) < 0.9 \cdot b \cdot |I|$), and let $\alpha_I \in \Lambda^I$ be a "heavy" value that satisfies $\Pr[X_I = \alpha_I] > 2^{-0.9 \cdot b \cdot |I|}$.
4. Condition X on $X_I = \alpha_I$, and update $\mathcal{X}$ accordingly. Proposition 11 implies that $X_{\text{free}(\rho)-I}$ is now 0.9-dense.
5. Query the coordinates in I, and update ρ accordingly.
6. Condition Y on $g^I(\alpha_I, Y_I) = \rho_I$, and update $\mathcal{Y}$ accordingly.
7. If $Y_{\text{free}(\rho)}$ is now 0.9-dense then we are done – the structuredness has been restored. Otherwise go back to Step 1 but replace the roles of X and Y.

In order for the steps of the above process to always be well-defined, we need to show that we never condition on events with probability 0. If this is always satisfied, it follows that the algorithm terminates and at termination the random variables X, Y are ρ-structured. To see this, note that the process only stops if $X_{\text{free}(\rho)}$ and $Y_{\text{free}(\rho)}$ are 0.9-dense, and the process clearly maintains the invariant that

$$g^{\text{fix}(\rho)}\left(X_{\text{fix}(\rho)}, Y_{\text{fix}(\rho)}\right) = \rho_{\text{fix}(\rho)}.$$

Moreover, the process always stops, since in every iteration the size of the set $\text{free}(\rho)$ decreases, and it cannot decrease below 0.

We turn to show that we never condition on a zero probability event. To this end, we will show that the process preserves the following property: At the beginning of every iteration, one of the variables $X_{\text{free}(\rho)}$ and $Y_{\text{free}(\rho)}$ is 0.9-dense, and the other is at least 0.4-dense. Observe that this property indeed holds at the beginning of the first iteration: at this point, Y is 0.9-dense, and X must be at least 0.4-dense – since we chose the next bit of Alice to be the one with the highest probability, and therefore the min-entropy of any set of coordinates could have dropped by at most 1.

Suppose that the property holds at the beginning of a given iteration. The first conditioning takes place at Step 1. When Step 1 is performed, we know by Theorem 7 that the event that $X_{\text{free}(\rho)}$ does not take values that are 0.4-bad for $Y_{\text{free}(\rho)}$ has non-zero probability: to see it, note that by assumption $\delta_X \geq 0.4$ and $\delta_Y \geq 0.9$, so it holds that $\delta_X + \delta_Y \geq 1.3$ and $\frac{\delta_Y}{2.01} \geq 0.4$, so the requirements of the theorem are satisfied.

The next conditioning takes place at Step 4, but here the event has non-zero probability by definition. The last conditioning takes place at Step 6, and here the event has non-zero probability due to the assumption that $X_{\text{free}(\rho)}$ does not take values that are bad for $Y_{\text{free}(\rho)}$ – and in particular

$$\Pr\left[g^I(\alpha_I, Y_I) = \rho_I\right] \geq 2^{-|I|-1}.$$

Finally, we need to show that the above property is maintained for the next iteration. As stated in Step 4, at this point X is 0.9-dense. Moreover, since we know that $X_{\text{free}(\rho)}$ does not take values that are 0.4-bad for $Y_{\text{free}(\rho)}$, it follows in particular that

$$Y_{\text{free}(\rho)} \left| g^I(\alpha_I, Y_I) = \rho_I \right.$$

is 0.4-dense. This concludes the proof. The rest of the analysis can be found in the full version [3] of the paper.

3.1 Concluding the simulation

In this section, we prove that when the simulation ends, the protocol Π outputs an answer in $S(z)$. To this end, all we need to prove is that when the simulation ends, we can find $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ such that $G(x, y) = z$: To see why, observe that the output of the protocol at this point must be its output on (x, y), since the rectangle $\mathcal{X} \times \mathcal{Y}$ is contained in the rectangle of the leaf to which the protocol arrived. Now, since we assumed that Π computes $S \circ G$, it follows that its output must be $(S \circ G)(x, y) = S(z)$.

We thus turn to show that there exist $x, y \in \mathcal{X} \times \mathcal{Y}$ such that $G(x, y) = z$. Recall that when the protocol ends, it holds that X, Y are ρ-structured (by the invariant that we maintained). This means that $g^{\text{fix}(\rho)}(X_{\text{fix}(\rho)}, Y_{\text{fix}(\rho)}) = z_{\text{fix}(\rho)}$, and that $X_{\text{free}(\rho)}, Y_{\text{free}(\rho)}$ are 0.9-dense. By Theorem 7, it follows that $X_{\text{free}(\rho)}$ takes a value that is not 0.4-bad for $Y_{\text{free}(\rho)}$ with non-zero probability. This means that there exists some $x \in \mathcal{X}$ such that $x_{\text{free}(\rho)}$ is not 0.4-bad for $Y_{\text{free}(\rho)}$. By the definition of badness, it follows that

$$\Pr\left[g^{\text{free}(\rho)}(x_{\text{free}(\rho)}, Y_{\text{free}(\rho)}) = z_{\text{free}(\rho)}\right] \geq 2^{-|\text{free}(\rho)|-1} > 0$$

and therefore there exists some $y \in \mathcal{Y}$ such that $g^{\text{free}(\rho)}(x_{\text{free}(\rho)}, y_{\text{free}(\rho)}) = z_{\text{free}(\rho)}$. It follows that x and y satisfy

$$\begin{aligned} g^{\text{fix}(\rho)}(x_{\text{fix}(\rho)}, y_{\text{fix}(\rho)}) &= z_{\text{fix}(\rho)} \\ g^{\text{free}(\rho)}(x_{\text{free}(\rho)}, y_{\text{free}(\rho)}) &= z_{\text{free}(\rho)} \end{aligned}$$

and therefore $G(x, y) = z$, as required.

3.2 The query complexity

We conclude by showing that the total number of queries the tree T makes is $O(\frac{c}{b})$. To this end, we define the deficiency of X, Y to be

$$\Delta \stackrel{\text{def}}{=} 2 \cdot b \cdot |\text{free}(\rho)| - H_\infty(X_{\text{free}(\rho)}) - H_\infty(Y_{\text{free}(\rho)}).$$

We prove that whenever the protocol Π transmits a bit in the simulation, the deficiency increases by $O(1)$, and that whenever the tree T makes a query, the deficiency is decreased by $\Omega(b)$. Since the deficiency is always non-negative, and the protocol transmits at most c bits, it follows that the tree must make at most $O(\frac{c}{b})$ bits.

We start by showing that when the protocol Π transmits a bit in the simulation, the deficiency increases by $O(1)$. When a bit is transmitted, either X or Y is conditioned on an event of probability at least $\frac{1}{2}$, depending on which player spoke, and the other variable remains unchanged. This means that the sum $H_\infty(X_{\text{free}(\rho)}) + H_\infty(Y_{\text{free}(\rho)})$ decreases by at most 1, and therefore the deficiency increases by at most 1. Next, the simulation might perform Step 1 in the process above, i.e., condition X or Y on taking a value that is not bad. This event has probability $1 - 2^{-0.01 \cdot b} \geq \frac{1}{2}$, so conditioning on it increases the deficiency by at most 1. All in all, we increased the deficiency by at most 2. All the other steps that might be taken are only taken if a query is being made, so we account their deficiency increases to the following "query part" of the analysis.

We turn to show that when a query is being made, the deficiency decreases by $\Omega(b)$. Suppose that the decision tree queried a set $I \subseteq \text{free}(\rho)$. This applies the following changes to the deficiency:

- The variable X is conditioned on the event $X_I = \alpha_I$, which has probability greater than $2^{-0.9 \cdot b \cdot |I|}$ by the definition of α_I. Hence, this conditioning increases the deficiency by at most $0.9 \cdot b \cdot |I|$.
- The variable Y is conditioned on the event $g^I(\alpha_I, Y_I) = \rho_I$, which has probability at least $2^{-|I|-1}$ by the assumption that X does not take bad values. This increases the deficiency by at most $|I| + 1$.
- The set I is removed from the set $\text{free}(\rho)$. Looking at the definition of deficiency, this decreases the first term, $2 \cdot b \cdot |\text{free}(\rho)|$, by at most $2 \cdot b \cdot |I|$, decreases $H_\infty(Y_{\text{free}(\rho)})$ by at most $b \cdot |I|$, and does not change $H_\infty(X_{\text{free}(\rho)})$ (since at this point X_I is fixed to α_I). All in all, the deficiency is decreased by $b \cdot |I|$.
- Finally, the queries may make the process repeat for another iteration, so Step 1 may be performed again, increasing the deficiency by another 2 bits.

Summing all those effects together, we get that the deficiency was decreased by at least

$$b \cdot |I| - 0.9 \cdot b \cdot |I| - (|I| + 1) - 2 \geq 0.05 \cdot b \cdot |I|$$

in each iteration, as required. This concludes the proof.

4 The randomized lifting theorem

In this section, we prove the randomized lifting theorem, restated next.

▶ **Theorem 14** (Restatement of Theorem 1). *Let S be any search problem that takes inputs from $\{0,1\}^n$, and let Π be a randomized communication protocol that solves $S \circ G$ with complexity c and error probability ε. Then, there exists a decision tree T that solves S with complexity $O(\frac{c}{b})$ and error probability $\varepsilon + \frac{1}{10}$.*

As noted earlier, the decision tree T we construct simulates the protocol Π. The simulation is similar to the deterministic one, with two main differences:

- Instead of choosing the next bit of the protocol to be the most likely bit, we choose it randomly according to the distribution of the next bit (except that we abort the simulation on bits of very small probability).
- Instead of choosing I and α_I arbitrarily, we choose them from the density-restoring partition of Lemma 12, according to the distribution induced by this partition (except that we truncate parts of the partition that have very small probability).

In the following sections, we describe the simulation, analyze its error probability, and analyze its query complexity, respectively. For simplicity, we describe a simulation that has a better error probability of $\varepsilon + o(1)$ but query complexity that is efficient *only in expectation*. This simulation can be transformed into one with error probability $\varepsilon + \frac{1}{10}$, and efficient query complexity in the worst case, using standard arguments.

4.1 The simulation

As before, the decision tree T simulates the protocol Π while maintaining a rectangle $\mathcal{X} \times \mathcal{Y}$ that is contained in the rectangle of the current node of Π. When the simulation ends, T outputs the output of Π. Throughout the simulation, the decision tree T considers random variables X, Y that are uniformly distributed over $\mathcal{X} \times \mathcal{Y}$ and maintains the invariant that they are ρ-structured (for a restriction ρ that records the queries made so far). For the purpose of the simulation, we may assume without loss of generality that Π is deterministic (since T can use its randomness to choose the randomness of Π, and then pretend that Π is deterministic for the rest of the simulation).

We turn to explain how to simulate a single bit of the protocol. Suppose that at a given point it is Alice's turn to speak. The protocol partitions $\mathcal{X}$ into $\mathcal{X}_0 \cup \mathcal{X}_1$. The tree now chooses the next bit to be 0 with probability $\frac{|\mathcal{X}_0|}{|\mathcal{X}|}$ and to be 1 otherwise. If the bit that was chosen had probability less than $\frac{1}{n^2}$, the tree halts and declares error. Otherwise, the tree updates $\mathcal{X}$ to the corresponding set among $\mathcal{X}_0, \mathcal{X}_1$ and updates the random variable X accordingly.

Now, if the ρ-structure property of X, Y has been violated, then it must be because $X_{\mathrm{free}(\rho)}$ is no longer 0.9-dense, since the new bit did not affect Y. The tree now modifies the sets $\mathcal{X}, \mathcal{Y}$ and the restriction ρ to restore the structuredness of X, Y. In order to do so, the tree T repeats the following steps iteratively until X, Y are ρ-structured:

1. Condition $X_{\mathrm{free}(\rho)}$ on not taking a value that is 0.4-bad for $Y_{\mathrm{free}(\rho)}$, and update $\mathcal{X}$ accordingly.
2. If X is now 0.9-dense, then we are done – the structuredness has been restored. Otherwise continue.
3. Let $\mathcal{X}_{\mathrm{free}(\rho)} = \mathcal{X}^1 \cup \ldots \cup \mathcal{X}^r$ be the density-restoring partition of Lemma 12 with respect to $X_{\mathrm{free}(\rho)}$. Choose a random class in the partition, where the class $\mathcal{X}^i$ is chosen with probability $\Pr\left[X_{\mathrm{free}(\rho)} \in \mathcal{X}^i\right]$.

4. Recall that we defined the probability

$$p_{\geq i} \stackrel{\text{def}}{=} \Pr\left[X_{\text{free}(\rho)} \in \mathcal{X}^i \cup \ldots \cup \mathcal{X}^r\right].$$

   If $p_{\geq i} < \frac{1}{n^3}$, the tree T halts and declares error.
5. Let I_i and α_i be the set and the value associated with the class $\mathcal{X}^i$. The tree conditions X on the event $X_{\text{free}(\rho)} \in \mathcal{X}^i$ and updates $\mathcal{X}$ accordingly. The variable $X_{\text{free}(\rho)-I_i}$ is now 0.9-dense by the properties of the density-restoring partition.
6. Query the coordinates in I_i, and update ρ based on the query answers.
7. Condition Y on $g^I(\alpha_i, Y_{I_i}) = \rho_{I_i}$, and update $\mathcal{Y}$ accordingly.
8. If $Y_{\text{free}(\rho)}$ is now 0.9-dense then we are done – the structuredness has been restored. Otherwise go back to Step 1 but replace the roles of X and Y.

The proof that the process is well-defined and always halts, and that the ρ-structuredness invariant is maintained, is the same as in the deterministic simulation. The only difference here is that choosing the next bit of the protocol decreases the min-entropy of the blocks by at most $2\log n$ bits rather than by at most 1 bit. Nevertheless, since the random variable X started as 0.9-dense and $b > 20\log n$, the variable X is still 0.4-dense after choosing the next bit.

4.2 Correctness

We prove that the decision tree errs with probability at most $\varepsilon + o(1)$ (recall that ε is the error probability of the protocol Π). Fix an input $z \in \{0,1\}^n$. Let π be the (random) transcript generated by the simulation of T on z (if we the simulation declares error, we set $\pi = \perp$). Let π' denote the (random) transcript of Π on random inputs (X', Y') that are distributed uniformly over $G^{-1}(z)$ (again, we assume that Π' is deterministic and that the only randomness comes from the choice of (X', Y')). We will prove that the distributions of π and π' are $o(1)$-close. Since π' outputs the correct answer on z with probability at least $1-\varepsilon$, it will follow that π outputs the correct answer on z with probability at least $1 - \varepsilon - o(1)$.

To prove that π and π' are $o(1)$-close, we describe a coupling of π with π' that satisfies that $\pi = \pi'$ with probability at least $1 - o(1)$. To this end, we show that there exists a coupling of the random choices of the simulation with X', Y' such that, up to some bad event $\mathcal{E}$ of small probability, it holds that the pair (X', Y') is uniformly distributed in $G^{-1}(z) \cap (\mathcal{X} \times \mathcal{Y})$. Since $\mathcal{X} \times \mathcal{Y}$ determines the transcript π of the simulation (as $\mathcal{X} \times \mathcal{Y}$ is contained the rectangle of the current node in the protocol), whenever $(X', Y') \in (\mathcal{X}, \mathcal{Y})$ it holds that $\pi = \pi'$.

More specifically, we prove that there exists a coupling and an event $\mathcal{E}$ with probability at most $\frac{6 \cdot b}{n} = o(1)$ such that, when the simulation ends, conditioned on $\neg\mathcal{E}$ it holds that the pair (X', Y') is uniformly distributed in $G^{-1}(z) \cap (\mathcal{X} \times \mathcal{Y})$. To this end, we define a sequence of events $\mathcal{E}_1, \mathcal{E}_2, \ldots$ such that $\Pr[\mathcal{E}_t] \leq \frac{6}{n^2} \cdot (t-1)$ and at the begining of the t-th iteration, conditioned on $\neg\mathcal{E}_t$ it holds that the pair (X', Y') is uniformly distributed in $G^{-1}(z) \cap (\mathcal{X} \times \mathcal{Y})$. We then set $\mathcal{E}$ to be the event at the end of the last iteration. Since the number of iterations is at most $c \leq n \cdot b$ (as each iteration transmits 1-bit), it follows that the probability of $\mathcal{E}$ is at most $\frac{6}{n^2} \cdot c \leq \frac{6b}{n}$. In order to construct the coupling and the events $\mathcal{E}_1, \mathcal{E}_2, \ldots$, we prove the following auxiliary result.

▶ **Lemma 15.** *Suppose that we constructed the coupling until the beginning of the t-th iteration, and there is an event $\mathcal{E}_t$ such that conditioned on $\neg\mathcal{E}_t$ it holds that the pair (X', Y') is uniformly distributed in $G^{-1}(z) \cap (\mathcal{X} \times \mathcal{Y})$. Then, there exists a way to extend the coupling until the end of the t-th iteration, and there exists an event $\mathcal{E}_{t+1}$, such that $\Pr[\mathcal{E}_{t+1}] \leq \Pr[\mathcal{E}_t] + \frac{6}{n^2}$ and at the end of the t-th iteration, conditioned on $\neg\mathcal{E}_{t+1}$ it holds that the pair (X', Y') is uniformly distributed in $G^{-1}(z) \cap (\mathcal{X} \times \mathcal{Y})$.*

Given Lemma 15, we design the coupling and the events $\mathcal{E}_1, \mathcal{E}_2, \ldots$ by setting $\mathcal{E}_1$ to be the empty event and then applying Lemma 15 repeatedly until we reach the last iteration.

Proof. Suppose that the simulation ran until the beginning of the t-th iteration according to our coupling. If the event $\mathcal{E}_t$ happened, then the coupling behaves arbitrarily until the end of the simulation, and we assume that the simulation failed. Let us now condition on the event $\mathcal{E}_t$ not having happened, so we may assume that at the beginning of the t-th iteration, the pair (X', Y') is uniformly distributed in $G^{-1}(z) \cap (\mathcal{X} \times \mathcal{Y})$. We start by setting $\mathcal{E}_{t+1}$ to be the event $\mathcal{E}_t$, and we will add more events to it as the simulation progresses.

The simulation starts by choosing the next bit of the protocol, and suppose that it is Alice's turn to speak. The simulation has probability $\frac{|\mathcal{X}_0|}{|\mathcal{X}|}$ to choose 0, and by the uniform marginals lemma (Lemma 8), the random variable X' has probability $\frac{|\mathcal{X}_0|}{|\mathcal{X}|} \pm \frac{1}{n^3}$ to be in $\mathcal{X}_0$. In other words, the distribution of the class that the simulation chooses among $\mathcal{X}_0, \mathcal{X}_1$, and the distribution of the class that X' chooses, are $\frac{1}{n^3}$-close, and therefore there exists a coupling of those choices such that the same class is chosen in both with probability at least $1 - \frac{1}{n^3}$, so we use it to extend our coupling. We add to $\mathcal{E}_{t+1}$ the event in which the simulation and X' choose a different class among $\mathcal{X}_0, \mathcal{X}_1$, and for the rest of the proof we assume that it did not happen. We also add to $\mathcal{E}_{t+1}$ the event in which the simulation declared failure since it chose a bit with probability less than $\frac{1}{n^2}$ (clearly, this event has probability less than $\frac{1}{n^2}$), and for the rest of the proof we assume that it did not happen. We may thus assume that after this step, the pair (X', Y') is uniformly distributed in $G^{-1}(z) \cap (\mathcal{X} \times \mathcal{Y})$.

Next, the simulation removes from $\mathcal{X}$ the values that are 0.4-bad for Y. The probability that X takes such a value is at most $2^{-0.01 \cdot b} \leq \frac{1}{n^3}$, and therefore the probability that X' takes such a value is at most $\frac{2}{n^3}$ by the uniform marginals lemma. We add the event that X' takes a bad value to $\mathcal{E}_{t+1}$ and assume for the rest of the proof that it did not happen. Hence, we may again assume that after this step, X' belongs to $\mathcal{X}$, and that the pair (X', Y') is uniformly distributed in $G^{-1}(z) \cap (\mathcal{X} \times \mathcal{Y})$.

In the following step, a class $\mathcal{X}^i$ is chosen according to the distribution induced by $X_{\mathrm{free}(\rho)}$. Let us now choose the class $\mathcal{X}^{i'}$ to which $X'_{\mathrm{free}(\rho)}$ belongs. By the uniform marginals lemma, the distributions of $\mathcal{X}^i$ and $\mathcal{X}^{i'}$ are $\frac{1}{n^3}$-close, and therefore there is a coupling of those classes such that they are equal with probability at least $1 - \frac{1}{n^3}$, so we use it to extend our coupling. We add to $\mathcal{E}_{t+1}$ the event in $\mathcal{X}^i \neq \mathcal{X}^{i'}$, and for the rest of the proof we assume that it did not happen. We also add to $\mathcal{E}_{t+1}$ the event in which the simulation declared error since $p_{\leq i} < \frac{1}{n^3}$ (clearly, this event has probability less than $\frac{1}{n^3}$), and for the rest of the proof we assume that it did not happen. We therefore assume again that after this step, X' belongs to $\mathcal{X}$, and that the pair (X', Y') is uniformly distributed in $G^{-1}(z) \cap (\mathcal{X} \times \mathcal{Y})$.

Finally, the simulation conditions Y on $g^I(\alpha_i, Y_{I_i}) = \rho_{I_i}$. This conditioning trivially holds for Y' (since by assumption $(X', Y') \in G^{-1}(z)$ and by this point we chose $X'_{I_i} = \alpha_{I_i}$), and no further coupling needs to be done.

We conclude the proof by upper bounding the probability of the event $\mathcal{E}_{t+1}$. At the beginning, we set $\mathcal{E}_{t+1}$ to be $\mathcal{E}_t$, and therefore at this point its probability is $\Pr[\mathcal{E}_t]$. The step of choosing the next bit of the protocol contribute to $\mathcal{E}_{t+1}$ events whose total probability is at most $\frac{1}{n^3} + \frac{1}{n^2}$. Steps 1 to 7 above add to $\mathcal{E}_{t+1}$ events of total probability at most $\frac{4}{n^3}$. Those latter steps are now repeated until (X, Y) are ρ-structured. However, they may be repeated at most n times, since each time they are repeated, the tree makes at least one query, and it cannot make more than n queries. Hence, in all of those repetitions together, those steps in the simulation contribute to $\mathcal{E}_{t+1}$ events whose total probability is at most $\frac{4}{n^2}$.

It follows that

$$\Pr[\mathcal{E}_{t+1}] \leq \Pr[\mathcal{E}_t] + \frac{1}{n^3} + \frac{1}{n^2} + \frac{4}{n^2} \leq \Pr[\mathcal{E}_t] + \frac{6}{n^2},$$

as required. ◀

4.3 The query complexity

We show that the *expected* query complexity of this simulation is $O(\frac{c}{b})$. Again, we define the deficiency of X, Y to be

$$\Delta \stackrel{\text{def}}{=} 2 \cdot b \cdot |\text{free}(\rho)| - H_\infty(X_{\text{free}(\rho)}) - H_\infty(Y_{\text{free}(\rho)}).$$

We will show that whenever the simulation sends one bit in the protocol, the deficiency is increased by $O(1)$ *in expectation*. On the other hand, we will show that whenever a query is made, the deficiency is always decreased by at least $\Omega(b)$. Thus, the expected deficiency at any point is at most

$$O(\#\text{bits communicated}) - \Omega(b \cdot \#\text{queries}).$$

Since the deficiency is always at least 0 and the number of bits communicated is at most c, it follows that the expected number of queries is upper bounded by $O(\frac{c}{b})$.

Whenever we choose the next bit for Alice, the deficiency increases by $\log \frac{|\mathcal{X}|}{|\mathcal{X}_0|}$ (if the next bit is 0) or by $\log \frac{|\mathcal{X}|}{|\mathcal{X}_1|}$ (if the next bit is 1). Thus, the expected increase in deficiency is

$$\frac{|\mathcal{X}_0|}{|\mathcal{X}|} \cdot \log \frac{|\mathcal{X}|}{|\mathcal{X}_0|} + \frac{|\mathcal{X}_1|}{|\mathcal{X}|} \cdot \log \frac{|\mathcal{X}|}{|\mathcal{X}_1|}.$$

This is the value of the binary entropy function on $\frac{|\mathcal{X}_0|}{|\mathcal{X}|}$, and hence it is upper bounded by 1. Conditioning on X not taking a value that is 0.4-bad for Y increases the deficiency by at most 1 bit since its probability is at least $\frac{1}{2}$. All in all, the expected increase in the deficiency is at most 2.

We turn to show that when a query is being made, the deficiency decreases by $\Omega(b)$. Suppose that the decision tree queried a set $I_i \subseteq \text{free}(\rho)$. This brings about the following changes to the deficiency:

- The variable X was conditioned on the event $X_{\text{free}(\rho)} \in \mathcal{X}^i$. By Lemma 12, this decreases the min-entropy of X by at most $0.9 \cdot b \cdot |I_i| + \log \frac{1}{p_{\geq i}}$. Now, Step 4 guarantees that $p_i \geq \frac{1}{n^3}$, and therefore $\log \frac{1}{p_i} \leq 3 \log n < 0.01 \cdot b$. All in all, this step increases the deficiency by at most $0.91 \cdot |I_i|$
- The variable Y is conditioned on the event $g^{I_i}(\alpha_{I_i}, Y_{I_i}) = \rho_{I_i}$, which has probability at least $2^{-|I_i|-1}$ by the assumption that X does not take bad values. This increases the deficiency by at most $|I_i| + 1$.
- The set I_i is removed from the set $\text{free}(\rho)$. By definition of deficiency, this dereases the term of $2 \cdot b \cdot |\text{free}(\rho)|$ by $2 \cdot b \cdot |I_i|$, decreases $H_\infty(Y_{\text{free}(\rho)})$ by at most $b \cdot |I_i|$, and does not change $H_\infty(X_{\text{free}(\rho)})$ (since at this point X_{I_i} is fixed to α_{I_i}). All in all, the deficiency is decreased by at least $b \cdot |I_i|$.
- Finally, the queries may make the process repeat for another iteration, so Step 1 may be performed again, increasing the deficiency by another 2 bits.

Summing all those effects together, we get that the deficiency was decreased by at least

$$b \cdot |I_i| - 0.91 \cdot b \cdot |I_i| - (|I_i| + 1) - 2 \geq 0.05 \cdot b \cdot |I_i|,$$

as required. This concludes the proof.

5 Fourier-theoretic result

We recall our notation, some definitions and the result. Let $n \in \mathbb{N}$ and let $b \in \mathbb{N}$ be such that $b \geq 40,000 \cdot \log n$. We denote the domain of the inner product gadget by $\Lambda = \{0,1\}^b$ (so the inner product is over $\Lambda \times \Lambda$), and denote $q = |\Lambda| = 2^b$. Given a string $\gamma \in \Lambda$, we denote the corresponding Fourier character by $\chi_\gamma(x) \stackrel{\text{def}}{=} (-1)^{\langle \gamma, x \rangle}$. When considering a set $I \subseteq [n]$ and the space of functions $f \colon \Lambda^I \to \mathbb{R}$, we index the corresponding Fourier characters by tuples from Λ^I, such that for every $\gamma \in \Lambda^I$ it holds that $\chi_\gamma = \prod_{i \in I} \chi_{\gamma_i}$.

▶ **Definition 16.** *Let $\alpha \in \Lambda^n$ and let Y be a random variable taking values in Λ^n. We say that α is* η-bad *for Y if there exists a set $I \subset [n]$ and a string $\sigma \in \{0,1\}^I$ such that the random variable*

$$Y_{[n]-I} \,|\forall_{i \in I} \langle \alpha_i, Y_i \rangle = \sigma_i$$

is not η-dense or

$$\Pr\left[\forall_{i \in I} \langle \alpha_i, Y_i \rangle = \sigma_i\right] < 2^{-|I|-1}.$$

In this section we prove the following result.

▶ **Theorem 17** (Restatement of Theorem 7)**.** *Let X and Y be random variables taking values in Λ^n that are δ_X-dense and δ_Y-dense respectively. Suppose that $\delta_X + \delta_Y \geq 1.3$ and $\delta_Y \geq 0.1$. Then, the probability that X takes a value that is $\frac{\delta_Y}{2.01}$-bad for Y is at most $q^{-0.01}$.*

For the rest of this section, fix the random variables X and Y, and suppose that they are δ_X-dense and δ_Y-dense respectively where $\delta_X + \delta_Y \geq 1.3$ and $\delta_Y \geq 0.1$. We use the following definition, which essentially isolates "badness" to a particular set of coordinates.

▶ **Definition 18.** *Let $\varepsilon > 0$. We say that $\alpha \in \Lambda^n$ is* ε-bad *for Y on $J \subseteq [n]$ if there exist a string $\beta_J \in \Lambda^J$, a non-empty set $I \subset [n] - J$ and a string $\sigma \in \{0,1\}^I$ such that*

$$\Pr\left[Y_J = \beta_J \text{ and } \forall_{i \in I} \langle \alpha_i, Y_i \rangle = \sigma_i\right] \notin 2^{-|I|} \cdot \left(\Pr\left[Y_J = \beta_J\right] \pm \varepsilon\right).$$

In particular, if $J = \emptyset$, we view Y_J, β_J as the empty string and the event $Y_J = \beta_J$ as an event that occurs with probability 1 vacuously.

Morally, a value is not bad if it is not bad on any J. Theorem 17 will follow as a corollary from the following result (see that last part of the full version [3] of the paper).

▶ **Lemma 19.** *For every $J \subseteq [n]$, the probability that X takes a value that is ε-bad for Y on J is at most $q^{-\delta_Y \cdot |J| - 0.05}/\varepsilon^2$.*

In order to analyze the probability of bad values, it is more convenient to consider "unbiased" values, i.e., values α for which the event $Y_J = \beta_J$ is not correlated with inner products of the form $\forall_{i \in I} \langle \alpha_i, Y_i \rangle = \sigma_i$. This bias is naturally measured using Fourier coefficients. We denote by $D \colon \Lambda^n \to [0,1]$ the distribution of Y, i.e., the function that for every $\beta \in \Lambda^n$ outputs $\Pr[Y = \beta]$. For a set of indices $K \subseteq [n]$, we denote by D_K the function corresponding to the marginal distribution over K. Moreover, given disjoint sets $J, K \subseteq [n]$ and a string $\beta_J \in \Lambda^J$ we denote by $D_{K,\beta_J} \colon \Lambda^K \to [0,1]$ the function that maps each $\beta_K \in \Lambda^K$ to $\Pr[Y_K = \beta_K \text{ and } Y_J = \beta_J]$.

▶ **Definition 20.** *We say that a value $\alpha \in \Lambda^n$ is* ε-biased *for Y with respect to $J \subseteq [n]$ if for every non-empty $I \subseteq [n] - J$ and for every $\beta_J \in \Lambda^J$ it holds that $\left|\hat{D}_{I,\beta_J}(\alpha_I)\right| \leq \varepsilon \cdot q^{-1.1 \cdot |I|}$.*

Lemma 19 follows immediately from the next two propositions. The first proposition is a "Vazirani lemma" type of result that shows that small bias implies small distortion of probabilities.

▶ **Proposition 21.** *If a value $\alpha \in \Lambda^n$ is ε-biased for Y with respect to $J \subseteq [n]$, then it is not ε-bad with respect to J.*

The second proposition upper bounds the probability of X taking a value with large bias using the fact that X and Y are δ_X-dense and δ_Y-dense respectively.

▶ **Proposition 22.** *For every $J \subseteq [n]$, the probability that X takes a value that is not ε-biased for Y with respect to J is at most $q^{-\delta_Y \cdot |J| - 0.05}/\varepsilon^2$.*

The rest of the proof can be found in the full version [3] of the paper.

References

1. Boaz Barak, Mark Braverman, Xi Chen, and Anup Rao. How to compress interactive communication. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 67–76, 2010.
2. Mark Braverman, Anup Rao, Omri Weinstein, and Amir Yehudayoff. Direct Products in Communication Complexity. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 746–755, 2013.
3. Arkadev Chattopadhyay, Yuval Filmus, Sajin Koroth, Or Meir, and Toniann Pitassi. Query-to-communication lifting for BPP using inner product, April 2019. `arXiv:1904.13056`.
4. Arkadev Chattopadhyay, Michal Koucký, Bruno Loff, and Sagnik Mukhopadhyay. Simulation Theorems via Pseudorandom Properties. *CoRR*, abs/1704.06807, 2017. `arXiv:1704.06807`.
5. Arkadev Chattopadhyay, Michal Koucký, Bruno Loff, and Sagnik Mukhopadhyay. Simulation beats richness: new data-structure lower bounds. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1013–1020, 2018.
6. Susanna F. de Rezende, Jakob Nordström, and Marc Vinyals. How Limited Interaction Hinders Real Communication (and What It Means for Proof and Circuit Complexity). In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 295–304, 2016.
7. Tomás Feder, Eyal Kushilevitz, Moni Naor, and Noam Nisan. Amortized Communication Complexity. *SIAM J. Comput.*, 24(4):736–750, 1995. `doi:10.1137/S0097539792235864`.
8. Ankit Garg, Mika Göös, Pritish Kamath, and Dmitry Sokolov. Monotone circuit lower bounds from resolution. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 902–911, 2018.
9. Mika Göös and T. S. Jayram. A Composition Theorem for Conical Juntas. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 5:1–5:16, 2016. `doi:10.4230/LIPIcs.CCC.2016.5`.
10. Mika Göös, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman. Rectangles Are Nonnegative Juntas. *SIAM J. Comput.*, 45(5):1835–1869, 2016.
11. Mika Göös and Toniann Pitassi. Communication Lower Bounds via Critical Block Sensitivity. *SIAM J. Comput.*, 47(5):1778–1806, 2018. `doi:10.1137/16M1082007`.
12. Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic Communication vs. Partition Number. In *Proceedings of IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1077–1088, 2015.
13. Mika Göös, Toniann Pitassi, and Thomas Watson. Query-to-Communication Lifting for BPP. In *Proceedings of IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 132–143, 2017.

14 Hamed Hatami, Kaave Hosseini, and Shachar Lovett. Structure of Protocols for XOR Functions. *SIAM J. Comput.*, 47(1):208–217, 2018.

15 Mauricio Karchmer, Ran Raz, and Avi Wigderson. Super-logarithmic Depth Lower Bounds via Direct Sum in Communication Coplexity. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference, Chicago, Illinois, USA, June 30 - July 3, 1991*, pages 299–304, 1991.

16 Hartmut Klauck. A strong direct product theorem for disjointness. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 77–86, 2010.

17 Alexander Kozachinskiy. From Expanders to Hitting Distributions and Simulation Theorems. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, pages 4:1–4:15, 2018.

18 Bruno Loff and Sagnik Mukhopadhyay. Lifting Theorems for Equality. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:175, 2018.

19 Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, 1999.

20 Alexander A. Sherstov. The Pattern Matrix Method. *SIAM J. Comput.*, 40(6):1969–2000, 2011.

21 Yaoyun Shi and Yufan Zhu. Quantum communication complexity of block-composed functions. *Quantum Information & Computation*, 9(5):444–460, 2009.

22 Xiaodi Wu, Penghui Yao, and Henry S. Yuen. Raz-McKenzie simulation with the inner product gadget. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:10, 2017. URL: `https://eccc.weizmann.ac.il/report/2017/010`.

23 Andrew C. Yao. Theory and Application of Trapdoor Functions. In *Proceedings of IEEE 23rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 80–91, 1982.

Estimating the Frequency of a Clustered Signal

Xue Chen
Northwestern University, Evanston, IL, USA
xue.chen1@northwestern.edu

Eric Price
The University of Texas at Austin, USA
ecprice@cs.utexas.edu

Abstract

We consider the problem of locating a signal whose frequencies are "off grid" and clustered in a narrow band. Given noisy sample access to a function $g(t)$ with Fourier spectrum in a narrow range $[f_0 - \Delta, f_0 + \Delta]$, how accurately is it possible to identify f_0? We present generic conditions on g that allow for efficient, accurate estimates of the frequency. We then show bounds on these conditions for k-Fourier-sparse signals that imply recovery of f_0 to within $\Delta + \widetilde{O}(k^3)$ from samples on $[-1, 1]$. This improves upon the best previous bound of $O\big(\Delta + \widetilde{O}(k^5)\big)^{1.5}$. We also show that no algorithm can do better than $\Delta + \widetilde{O}(k^2)$.

In the process we provide a new $\widetilde{O}(k^3)$ bound on the ratio between the maximum and average value of continuous k-Fourier-sparse signals, which has independent application.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases sublinear algorithms, Fourier transform

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.36

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1904.13043`.

Funding *Xue Chen*: Supported by research funding from Northwestern University. Part of this work was done while the author was in the University of Texas at Austin supported by NSF Grant CCF-1526952 and a Simons Investigator Award (#409864, David Zuckerman).
Eric Price: Supported in part by NSF Award CCF-1751040 (CAREER).

Acknowledgements We thank Daniel Kane and Zhao Song for many helpful discussions. We also thank the anonymous referee for the detailed feedback and comments.

1 Introduction

A natural question, dating at least to the work of Prony in 1795, is to estimate a signal from samples, assuming the signal has a k-sparse Fourier representation, i.e., that the signal is a sum of k complex exponentials: $g(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$ for some set of frequencies f_j and coefficients v_j.

If the frequencies are located on a discrete grid (giving a sparse discrete Fourier transform), then a long line of work has studied efficient algorithms for recovering the signal (e.g., [11, 7, 1, 8, 9, 10]). If the frequencies are not on a grid, then Prony's method from 1795 [14] or matrix pencil [3] can still identify them in the absence of noise. With noise, however, one cannot robustly recover frequencies that are too close together: if one listens to a signal for the interval $[-T, T]$ then any two frequencies θ and $\theta + \varepsilon/T$ will be $O(\varepsilon)$-close to each other, and so cannot be distinguished with noise. As shown in [12], this nonrobustness grows exponentially in k. On the other hand, [12] also showed that recovery with polynomially

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 36; pp. 36:1–36:13


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

small noise is possible if all the frequencies have separation $1/2T$, and [13] showed that a constant fraction of noise is tolerable with separation $\log^{O(1)}(FT)/T$, where F is the bandlimit of the signal.

So what *is* possible for arbitrary Fourier-sparse signals, without any assumption of frequency separation? One cannot hope to identify the frequencies exactly, but one can still estimate the *signal itself*. If two frequencies are similar enough to be indistinguishable over the sampled interval, we do not need to distinguish them. In [4], this led to an algorithm for an arbitrary k-Fourier-sparse signal that used $\mathrm{poly}(k, \log(FT))$ samples to estimate it with only a constant factor increase in the noise. However, this polynomial is fairly poor.

Since prior work could handle the case of well-separated frequencies, a key challenge in [4] is the setting with all the frequencies in a narrow cluster. Formally, consider the following subproblem: if all the frequencies f_i of the signal lie in a narrow band $[f_0 - \Delta, f_0 + \Delta]$, how accurately can we estimate f_0? Note that while we would like an efficient algorithm that takes a small number of samples, the key question is *information theoretic*. And we can ask this question more generally: if the signal is not k-sparse, but still has all its frequencies in a narrow band, can we locate that band?

▶ **Question 1.** *Let $g(t)$ be a signal with Fourier transform supported on $[f_0 - \Delta, f_0 + \Delta]$, for some $f_0 \in [-F, F]$. Suppose that we can sample from $y(t) = g(t) + \eta(t)$ at points in $[-T, T]$, where $\eta(t)$ could be any ℓ_2 bounded noise on $[-T, T]$ with*

$$\mathop{\mathbb{E}}_{t\in[-T,T]}\left[|\eta(t)|^2\right] \le \varepsilon \mathop{\mathbb{E}}_{t\in[-T,T]}\left[|g(t)|^2\right]$$

for a small constant ε. Under what conditions on g can we estimate f_0, and how accurately?

One might expect to be able to estimate f_0 to $\pm(\Delta + O(\frac{1}{T}))$ for all functions g; after all, g is just a combination of individual frequencies, each of which points to some frequency in the right range, and each individual frequency in isolation can be estimated to within $\pm O(\frac{1}{T})$ in the presence of noise. Unfortunately, this intuition is false.

To see this, consider the family of k-sparse Fourier functions with $f_j = \varepsilon j$, i.e.,

$$\mathrm{span}(e^{2\pi \mathbf{i}(j\varepsilon)t} \mid j \in [k]).$$

By sending $\varepsilon \to 0$ and taking a Taylor expansion, this family can get arbitrarily close to any degree $k-1$ polynomial, on any interval $[-T', T']$. Thus, to solve the question, one would also need to solve it when $g(t)$ is a polynomial even for arbitrarily small Δ.

There are two ways in which $g(t)$ being a degree d polynomial can lead to trouble. The first is that $g(t)$ could itself be a Taylor expansion of $e^{\pi \mathbf{i} f t}$. If $d \gtrsim fT$, this Taylor approximation will be quite accurate on $[-T, T]$; with the noise η, the observed signal can *equal* $e^{\pi \mathbf{i} f t}$. Thus the algorithm has to output f, which can be $\Theta(d/T)$ far from the "true" answer $f_0 = 0$.

The second way in which $g(t)$ can lead to trouble is by removing most of the signal energy. If $g(t)$ is the (slightly shifted) Chebyshev polynomial $g(t) = T_d\big(t/T + O(\frac{\log^2 d}{d^2})\big)$, then $|g(t)| \le 1$ for $t \le \big(1 - O(\frac{\log^2 d}{d^2})\big)T$, while $g(t) \ge d$ for $t \ge \big(1 - O(\frac{\log^2 d}{d^2})\big)T$. That is to say, the majority of the ℓ_2 energy of g can lie in the final $O(\frac{\log^2 d}{d^2})$ fraction of the interval. In such a case, a small constant noise level η can make samples outside that $T \cdot \widetilde{O}(1/d^2)$ size region equal to zero, and hence completely uninformative; and samples in that region still have to tolerate noise. This leads to an "effective" interval size of $T' = T \cdot \widetilde{O}(\frac{1}{d^2})$, leading to accuracy $O(1/T') = \widetilde{O}(d^2)/T$.

Our main result is that, in a sense, these two types of difficulties are the only ones that arise. We can measure the second type of difficulty by looking at how much larger the maximum value of g is than its average:

$$R := \frac{\sup_{t\in[-T,T]} |g(t)|^2}{\mathbb{E}_{t\in[-T,T]} |g(t)|^2}.$$

We can measure the former by observing that while a polynomial may approximate a complex exponential on a bounded region, as $t \to \infty$ the polynomial will blow up. In particular, we take the S such that

$$|g(t)|^2 \le \mathsf{poly}(R) \cdot \mathop{\mathbb{E}}_{t\in[-T,T]} \left[|g(t)|^2\right] \cdot |\frac{t}{T}|^S$$

for all $|t| \ge T$. We show that if R and S are bounded, one can estimate f_0 to within $\Delta + \widetilde{O}(R+S)/T$, which is almost tight from the above discussion of polynomials. Moreover, the time and number of samples required are fairly efficient:

▶ **Theorem 2.** *Given any $T > 0, F > 0, \Delta > 0, R$, and $S > 0$, let $g(t)$ be a signal with the following properties:*

1. $\mathsf{supp}(\widehat{g}) \subseteq [f_0 - \Delta, f_0 + \Delta]$ *where* $f_0 \in [-F, F]$.
2. $\sup_{t\in[-T,T]} \left[|g(t)|^2\right] \le R \cdot \mathop{\mathbb{E}}_{t\in[-T,T]} \left[|g(t)|^2\right]$.
3. $|g(t)|^2$ *grows as at most* $\mathsf{poly}(R) \cdot \mathop{\mathbb{E}}_{t\in[-T,T]} \left[|g(t)|^2\right] \cdot |\frac{t}{T}|^S$ *for* $t \notin [-T,T]$.

Let $y(t) = g(t) + \eta(t)$ be the observable signal on $[-T,T]$, where $\mathop{\mathbb{E}}_{t\in[-T,T]} \left[|\eta(t)|^2\right] \le \epsilon \cdot \mathop{\mathbb{E}}_{t\in[-T,T]} \left[|g(t)|^2\right]$ for a sufficiently small constant ϵ. For $\Delta' = \Delta + \frac{\widetilde{O}(R+S)}{T}$ and any $\delta > 0$, there exists an efficient algorithm that takes $O(R \log \frac{F}{\Delta'\cdot\delta})$ samples from $y(t)$ and outputs $\widetilde{f}$ satisfying $|f_0 - \widetilde{f}| \le O(\Delta')$ with probability at least $1 - \delta$.

Application to sparse Fourier transforms. Specializing to k-Fourier-sparse signals, we give bounds on R and S for this family. Since (as described above) this family can approximate degree-$(k-1)$ polynomials, we know that $R \gtrsim k^2$ and $S \gtrsim k$; we show that $R \lesssim k^3 \log^2 k$ and $S \lesssim k^2 \log k$. Thus, whenever R is between k^2 and $\widetilde{O}(k^3)$, we can identify k-Fourier-sparse signals to within $\Delta + \widetilde{O}(R)/T$. This is an improvement over the results in [4] in several ways.

Formally, for a given sparsity level k, we consider signals in

$$\mathcal{F} := \left\{ g(t) = \sum_{j=1}^{k} v_j e^{2\pi \boldsymbol{i} f_j t} \middle| f_j \in [-F, F] \right\}.$$

▶ **Theorem 3.** *For any k and T,*

$$R := \sup_{g\in\mathcal{F}} \frac{\sup_{x\in[-T,T]} |g(x)|^2}{\mathop{\mathbb{E}}_{x\in[-T,T]} [|g(x)|^2]} = O(k^3 \log^2 k). \tag{1}$$

It was previously known that $R \lesssim k^4 \log^3 k$ [4], and this fact was used in [2]. (Thus, our improved bound on R immediately implies an improvement in Theorem 8 of [2], from $s_{\mu,\varepsilon}^5 \log^3 s_{\mu,\varepsilon}$ to $s_{\mu,\varepsilon}^4 \log^2 s_{\mu,\varepsilon}$.)

Next we bound the growth $S = \widetilde{O}(k^2)$ for any $|t| \ge T$.

▶ **Theorem 4.** *There exists $S = O(k^2 \log k)$ such that for any $|t| > T$ and $g(t) = \sum_{j=1}^{k} v_j \cdot e^{2\pi \boldsymbol{i} f_j t}$, $|g(t)|^2 \le \mathsf{poly}(k) \cdot \mathop{\mathbb{E}}_{x\in[-T,T]}[|g(x)|^2] \cdot |\frac{t}{T}|^S$.*

This is analogous to Theorem 5.5 of [4], which proves a bound of $(kt)^k$ rather than $t^{\widetilde{O}(k^2)}$. These bounds are incomparable, but the $t^{\widetilde{O}(k^2)}$ bound is actually more useful for this problem: what really matters is showing that $g(t)$ is not too large just outside the interval. Theorem 4 gives the "correct" polynomial dependence at $t = (1 + 1/k^2)T$.

We can now apply Theorem 2 to get an efficient algorithm to recover the center of a cluster of k frequencies within accuracy $\tilde{O}(R)$.

▶ **Theorem 5.** *Given F, T, and k, let R be the ratio between the maximum and average value of continuous k-Fourier-sparse signals defined in (1). Given Δ, let $g(t)$ be a k-Fourier-sparse signal centered around f_0: $g(t) = \sum_{i\in[k]} v_i \cdot e^{2\pi \mathbf{i} f_i t}$ where each $f_i \in [f_0 - \Delta, f_0 + \Delta]$ and $y(t) = g(t) + \eta(t)$ be the observable signal on $[-T, T]$, where $\mathop{\mathbb{E}}_{t\in[-T,T]}\left[|\eta(t)|^2\right] \le \epsilon \cdot \mathop{\mathbb{E}}_{t\in[-T,T]}\left[|g(t)|^2\right]$ for a sufficiently small constant ϵ.*

For any $\delta > 0$, there exist $\Delta' = \Delta + \frac{\tilde{O}(R)}{T}$ and an efficient algorithm that takes $O(k \log^2 k \log \frac{F}{\Delta' \cdot \delta})$ samples from $y(t)$ and outputs $\widetilde{f}$ satisfying $|f_0 - \widetilde{f}| \le O(\Delta')$ with probability at least $1 - \delta$.

Note that the sample complexity here is $\widetilde{O}(k)$ not $\widetilde{O}(R)$. This is because, based on the structure of the problem, we can use a nonuniform sampling procedure that performs better. Otherwise this theorem is just Theorem 2 applied to the R and S from Theorems 3 and 4.

Theorem 5 is a direct improvement on Theorem 7.5 of [4], which for $T = 1$ could estimate to within $O\left(\Delta + \widetilde{O}(k^5)\right)^{1.5}$ accuracy and used $\mathrm{poly}(k)$ samples. In particular, in addition to improving the additive $\mathrm{poly}(k)$ term, our result avoids a multiplicative increase in the bandwidth Δ of g.

The main technical lemma in proving Theorems 2 and 5 is a filter function H with a compact supported Fourier transform $\widehat{H}$ that simulates a box function on $[-T, T]$ for any g satisfying the conditions in Theorem 2.

▶ **Lemma 6.** *Given any T, S, and R, there exists a filter function H with $\left|\mathsf{supp}(\widehat{H})\right| \le \frac{\tilde{O}(R+S)}{T}$ such that for any $g(t)$ satisfying the second and third conditions in Theorem 2,*

1. *H is close to a box function on $[-T, T]$: $\int_{-T}^{T} |g(t) \cdot H(t)|^2 \mathrm{d}t \ge 0.9 \int_{-T}^{T} |g(t)|^2 \mathrm{d}t$.*
2. *The tail of $H(t) \cdot g(t)$ is small: $\int_{-T}^{T} |g(t) \cdot H(t)|^2 \mathrm{d}t \ge 0.95 \int_{-\infty}^{\infty} |g(t) \cdot H(t)|^2 \mathrm{d}t$.*

Organization. We introduce some notation and tools in Section 2. Then we provide a technical overview in Section 3. We show our filter function and prove Lemma 6 in Section 4. Next we present the algorithm about frequency estimation of Theorem 2 in Section 5. Finally we prove the results about sparse Fourier transform – Theorem 3 and Theorem 4 in Section 6.

2 Preliminaries

In the rest of this work, we fix the observation interval to be $[-1, 1]$ and define

$$\|g\|_2 = \left(\mathop{\mathbb{E}}_{x\sim[-1,1]} |g(x)|^2\right)^{1/2}, \tag{2}$$

because we could rescale $[-T, T]$ to $[-1, 1]$ and $[-F, F]$ to $[-FT, FT]$.

We first review several facts about the Fourier transform. The Fourier transform $\widehat{g}(f)$ of an integrable function $g : \mathbb{R} \to \mathbb{C}$ is

$$\widehat{g}(f) = \int_{-\infty}^{+\infty} g(t)e^{-2\pi \boldsymbol{i} ft}\mathrm{d}t \text{ for any real } f.$$

We use $g \cdot h$ to denote the pointwise dot product $g(t) \cdot h(t)$ and g^k to denote $\underbrace{g(t) \cdots g(t)}_{k}$. Similarly, we use $g*h$ to denote the convolution of g and h: $\int_{-\infty}^{+\infty} g(x) \cdot h(t-x)\mathrm{d}x$. In this work, we always set g^{*k} as the convolution $\underbrace{g(t) * \cdots * g(t)}_{k}$. Notice that $\mathsf{supp}(g \cdot h) = \mathsf{supp}(g) \cap \mathsf{supp}(h)$ and $\mathsf{supp}(g * h) = \mathsf{supp}(g) + \mathsf{supp}(h)$.

We define the box function and its Fourier transform sinc function as follows. Given a width $s > 0$, the box function $\mathrm{rect}_s(t) = 1/s$ iff $|t| \leq s/2$; and its Fourier transform is $\mathrm{sinc}(sf) = \frac{\sin(\pi fs)}{\pi fs}$ for any f.

We state the Chernoff bound for random sampling [6].

▶ **Lemma 7.** *Let $X_1, X_2, \cdots, X_n$ be independent random variables in $[0, R]$ with expectation 1. For any $\varepsilon < 1/2$ and $n \gtrsim \frac{R}{\epsilon^2}$, $X = \frac{\sum_{i=1}^{n} X_i}{n}$ with expectation 1 satisfies*

$$\Pr[|X - 1| \geq \varepsilon] \leq 2\exp(-\frac{\varepsilon^2}{3} \cdot \frac{n}{R}).$$

3 Proof Overview

We first outline the proofs of Lemma 6 and Theorem 2. Then we show the proof sketch of $R = \tilde{O}(k^3)$ and $S = \tilde{O}(k^2)$ of k-Fourier-sparse signals.

The filter functions $(H, \widehat{H})$ in Lemma 6. Ideally, to satisfy the two claims in Lemma 6, we could set $H(t)$ to be the box function $2\,\mathrm{rect}_2(t)$ on $[-1, 1]$. However, by the uncertainty principle, it is impossible to make its Fourier transform $\widehat{H}$ compact using such an $H(t)$. Hence our construction of $(H, \widehat{H})$ is in the inverse direction: we build $\widehat{H}(f)$ by box functions and $H(t)$ by the Fourier transform of box functions – the sinc function. In the rest of this discussion, we focus on using the sinc function to prove Lemma 6 given the properties of g in Theorem 2.

We first notice that any H with the following two properties is effective in Lemma 6 for g satisfying $|g(t)|^2 \leq R \cdot \|g\|_2^2$ for any $|t| \leq 1$ and $|g(t)|^2 \leq \mathsf{poly}(R)\|g\|_2^2 \cdot |t|^S$ for $|t| > 1$:

1. $H(t) = 1 \pm 0.01$ for any $t \in [-1 + \frac{1}{C \cdot R}, 1 - \frac{1}{C \cdot R}]$ of a large constant C. This shows

$$\int_{-1}^{1} |H(t) \cdot g(t)|^2\mathrm{d}t \geq 0.99^2 \int_{-1+\frac{1}{C \cdot R}}^{1-\frac{1}{C \cdot R}} |g(t)|^2\mathrm{d}t.$$

   Because $|g(t)|^2 \leq R \cdot \|g\|_2^2$ for any $t \in [-1, 1] \setminus [-1 + \frac{1}{C \cdot R}, 1 - \frac{1}{C \cdot R}]$, the constant on the R.H.S. is at least $0.99^2 \cdot (1 - \frac{1}{C}) \geq 0.9$, which implies the first claim of Lemma 6.
2. $H(t)$ declines to $\frac{1}{\mathsf{poly}(R) \cdot t^{2S}}$ for any $|t| > 1$. This shows

$$\int_{1}^{\infty} |H(t) \cdot g(t)|^2\mathrm{d}t \leq 0.01 \int_{-1}^{1} |g(t)|^2\mathrm{d}t,$$

   which implies the second claim.

For ease of exposition, we start with $S = 0$. We plan to design a filter $H_0(t)$ with compact $\widehat{H_0}$ dropping from 0.99 at $t = 1 - \frac{1}{C \cdot R}$ to $\frac{1}{\mathsf{poly}(R)}$ at $t = 1$ in a small range $\frac{1}{CR}$ using the sinc function. To apply the sinc function, we notice that

$$\mathrm{sinc}(CR \cdot t)^{O(\log R)} = \left(\frac{\sin(\pi CR \cdot t)}{\pi CR \cdot t}\right)^{O(\log R)}$$

decays from 1 at $t = 0$ to $1/\mathsf{poly}(R)$ at $t = \frac{1}{C \cdot R}$, which matches the dropping of $H_0(t)$ from $t = 1 - \frac{1}{C \cdot R}$ to $t = 1$.

Then, to make $H(t) \approx 1$ for any $|t| \le 1 - \frac{1}{C \cdot R}$, let us consider a convolution of $\mathrm{rect}_1(t)$ and $\mathrm{sinc}(CR \cdot t)^{O(\log R)}$. Because most of the mass of the latter is in $[-\frac{1}{CR}, \frac{1}{CR}]$, this convolution keeps almost the same value in $[-\frac{1}{2} + \frac{1}{CR}, \frac{1}{2} - \frac{1}{CR}]$ and drops down to $1/\mathsf{poly}(R)$ at $t = \frac{1}{2} + \frac{1}{CR}$. At the same time, it will keep the compactness of $\widehat{H_0}$ since it corresponds to the dot product on the Fourier domain. By normalizing and scaling, this gives the desired $(H_0, \widehat{H_0})$ for $S = 0$.

Next we describe the construction of $S > 0$. The high level idea is to consider the decays of $H(t)$ in $\log_2 S + O(1)$ segments rather than one segment of $S = 0$:

$$(1 - \frac{1}{CR}, 1], (1, 1 + \frac{1}{S}], (1 + \frac{1}{S}, 1 + \frac{2}{S}], \ldots, (1 + \frac{2^j}{S}, 1 + \frac{2^{j+1}}{S}], \ldots, (1 + \frac{S/2}{S}, 2], (2, +\infty).$$

For each segment, we provide a power of sinc functions matching its decay in $H(t)$ like the construction of H_0 on $(1 - \frac{1}{CR}, 1]$. The final construction is the convolution of the dot product of all sinc powers and a box function, which appears in Section 4.

The Algorithm of Theorem 2. Now we show how to estimate f_0 given the observable signal $y = g + \eta$ where $\mathsf{supp}(\widehat{g}) \subseteq [f_0 - \Delta, f_0 + \Delta]$ and $\|\eta\|_2^2 \le \varepsilon \|g\|_2^2$ (with ℓ_2 norm taken over $[-T, T]$ defined in (2)). We instead consider $y_H(t) = y(t) \cdot H(t)$ with the filter function $(H, \widehat{H})$ from Lemma 6 and the corresponding dot products $g_H = g \cdot H$ and $\eta_H = \eta \cdot H$. The starting point is that for a sufficiently small β, we expect

$$y_H(t + \beta) \approx e^{2\pi \boldsymbol{i} f_0 \beta} \cdot y_H(t)$$

because y_H has Fourier spectrum concentrated around f_0. This does not hold for *all* t, but it does hold on average:

$$\int_{-1}^{1} |y_H(t + \beta) - e^{2\pi \boldsymbol{i} f_0 \beta} \cdot y_H(t)|^2 \mathrm{d}t \lesssim \varepsilon \cdot \int_{-1}^{1} |y_H(t)|^2 \mathrm{d}t. \tag{3}$$

This is because we can use Parseval's identity to replace these integrals by an integral over Fourier domain – Parseval's identity would apply if the integrals were from $-\infty$ to ∞, but because of the filter function H, relatively little mass in y_H lies outside $[-1, 1]$. Then, the Fourier transform of the term inside the left square is $e^{2\pi \boldsymbol{i} f \beta} \cdot \widehat{y_H}(f) - e^{2\pi \boldsymbol{i} f_0 \beta} \cdot \widehat{y_H}(f)$. Note that $\widehat{y_H} = \widehat{g_H} + \widehat{\eta_H}$ has most of its ℓ_2 mass in $\mathsf{supp}(g_H) \subseteq [f_0 - \Delta', f_0 + \Delta']$ for $\Delta' = \Delta + |\mathsf{supp}(\widehat{H})|$, and every such frequency shrinks in the left by a factor $|e^{2\pi \boldsymbol{i} f \beta} - e^{2\pi \boldsymbol{i} f_0 \beta}| = O(\beta \Delta')$. Thus, for $\beta \ll 1/\Delta'$, (3) holds.

To learn f_0 through $e^{2\pi \boldsymbol{i} f_0 \beta}$, we design a sampling procedure to output α satisfying

$$|y_H(\alpha + \beta) - e^{2\pi \boldsymbol{i} f_0 \beta} y_H(\alpha)| \le 0.3 \cdot y_H(\alpha) \text{ with probability } \textit{more than half}.$$

Even though the above discussion shows the left hand side is smaller than the R.H.S. on average, a uniformly random $\alpha \sim [-1, 1]$ may not satisfy it with good probability: $|y_H(\alpha)| \ge \|y_H\|_2$ may be only true for $1/R$ fraction of $\alpha \in [-1, 1]$, while the corruption by

adversarial noise η has $\|\eta\|_2^2 \gtrsim \varepsilon\|y_H\|_2^2$ for a constant $\varepsilon \gg 1/R$. At the same time, even for many points $\alpha_1, \ldots, \alpha_m$ where some of them satisfy the above inequality, it is infeasible to verify such an α_i given f_0 is unknown. We provide a solution by adopting the importance sampling: for $m = O(R)$ random samples $\alpha_1, \ldots, \alpha_m \in [-1, 1]$, we output α with probability proportional to the weight $|y_H(\alpha_i)|^2$.

We prove the correctness of this sampling procedure in Lemma 11 in Section 5.

Finally, learning $e^{2\pi \mathbf{i} f_0 \beta}$ is not enough to learn f_0: because of the noise, we only learn $e^{2\pi \mathbf{i} f_0 \beta}$ to within a constant ε, which gives f_0 to within $\pm O(\varepsilon/\beta)$; and because of the different branches of the complex logarithm, this is only up to integer multiples of $1/\beta$. Therefore to fully learn f_0, we repeat the sampling procedure at logarithmically many different scales of β, from $\beta = 1/2F$ to $\beta = \frac{\Theta(1)}{\Delta'}$.

***k*-Fourier-sparse signals.** Finally, we show $R = \widetilde{O}(k^3)$ and $S = \widetilde{O}(k^2)$ such that for any $g(t) = \sum_{j=1}^k v_j \cdot e^{2\pi \mathbf{i} f_j t}$ – not necessarily one with the f_j clustered together –

$$\frac{\sup_{t \in [-1,1]} |g(t)|^2}{\|g\|_2^2} \le R \text{ and } |g(t)|^2 \le \mathsf{poly}(R) \cdot \|g\|_2^2 \cdot |t|^S.$$

We first review the previous argument of $R = \widetilde{O}(k^4)$ [4]. The key point is to show for some $d = \widetilde{O}(k^2)$ that $g(1)$ is a linear combination of $g(1-\theta), \ldots, g(1-d\cdot\theta)$ using bounded integer coefficients $c_1, \ldots, c_d = O(1)$ for any $\theta \le \frac{2}{d}$. Then

$$g(1) = \sum_{j \in [d]} c_j \cdot g(1 - j\cdot\theta) \text{ implies } |g(1)|^2 \le (\sum_{j\in[d]} |c_j|^2) \cdot (\sum_{j\in[d]} |g(1-j\cdot\theta)|^2). \tag{4}$$

If we think of $g(1)$ as the supremum and $g(1-j\cdot\theta)$ as the average $\|g\|_2$ – which we can formally do up to logarithmic factors by averaging over θ – this shows $|g(1)|^2 \le \widetilde{O}(d^2)\|g\|_2^2$. One natural idea to improve it is to use a smaller value d and a shorter linear combination [5]. However, $d = \tilde{\Omega}(k^2)$ for such a combination when g is approximately the degree $k-1$ Chebyshev polynomial. In this work, we use a geometric sequence to control c_j such that $\sum_j |c_j|^2 = O(d/k)$ instead of $O(d)$, which provides an improvement of a factor $\widetilde{O}(k)$ on R.

Then we bound $S = \widetilde{O}(k^2)$ for $g(t)$ at $|t| > 1$. The intuition is that given (4) holds for any $g(t)$ in terms of $g(t-\theta), \ldots, g(t-d\cdot\theta)$ with $\theta = \frac{2}{d}$, it implies $|g(t)|^2 \le \mathsf{poly}(k)\cdot\|g\|_2^2 \cdot e^{(t-1)\cdot O(d)}$ for $t > 1$. Combining this with an alternate bound $|g(t)|^2 \le \mathsf{poly}(k) \cdot \|g\|_2^2 \cdot (k\cdot t)^{O(k)}$ for $t > 1 + 1/k$, it completes the proof of Theorem 4 about S.

Finally we notice that we could improve the sample complexity in Theorem 5 to $\widetilde{O}(k) \log \frac{F}{\Delta'}$ using a biased distribution [5] to generate α. These results about k-Fourier-sparse signals appear in Section 6.

4 Our Filter Function

The main result is an explicit filter function H with compact support $\widehat{H}$ that is close to the box function on $[-1, 1]$ for any g satisfying the conditions in Theorem 2.

We show our filter function as follows.

▶ **Definition 8.** *Given R, the growth rate S and an even constant C, we define the filter function*

$$H(t) = s_0 \cdot \left(\mathrm{sinc}(CR\cdot t)^{C\log R} \cdot \mathrm{sinc}\left(C\cdot S\cdot t\right)^C \cdot \mathrm{sinc}\left(\frac{C\cdot S}{2}\cdot t\right)^{2C} \cdots \mathrm{sinc}\left(C\cdot t\right)^{C\cdot S}\right) * \mathrm{rect}_2(t)$$

where $s_0 \in \mathbb{R}^+$ *is a parameter to normalize* $H(0) = 1$. *On the other hand, its Fourier transform is*

$$\widehat{H}(f) = s_0 \cdot \left(\mathrm{rect}_{CR}(f)^{*C\log R} * \mathrm{rect}_{C\cdot S}(f)^{*C} * \mathrm{rect}_{\frac{C\cdot S}{2}}(f)^{*2C} * \cdots * \mathrm{rect}_{C}(f)^{*CS}\right) \cdot \mathrm{sinc}(2t),$$

whose support size is $O(CR \cdot C\log R + CS \cdot C + \cdots + C \cdot C \cdot S) = O(R\log R + S\log S)$.

We prove Lemma 6 using $H(\alpha x)$ with a large constant C and a scale parameter $\alpha = \frac{1}{2} + \frac{1.2}{\pi CR}$. For convenience, we state the full version of Lemma 6 for $T = 1$ as follows.

▶ **Theorem 9.** *Let* $R, S > 0$, *let* C *be a large even constant, and define* $\alpha = (\frac{1}{2} + \frac{1.2}{\pi CR})$. *Consider any function* g *satisfying the following two conditions:*

1. $\sup_{t\in[-1,1]} |g(t)|^2 \le R \cdot \|g\|_2^2$
2. *And* $|g(t)|^2 \le \mathsf{poly}(R) \cdot \|g\|_2^2 \cdot |t|^S$ *for* $t \notin [-1,1]$,

Then the filter function $H\big(\alpha x\big)$ *is such that* $H\big(\alpha x\big) \cdot g(x)$ *satisfies*

1. $\int_{-1}^{1} |g(x) \cdot H\big(\alpha x\big)|^2 \mathrm{d}x \ge 0.9 \int_{-1}^{1} |g(x)|^2 \mathrm{d}x$.
2. $\int_{-1}^{1} |g(x) \cdot H\big(\alpha x\big)|^2 \mathrm{d}x \ge 0.95 \int_{-\infty}^{\infty} |g(x) \cdot H\big(\alpha x\big)|^2 \mathrm{d}x$.
3. $|H(x)| \le 1.01$ *for any* x.

Due to the space constraint, we defer the proof of Theorem 9 to the full version.

5 Frequency Estimation

We show the algorithm for frequency estimation and prove Theorem 2 in this section. We fix $T = 1$ and use the definition $\|h\|_2^2 = \mathop{\mathbb{E}}_{x\sim[-1,1]}[|h(x)|^2]$ to restate the theorem.

▶ **Theorem 10.** *Given any* $F > 0, \Delta > 0, R$, *and* $S > 0$, *let* $g(t)$ *be a signal with the following properties:*

1. $\mathsf{supp}(\widehat{g}) \subseteq [f_0 - \Delta, f_0 + \Delta]$ *where* $f_0 \in [-F, F]$.
2. $\sup_{t\in[-1,1]} \big[|g(t)|^2\big] \le R \cdot \|g\|_2^2$.
3. $|g(t)|^2$ *grows as at most* $\mathsf{poly}(R) \cdot \|g\|_2^2 \cdot |t|^S$ *for* $t \notin [-1,1]$.

Let $y(t) = g(t) + \eta(t)$ *be the observable signal on* $[-1,1]$, *where* $\|\eta\|_2^2 \le \epsilon \cdot \|g\|_2^2$ *for a sufficiently small constant* ϵ. *For* $\Delta' = \Delta + \widetilde{O}(R+S)$ *and any* δ, *there exists an efficient algorithm that takes* $O(R\log\frac{F}{\Delta'\cdot\delta})$ *samples from* $y(t)$ *and outputs* $\widetilde{f}$ *satisfying* $|f_0 - \widetilde{f}| \le O(\Delta')$ *with probability at least* $1 - \delta$.

For convenience, we set $h_H(t) = h(t) \cdot H(\alpha t)$ for any signal $h(t)$ with the filter function H defined in Theorem 9 such that $y_H(t) = y(t) \cdot H(\alpha t)$.

Given the observation $y(t)$ with most Fourier mass concentrated around f_0, the main technical result in this section is an estimation of $e^{2\pi \mathbf{i}\beta f_0}$ through $y_H(\alpha)e^{2\pi i f_0 \beta} \approx y_H(\alpha+\beta)$.

▶ **Lemma 11.** *Given parameters* F, R, S, *and* Δ, *let* g *be a signal satisfying the three conditions in Theorem 2 for some* $f_0 \in [-F, F]$ *and* $\Delta' = \Delta + O(R\log R + S\log S)$.

Let $y(t) = g(t) + \eta(t)$ *be the observable signal on* $[-1,1]$ *where the noise* $\|\eta\|_2^2 \le \epsilon\|g\|_2^2$ *for a sufficiently small constant* ϵ. *There exist a constant* γ *and an algorithm such that for any* $\beta \le \frac{\gamma}{\Delta'}$, *it takes* $O(R)$ *samples to output* α *satisfying* $|y_H(\alpha)e^{2\pi i f_0\beta} - y_H(\alpha+\beta)| \le 0.3|y_H(\alpha)|$ *with probability at least 0.6.*

We show our algorithm in Algorithm 1. We finish the proof of Theorem 5 here and defer the proof of Lemma 11 to Section 5.1.

Algorithm 1 Obtain one good α.

procedure ObtainOneGoodSample($R, y(t)$)
  Let $m = C \cdot R$ for a large constant C.
  Take m random samples $x_1, \cdots, x_m$ uniform in $[-1, 1]$.
  Query $y(x_i)$ and compute $y_H(x_i) = y(x_i) \cdot H(x_i)$ for each i.
  Set a distribution D_m proportional to $|y_H(x_i)|^2$, i.e., $D_m(x_i) = \frac{|y_H(x_i)|^2}{\sum_{j=1}^m |y_H(x_j)|^2}$.
  Output $\alpha \sim D_m$.
end procedure

Proof of Theorem 10. From Lemma 11, $\frac{y_H(\alpha+\beta)}{y_H(\alpha)}$ gives a good estimation of $e^{2\pi i f_0 \beta}$ with probability 0.6 for any $\beta \le \frac{\gamma}{\Delta'}$. We use the frequency search algorithm of Lemma 7.3 in [4] with the sampling procedure in Lemma 11. Because the algorithm in [4] uses the sampling procedure $O(\log \frac{F}{\Delta' \cdot \delta})$ times to return a frequency $\widetilde{f}$ satisfying $|\widetilde{f} - f_0| \le \Delta'$ with prob. at least $1 - \delta$, the sample complexity is $O(R \cdot \log \frac{F}{\Delta' \cdot \delta})$. ◀

5.1 Proof of Lemma 11

For $y_H(x) = g_H(x) + \eta_H(x)$, we have the following concentration lemma for estimation $g_H(x)$.

▷ Claim 12. Given any g satisfying the three conditions in Theorem 2 and any ε and δ, there exists $m = O(R \log \frac{1}{\delta}/\varepsilon^2)$ such that for m random samples $x_1, \ldots, x_m \sim [-1, 1]$, with probability $1 - \delta$,

$$\frac{\sum_{i=1}^m |g_H(x_i)|^2}{m} \in [1 - \varepsilon, 1 + \varepsilon] \cdot \mathop{\mathbb{E}}_{x \sim [-1,1]} [|g_H(x)|^2].$$

Proof. Notice that $\frac{\sup_{x \sim [-1,1]} [|g_H(x)|^2]}{\mathop{\mathbb{E}}_{x \sim [-1,1]} [|g_H(x)|^2]} \le 2R$. From the Chernoff bound in Lemma 7, $m = O(R \log \frac{1}{\delta}/\varepsilon^2)$ suffices to estimate $\|g_H\|_2^2$. ◁

Next we consider the effect of noise $\eta_H(x_i)$ and $y_H(x_i)$.

▷ Claim 13. With probability 0.9 over m random samples in $[-1, 1]$, $\sum_{i=1}^m |y_H(x_i)|^2/m \ge 0.8\|g\|_2^2$.

Proof. From Theorem 9, $\|g_H\|_2^2 \ge 0.95\|g\|_2^2$. Thus Claim 12 implies $\sum_{i=1}^m |g_H(x_i)|^2/m \ge 0.98 \cdot 0.95\|g\|_2^2$ for $m = O(R)$ with probability 0.99.

At the same time, because $\mathbb{E}[\sum_{i=1}^m |\eta_H(x_i)|^2/m] = \|\eta_H\|_2^2$, $\sum_{i=1}^m |\eta_H(x_i)|^2/m \le 14\|\eta_H\|_2^2$ with probability at least $1 - \frac{1}{14}$ from the Markov inequality. This is also less than $14 \cdot 1.02^2\|\eta\|_2^2 \le 15\epsilon\|g\|_2^2$ from the upper bound on $H(t)$.

We have

$$\frac{1}{m}\sum_{i=1}^m |y_H(x_i)|^2 \ge \frac{1}{m}\sum_{i=1}^m \Big(|g_H(x_i)|^2 - 2|g_H(x_i)| \cdot |\eta_H(x_i)| + |\eta_H(x_i)|^2\Big).$$

By the Cauchy-Schwartz inequality, the cross term $\sum_{i=1}^m |g_H(x_i)| \cdot |\eta_H(x_i)| \le (\sum_{i=1}^m |g_H(x_i)|^2)^{1/2} \cdot (\sum_{i=1}^m |\eta_H(x_i)|^2)^{1/2}$. From all discussion above,

$$\frac{1}{m}\sum_{i=1}^m |y_H(x_i)|^2 \ge \big(0.93 - 2\sqrt{0.93 \cdot 15\epsilon}\big)\|g\|_2^2.$$

When ε is a small constant, it is at least $0.8 \cdot \|g\|_2^2$. ◁

We set $z(t) = y_H(t) \cdot e^{2\pi \boldsymbol{i} f_0 \beta} - y_H(t+\beta)$ for convenience and bound it as follows.

▷ Claim 14. Given any small constant γ, $\Delta' = \Delta + \mathsf{supp}(H)$, and $z(t) = y_H(t) \cdot e^{2\pi \boldsymbol{i} f_0 \beta} - y_H(t+\beta)$ for $\beta \le \frac{\gamma}{\Delta'}$, $\|z\|_2^2 \lesssim (\gamma^2 + \epsilon)\|g\|_2^2$.

Proof. Notice that $y_H = g_H + \eta_H$ where $\mathsf{supp}(\widehat{g_H}) \in [f_0 - \Delta, f_0 + \Delta]$ such that

$$\int_{f \notin [f_0-\Delta', f_0+\Delta']} |\widehat{y}(f)|^2 \mathrm{d}f \le \int_{-\infty}^{\infty} |\widehat{\eta_H}(f)|^2 \mathrm{d}f = \int_{-\infty}^{\infty} |\eta_H(t)|^2 \mathrm{d}t \le 1.02^2 \epsilon \int_{-1}^{1} |g(t)|^2 \mathrm{d}t.$$

We bound $\|z\|_2^2$ through

$$\begin{aligned}\int_{-1}^{1} |z(t)|^2 \mathrm{d}t \le \int_{-\infty}^{\infty} |z(t)|^2 \mathrm{d}t &= \int_{-\infty}^{\infty} |\widehat{z}(f)|^2 \mathrm{d}f \\ &= \int_{f_0-\Delta'}^{f_0+\Delta'} |\widehat{z}(f)|^2 \mathrm{d}f + \int_{f \notin [f_0-\Delta', f_0+\Delta']} |\widehat{z}(f)|^2 \mathrm{d}f.\end{aligned}$$

Therefore we write

$$\begin{aligned}\int_{f_0-\Delta'}^{f_0+\Delta'} |\widehat{z}(f)|^2 \mathrm{d}f &= \int_{f_0-\Delta'}^{f_0+\Delta'} |\widehat{y_H}(f) \cdot e^{2\pi \boldsymbol{i} f_0 \beta} - \widehat{y_H}(f) \cdot e^{2\pi \boldsymbol{i} f \beta}|^2 \mathrm{d}f \\ &\le \int_{f_0-\Delta'}^{f_0+\Delta'} |\widehat{y_H}(f)|^2 \cdot |e^{2\pi \boldsymbol{i} f_0 \beta} - e^{2\pi \boldsymbol{i} f \beta}|^2 \mathrm{d}f.\end{aligned}$$

Because $f \in [f_0 - \Delta', f_0 + \Delta']$ and $\beta \le \frac{\gamma}{\Delta'}$, $|e^{2\pi \boldsymbol{i} f_0 \beta} - e^{2\pi \boldsymbol{i} f \beta}| \le 4\pi\gamma$. So

$$\int_{f_0-\Delta'}^{f_0+\Delta'} |\widehat{z}(f)|^2 \mathrm{d}f \lesssim \gamma^2 \int_{-\infty}^{+\infty} |\widehat{y_H}(f)|^2 \mathrm{d}f = \gamma^2 \int_{-\infty}^{+\infty} |y_H(t)|^2 \mathrm{d}t \lesssim \gamma^2 (1 + 2\epsilon) \int_{-1}^{1} |g(t)|^2 \mathrm{d}t.$$

On the other hand,

$$\begin{aligned}\int_{f \notin [f_0-\Delta', f_0+\Delta']} |\widehat{z}(f)|^2 \mathrm{d}f &= \int_{f \notin [f_0-\Delta', f_0+\Delta']} |\widehat{y_H}(f) \cdot e^{2\pi \boldsymbol{i} f_0 \beta} - \widehat{y_H}(f) \cdot e^{2\pi \boldsymbol{i} f \beta}|^2 \mathrm{d}f \\ &\le 4 \int_{f \notin [f_0-\Delta', f_0+\Delta']} |\widehat{y_H}(f)|^2 \mathrm{d}f \\ &\le 4 \int_{-\infty}^{+\infty} |\widehat{\eta_H}(f)|^2 \mathrm{d}f = 4 \int_{-\infty}^{+\infty} |\widehat{\eta_H}(t)|^2 \mathrm{d}t\end{aligned}$$

which is less than $5\epsilon \int_{-1}^{1} |g(t)|^2 \mathrm{d}t$.

From all discussion above, $\int_{-1}^{1} |z(t)|^2 \mathrm{d}t \lesssim (\gamma^2 + \epsilon) \int_{-1}^{1} |g(t)|^2 \mathrm{d}t$. ◁

For sufficiently small γ and ε, by Markov inequality, we have the following corollary.

▶ **Corollary 15.** *For sufficiently small constants γ and ϵ, with probability 0.9 over m random samples in $[-1, 1]$, $\sum_{i=1}^{m} |z(x_i)|^2 \le 0.01\|g\|_2^2$.*

Finally we finish the proof of Lemma 11.

Proof of Lemma 11. We assume Claim 13 and Corollary 15 hold in this proof, i.e.,

$$\sum_{i=1}^{m} |y_H(x_i)|^2/m \ge 0.8\|g\|_2^2 \text{ and } \sum_{i=1}^{m} |z(x_i)|^2/m \le 0.01\|g\|_2^2.$$

For a random sample $\alpha \sim D_m$, we bound

$$\mathop{\mathbb{E}}_{\alpha\sim D_m}\left[\frac{|y_H(\alpha)e^{2\pi i f_0\beta} - y_H(\alpha+\beta)|^2}{|y_H(\alpha)|^2}\right] = \mathop{\mathbb{E}}_{\alpha\sim D_m}\left[\frac{|z(\alpha)|^2}{|y_H(\alpha)|^2}\right] = \sum_{i=1}^{m}\frac{|z(x_i)|^2}{|y_H(x_i)|^2}\cdot\frac{|y_H(x_i)|^2}{\sum_{j=1}^m|y_H(x_j)|^2}.$$

This is $\frac{\sum_{i=1}^m|z(x_i)|^2}{\sum_{j=1}^m|y_H(x_j)|^2} \le \frac{0.01}{0.8}$. Thus with probability 0.8, $\frac{|y_H(\alpha)e^{2\pi i f_0\beta}-y_H(\alpha+\beta)|^2}{|y_H(\alpha)|^2}$ is less than $0.05/0.8 \le 0.09$. From all discussion above, $\frac{|y_H(\alpha)e^{2\pi i f_0\beta}-y_H(\alpha+\beta)|}{|y_H(\alpha)|} \le 0.3$ with probability 0.6. ◀

6 Bounds on Fourier-sparse Signals

We consider $g(t) = \sum_{j=1}^k v_j e^{2\pi \boldsymbol{i} f_j t}$ where each $f_j \in [f_0-\Delta, f_0+\Delta]$ in this section. The main result is to prove $R = \tilde{O}(k^3)$ and $S = \tilde{O}(k^2)$ for k arbitrary real frequencies. We restate Theorem 5 after fixing $T = 1$.

▶ **Theorem 16.** *Given F, Δ, and k, let $g(t)$ be a k-Fourier-sparse signal centered around $f_0 \in [-F, F]$: $g(t) = \sum_{i\in[k]} v_i \cdot e^{2\pi \mathbf{i} f_i t}$ where $f_i \in [f_0-\Delta, f_0+\Delta]$ and $y(t) = g(t) + \eta(t)$ be the observable signal on $[-1, 1]$, where $\|\eta\|_2^2 \le \epsilon \cdot \|g\|_2^2$ for a sufficiently small constant ϵ.*

For any $\delta > 0$, there exist $\Delta' = \Delta + \tilde{O}(R)$ and an efficient algorithm that takes $O(k\log^2 k \log\frac{F}{\Delta'\cdot\delta})$ samples from $y(t)$ and outputs $\widetilde{f}$ satisfying $|f_0 - \widetilde{f}| \le O(\Delta')$ with probability at least $1-\delta$.

The main improvement is a biased distribution that saves the sample complexity from $O(R)\cdot\log\frac{F}{\Delta'\cdot\delta}$ to $\widetilde{O}(k)\cdot\log\frac{F}{\Delta'\cdot\delta}$.

We provide the main technical lemma here and defer the proofs of Theorem 3, 4, and 16 to the full version.

▶ **Theorem 17.** *Given $z_1, \ldots, z_k$ with $|z_1| = |z_2| = \cdots = |z_k| = 1$, there exists a degree $d = O(k^2\log k)$ polynomial $P(z) = \sum_{j=0}^d c(j)\cdot z^j$ satisfying*

1. *$P(z_i) = 0$ for each $i \in [k]$.*
2. *Coefficients $c(0) = \Omega(1)$, $c(j) = O(1)$ and $\sum_{j=1}^d |c(j)|^2 = O(k)\cdot|c(0)|^2$.*

▶ **Corollary 18.** *Given any $g(t) = \sum_{j=1}^k v_j e^{2\pi \boldsymbol{i} f_j t}$ and $\theta > 0$, there exist $d = O(k^2\log k)$ and a sequence of coefficients $(\alpha_1, \ldots, \alpha_d)$ such that*

1. *$\alpha_j = O(1)$ for any $j = 1, \ldots, d$.*
2. *For any x (not necessarily in $[-1, 1]$), $g(x) = \sum_{j=1}^d \alpha_j \cdot g(x - j\theta)$.*

Proof. Given θ, we set $z_i = e^{-2\pi \boldsymbol{i} f_j\theta}$ and apply Theorem 17 to obtain coefficients $c(0), \ldots, c(d)$. Then we set $\alpha_j = -c(j)/c(0)$. It is straightforward to verify the second property because of

$$e^{2\pi \boldsymbol{i} f_j x} - \sum_j \alpha_j \cdot e^{2\pi \boldsymbol{i} f_j(x-j\theta)} = 0.$$ ◀

The proof of Theorem 17 requires the following bound on the coefficients of residual polynomials, which is stated as Lemma 5.3 in [4].

▶ **Lemma 19.** *Given $z_1, \ldots, z_k$, for any integer n, let $r_{n,k}(z) = \sum_{i=0}^{k-1} r_{n,k}^{(i)}\cdot z^i$ denote the residual polynomial of $r_{n,k} \equiv z^n \mod \prod_{j=1}^k(z-z_j)$. Then each coefficient in $r_{n,k}$ is bounded: $|r_{n,k}^{(i)}| \le \binom{k-1}{i}\cdot\binom{n}{k-1}$ for $n \ge k$ and $|r_{n,k}^{(i)}| \le \binom{k-1}{i}\cdot\binom{|n|+k-1}{k-1}$ for $n < 0$.*

We finish the proof of Theorem 17 here.

Proof. Let C_0 be a large constant and $d = 5 \cdot k^2 \log k$. We use $\mathcal{P}$ to denote the following subset of polynomials with bounded coefficients:

$$\left\{ \sum_{j=0}^{d} \alpha_j \cdot 2^{-j/k} \cdot z^j \middle| \alpha_0, \ldots, \alpha_d \in [-C_0, C_0] \cap \mathbb{Z} \right\}.$$

For each polynomial $P(z) \in \mathcal{P}$, we rewrite $P(z) \mod \prod_{j=1}^{k}(z - z_j)$ as

$$\sum_{j=0}^{d} \alpha_j \cdot 2^{-j/k} \cdot \left(z^j \mod \prod_{j=1}^{k}(z - z_j) \right) = \sum_{i=0}^{k-1} \left(\sum_{j=0}^{d} \alpha_j \cdot 2^{-j/k} \cdot r_{n,k}^{(i)} \right) z^i.$$

The coefficient $\sum_{j=0}^{d} \alpha_j \cdot 2^{-j/k} \cdot r_{n,k}^{(i)}$ is bounded by

$$\sum_{j=0}^{d} C_0 \cdot 2^{-j/k} \cdot 2^k j^{k-1} \leq d \cdot C_0 \cdot 2^k \cdot d^k \leq d^{2k}.$$

Then we apply the pigeonhole principle on the $(2C_0 + 1)^d$ polynomials in $\mathcal{P}$ after module $\prod_{j=1}^{d}(z - z_j)$: there exist $m > (2C_0 + 1)^{0.9d}$ polynomials $P_1, \ldots, P_m$ such that each coefficient of $(P_i - P_j) \mod \prod_{j=1}^{k}(z - z_j)$ is d^{-2k} small from the counting

$$\frac{(2C_0 + 1)^d}{(d^{2k}/4d^{-2k})^k} > (2C_0 + 1)^{0.9d}.$$

Because $m > (2C_0 + 1)^{0.9d}$, there exists $j_1 \in [m]$ and $j_2 \in [m] \setminus \{j_1\}$ such that the lowest monomial z^l with different coefficients in P_{j_1} and P_{j_2} satisfies $l \leq 0.1d$. Eventually we set

$$P(z) = z^{-l} \cdot \big(P_{j_1}(z) - P_{j_2}(z)\big) - \left(z^{-l} \mod \prod_{j=1}^{k}(z - z_j) \right) \cdot \left(P_{j_1}(z) - P_{j_2}(z) \mod \prod_{j=1}^{k}(z - z_j) \right)$$

to satisfy the first property $P(z_1) = P(z_2) = \cdots = P(z_k) = 0$. We prove the second property in the rest of this proof.

We bound every coefficient in $\big(z^{-l} \mod \prod_{j=1}^{k}(z - z_j)\big) \cdot \big(P_{j_1}(z) - P_{j_2}(z) \mod \prod_{j=1}^{k}(z - z_j)\big)$ by

$$k \cdot \text{max-coefficient}\left(z^{-l} \mod \prod_{j=1}^{k}(z - z_j) \right) \cdot \text{max-coefficient}\left(P_{j_1}(z) - P_{j_2}(z) \mod \prod_{j=1}^{k}(z - z_j) \right),$$

which is less than $k \cdot 2^k (l + k)^{k-1} \cdot d^{-2k} \leq d \cdot 2^k d^{k-1} \cdot d^{-2k} \leq d^{-0.5k}$ from Lemma 19 and the above discussion.

On the other hand, the constant coefficient in $z^{-l} \cdot \big(P_{j_1}(z) - P_{j_2}(z)\big)$ is at least $2^{-l/k} \geq 2^{-0.1d/k} = k^{-0.5k}$ because z^l is the smallest monomial with different coefficients in P_{j_1} and P_{j_2} from $\mathcal{P}$. Thus the constant coefficient $|C(0)|^2$ of $P(z)$ is at least $0.5 \cdot 2^{-2l/k}$.

Next we upper bound the sum of the rest of the coefficients $\sum_{j=1}^{d} |C(j)|^2$ by

$$\sum_{j=1}^{d} (2C_0 \cdot 2^{-(l+j)/k} + d^{-0.5k})^2 \leq 2 \cdot 4C_0^2 \sum_{j=1}^{d} 2^{-2(l+j)/k} + 2 \cdot \sum_{j=1}^{d} d^{-0.5k \cdot 2} \lesssim k \cdot 2^{-2l/k},$$

which demonstrates the second property after normalizing $C(0)$ to 1. ◀

References

1. A. Akavia, S. Goldwasser, and S. Safra. Proving hard-core predicates using list decoding. *FOCS*, 44:146–159, 2003.
2. Haim Avron, Michael Kapralov, Cameron Musco, Christopher Musco, Ameya Velingker, and Amir Zandieh. A Universal Sampling Method for Reconstructing Signals with Simple Fourier Transforms. In *Proceedings of the 51st annual ACM symposium on Theory of computing (STOC 2019)*, 2019. arXiv:1812.08723.
3. Y. Bresler and A. Macovski. Exact maximum likelihood parameter estimation of superimposed exponential signals in noise. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(5):1081–1089, October 1986. doi:10.1109/TASSP.1986.1164949.
4. Xue Chen, Daniel M. Kane, Eric Price, and Zhao Song. Fourier-sparse interpolation without a frequency gap. In *Foundations of Computer Science(FOCS), 2016 IEEE 57th Annual Symposium on*, 2016. arXiv:1609.01361.
5. Xue Chen and Eric Price. Active Regression via Linear-Sample Sparsification. In *the 32nd Annual Conference on Learning Theory (COLT 2019)*, 2019.
6. Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23:493–507, 1952.
7. Anna C Gilbert, Sudipto Guha, Piotr Indyk, S Muthukrishnan, and Martin Strauss. Near-optimal sparse Fourier representations via sampling. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pages 152–161. ACM, 2002.
8. Anna C Gilbert, S Muthukrishnan, and Martin Strauss. Improved time bounds for near-optimal sparse Fourier representations. In *Optics & Photonics 2005*, pages 59141A–59141A. International Society for Optics and Photonics, 2005.
9. Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Simple and practical algorithm for sparse Fourier transform. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1183–1194. SIAM, 2012.
10. Piotr Indyk and Michael Kapralov. Sample-Optimal Fourier Sampling in Any Constant Dimension. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 514–523. IEEE, 2014.
11. Y. Mansour. Randomized Interpolation and Approximation of Sparse Polynomials. *ICALP*, 1992.
12. Ankur Moitra. The threshold for super-resolution via extremal functions. In *STOC*, 2015.
13. Eric Price and Zhao Song. A Robust Sparse Fourier Transform in the Continuous Setting. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 583–600. IEEE, 2015.
14. R Prony. Essai experimental et analytique. *J. de l'Ecole Polytechnique*, 1795.

Block Edit Errors with Transpositions: Deterministic Document Exchange Protocols and Almost Optimal Binary Codes

Kuan Cheng
Department of Computer Science, Johns Hopkins University, USA
kcheng17@jhu.edu

Zhengzhong Jin
Department of Computer Science, Johns Hopkins University, USA
zjin12@jhu.edu

Xin Li
Department of Computer Science, Johns Hopkins University, USA
lixints@cs.jhu.edu

Ke Wu
Department of Computer Science, Johns Hopkins University, USA
ashleymo@jhu.edu

Abstract

Document exchange and error correcting codes are two fundamental problems regarding communications. In the first problem, Alice and Bob each holds a string, and the goal is for Alice to send a short sketch to Bob, so that Bob can recover Alice's string. In the second problem, Alice sends a message with some redundant information to Bob through a channel that can add adversarial errors, and the goal is for Bob to correctly recover the message despite the errors. In both problems, an upper bound is placed on the number of errors between the two strings or that the channel can add, and a major goal is to minimize the size of the sketch or the redundant information. In this paper we focus on deterministic document exchange protocols and binary error correcting codes.

Both problems have been studied extensively. In the case of Hamming errors (i.e., bit substitutions) and bit erasures, we have explicit constructions with asymptotically optimal parameters. However, other error types are still rather poorly understood. In a recent work [7], the authors constructed explicit deterministic document exchange protocols and binary error correcting codes for edit errors with almost optimal parameters. Unfortunately, the constructions in [7] do not work for other common errors such as block transpositions.

In this paper, we generalize the constructions in [7] to handle a much larger class of errors. These include bursts of insertions and deletions, as well as block transpositions. Specifically, we consider document exchange and error correcting codes where the total number of block insertions, block deletions, and block transpositions is at most $k \leq \alpha n/\log n$ for some constant $0 < \alpha < 1$. In addition, the total number of bits inserted and deleted by the first two kinds of operations is at most $t \leq \beta n$ for some constant $0 < \beta < 1$, where n is the length of Alice's string or message. We construct explicit, deterministic document exchange protocols with sketch size $O((k\log n + t)\log^2 \frac{n}{k\log n + t})$ and explicit binary error correcting code with $O(k \log n \log\log\log n + t)$ redundant bits. As a comparison, the information-theoretic optimum for both problems is $\Theta(k\log n + t)$. As far as we know, previously there are no known explicit deterministic document exchange protocols in this case, and the best known binary code needs $\Omega(n)$ redundant bits even to correct just *one* block transposition [23].[1]

2012 ACM Subject Classification Mathematics of computing → Coding theory

[1] We note that by combining the techniques in [14] and [15], one can get an explicit binary code that corrects k block transpositions with $\widetilde{O}(\sqrt{kn})$ redundant bits. However to our knowledge this result has not appeared anywhere in the literature, and moreover it requires at least $\widetilde{\Omega}(\sqrt{n})$ redundant bits even to correct one block transposition.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 37; pp. 37:1–37:15

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

Keywords and phrases Deterministic document exchange, error correcting code, block edit error

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.37

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1809.00725`.

Acknowledgements We thank an anonymous referee for catching an error in the previous version of this paper, and Bernhard Haeupler for very useful feedbacks.

1 Introduction

In communications and more generally distributed computing environments, there often arise questions regarding the synchronization of files or messages. For example, a message sent from one party to another party through a channel may get modified by channel noise or adversarial errors, and files stored on distributed servers may become out of sync due to different edit operations by different users. In many situations, these questions can be formalized in the framework of the following two fundamental problems.

- *Document exchange.* In this problem, two parties Alice and Bob each holds a string x and y, and the two strings are within distance k in some metric space. The goal is for Alice to send a short sketch to Bob, so that Bob can recover x based on his string y and the sketch.
- *Error correcting codes.* In this problem, two parties Alice and Bob are linked by a channel, which can change any string sent into another string within distance k in some metric space. Alice's goal is to send a message to Bob. She does this by sending an encoding of the message through the channel, which contains some redundant information, so that Bob can recover the correct message despite any changes to the codeword.

These two problems are closely related. For example, in many cases a solution to the document exchange problem can also be used to construct an error correcting code, but the reverse direction is not necessarily true. In both problems, a major goal is to is to minimize the size of the sketch or the redundant information. For applications in computer science, we also require the computations of both parties to be efficient, i.e., in polynomial time of the input length. In this case we say that the solutions to these problems are *explicit*. Here we focus on deterministic document exchange protocols and error correcting codes with a binary alphabet, arguably the most important setting in computer science.

Both problems have been studied extensively, but the known solutions and our knowledge vary significantly depending on the distance metric in these problems. In the case of Hamming distance (or Hamming errors), we have a near complete understanding and explicit constructions with asymptotically optimal parameters. However, for other distance metrics/error types, our understanding is still rather limited.

An important generalization of Hamming errors is edit errors, which consist of bit insertions and deletions. These are strictly more general than Hamming errors since a bit substitution can be replaced by a deletion followed by an insertion. Edit errors can happen in many practical situations, such as reading magnetic and optical media, mutations in gene sequences, and routing packets in Internet protocols. However, these errors are considerably harder to handle, due to the fact that a single edit error can change the positions of all the bits in a string.

Non-explicitly, by using a greedy graph coloring algorithm or a sphere packing argument, one can show that the optimal size of the sketch in document exchange, or the redundant information in error correcting codes is roughly the same for both Hamming errors and edit errors. Specifically, suppose that Alice's string or message has length n and the distance bound k is relatively small (e.g., $k \leq n/4$), then for both Hamming errors and edit errors, the optimal size in both problems is $\Theta(k \log(\frac{n}{k}))$ [19]. For Hamming errors, this can be achieved by using sophisticated linear Algebraic Geometric codes [16], but for edit errors the situation is quite different. We now describe some of the previous works regarding both document exchange and error correcting codes for edit errors.

Document exchange

Orlitsky [21] first studied the document exchange problem for generally correlated strings x, y. Using the greedy graph coloring algorithm mentioned before, he obtained a deterministic protocol with sketch size $O(k \log n)$ for edit errors, but the running time is exponential in k. Subsequent improvements appeared in [9], [17], and [18], achieving sketch size $O(k \log(\frac{n}{k}) \log n)$ [17] and $O(k \log^2 n \log^* n)$ [18] with running time $\tilde{O}(n)$. A recent work by Chakraborty et al. [5] further obtained sketch size $O(k^2 \log n)$ and running time $\tilde{O}(n)$, by using a clever randomized embedding from the edit distance metric to the Hamming distance metric. Based on this work, Belazzougui and Zhang [2] gave an improved protocol with sketch size $O(k(\log^2 k + \log n))$, which is asymptotically optimal for $k = 2^{O(\sqrt{\log n})}$. The running time in [2] is $\tilde{O}(n + \mathsf{poly}(k))$.

Unfortunately, all of the above protocols, except the one in [21] which runs in exponential time, are randomized. Although randomized protocols are still useful in practice, having deterministic ones would certainly bring much more benefits. Furthermore, randomized protocols are also not suitable for the applications in constructing error correcting codes. However, designing an efficient deterministic protocol appears quite tricky, and it was not until 2015 when Belazzougui [1] gave the first deterministic protocol even for $k > 1$. The protocol in [1] has sketch size $O(k^2 + k \log^2 n)$ and running time $\tilde{O}(n)$.

Error correcting codes

As fundamental objects in both theory and practice, error correcting codes have been studied extensively from the pioneering work of Shannon and Hamming. While great success has been achieved in constructing codes for Hamming errors, the progress on codes for edit errors has been quite slow despite much research. A work by Levenshtein [19] in 1966 showed that the Varshamov-Tenengolts code [22] corrects one deletion with an optimal redundancy of roughly $\log n$ bits, but even correcting two deletions requires $\Omega(n)$ redundant bits. In 1999, Schulman and Zuckerman [23] gave an explicit asymptotically good code, that can correct up to $\Omega(n)$ edit errors with $O(n)$ redundant bits. However the same amount of redundancy is needed even for smaller number of errors. For more earlier works on this subject, we refer the reader to the survey by Mercier et al. [20].

In recent years there have been several works trying to improve the situation. Specifically, a line of work by Guruswami et. al [11], [10], [4] constructed explicit codes that can correct $1 - \varepsilon$ fraction of edit errors with rate $\Omega(\varepsilon^5)$ and alphabet size $\mathsf{poly}(1/\varepsilon)$; and codes that can correct $1 - \frac{2}{t+1} - \varepsilon$ fraction of errors with rate $(\varepsilon/t)^{\mathsf{poly}(1/\varepsilon)}$ for a fixed alphabet size $t \geq 2$. Another line of work by Haeupler et al. [13], [14], [6] introduced and constructed a combinatorial object called *synchronization string*, which can be used to transform standard error correcting codes into codes for edit errors by increasing the alphabet size. Via this

transformation, [13] achieved explicit codes that can correct δ fraction of edit errors with rate $1-\delta-\varepsilon$ and alphabet size exponential in $\frac{1}{\varepsilon}$, which approaches the singleton bound. All of these works however require a relatively large alphabet size.

In the case of binary alphabets, for any fixed constant k, a recent work by Brakensiek et. al [3] constructed an explicit code that can correct k edit errors with $O(k^2 \log k \log n)$ redundant bits. This is asymptotically optimal when k is a fixed constant, but the construction in [3] only works for constant k, and breaks down for larger k (e.g., $k = \log n$). Based on his deterministic document exchange protocol, Belazzougui [1] also gave an explicit code that can correct up to k edit errors with $O(k^2 + k \log^2 n)$ redundant bits. Finally, the work by Haeupler et. al [15] constructed explicit codes that can correct δ fraction of edit errors with rate $1 - \Theta(\sqrt{\delta \log(1/\delta)})$, whereas the (non-explicit) optimal rate is $1 - \Theta(\delta \log(1/\delta))$.

In a very recent work by the authors [7], we significantly improved the situation. Specifically, we constructed an explicit document exchange protocol with sketch size $O(k \log^2 \frac{n}{k})$, which is optimal except for an additional $\log \frac{n}{k}$ factor. This also implies an explicit binary code that can correct δ fraction of edit errors with rate $1 - \Theta(\delta \log^2(1/\delta))$, which is optimal up to an additional $\log(1/\delta)$ factor. These two results are also independently obtained by Haeupler [12]. We also constructed explicit codes for k edit errors with $O(k \log n)$ redundant bits, which is optimal for $k \leq n^{1-\alpha}$, any constant $0 < \alpha < 1$. These results bring our understanding of document exchange and error correcting codes for edit errors much closer to that of standard Hamming errors.

However, the constructions in [7] and [12] do not work for other common types of errors, such as *block transpositions*. Given any string x, a block transposition takes an arbitrary substring z of x, cuts it to make x become $\tilde{x}$, and then finds a different position in $\tilde{x}$ and insert z as a block into $\tilde{x}$. These errors happen frequently in distributed file systems and Internet protocols. For example, it is quite common that a user, when editing a file, moves a whole paragraph in the file to somewhere else; and in Internet routing protocols, packets can often get rearranged during the process. Block transpositions also arise naturally in biological processes, where a subsequence of genes can be moved in one step during mutation. In the setting of document exchange or error correcting codes, it is easy to see that even a single transposition of a block with length t can result in $2t$ edit errors, thus a naive application of document exchange protocols or codes for edit errors will result in very bad parameters.

Model of the adversary: block insertions, deletions, and transpositions

We consider edit errors that happen in *bursts*. This kind of errors is also pretty common, as most errors that happen in practice, such as in wireless or mobile communications and magnetic disk readings, tend to be concentrated. We model such errors as *block* insertions and deletions, where in one operation the adversary can insert or delete a whole block of bits. It is again easy to see that this is indeed a generalization of standard edit errors.

For some parameters k and t and an alphabet Σ, a (k, t) block edit adversary is allowed to perform three kinds of operations: block insertion, block deletion and block transposition. The adversary is allowed to perform at most k such operations, while the total number of symbols inserted/deleted by the first two operations is at most t. We also use (k, t) block edit errors to denote errors introduced by such an adversary. All our results focus on the case of binary alphabet, but in our protocols and analysis we will be using larger alphabets.

We note that by the result of Schulman and Zuckerman [23], to correct $\Omega(n/\log n)$ block transpositions one needs at least $\Omega(n)$ redundant bits. Thus we only consider $k \leq \alpha n/\log n$ for some constant $0 < \alpha < 1$. Similarly, we only consider $t \leq \beta n$ for some constant $0 < \beta < 1$ since otherwise the adversary can simply delete the whole string. We also note the following

subtle difference between the three block edit operations. While we need a bound t on the total number of bits that the adversary can insert or delete, for block transposition an adversary can choose to move an *arbitrarily long* substring. Therefore, we need to consider the three operations separately, and cannot simply replace a block transposition by a block deletion followed by a block insertion.

Edit errors with block transpositions have been studied before in several different contexts. For example, Shapira and Storer [24] showed that finding the distance between two given strings under this metric is NP-hard, and they gave an efficient algorithm that achieves $O(\log n)$ approximation. Interestingly, a work by Cormode and Muthukrishnan [8] showed that this metric can be embedded into the L_1 metric with distortion $O(\log n \log^* n)$; and they used it to give a near linear time algorithm that achieves $O(\log n \log^* n)$ approximation for this distance, something currently unknown for the standard edit distance. Coming back to document exchange and error correcting codes, in our model, we show in the appendix that non-explicitly, the information optimum for both the sketch size of document exchange, and the redundancy of error correcting codes, is $\Theta(k \log n + t)$.

Related previous work on block transpositions

When it comes to more general errors such as block transpositions, as far as we know, there are no known explicit deterministic document exchange protocols. The only known randomized protocols which can handle edit errors as well as block transpositions are the protocol of [17], which has sketch size $O(k \log(\frac{n}{k}) \log n)$; and the protocol of [18], which has sketch size $\tilde{O}(k \log^2 n)$. The protocol of [17] uses a recursive tree structure and random hash functions, while the protocol of [18] is based on the embedding of Cormode and Muthukrishnan [8]. We stress that both of these protocols are randomized, and there are very good reasons why it is not easy to modify them into deterministic ones. Specifically, unlike in our previous work [7] and the work of Haeupler [12], a direct derandomization of the hash functions used in [17] (for example by using almost k-wise independent sample space) does *not* give a deterministic protocol, because block transpositions will make the computation of a matching problematic. We shall discuss this in more details when we give an overview of our techniques. On the other hand, the embedding of Cormode and Muthukrishnan [8] results in an exponentially large dimension, thus directly sending a sketch deterministically will result in a prohibitively large size. This is why the protocol of [18] has to perform a dimension reduction first, which is necessarily randomized.

Similarly, the only previous explicit codes that can handle edit errors as well as block transpositions are the work of Schulman and Zuckerman [23], and the work of Haeupler et al. [14]. Both can recover from $\Omega(n/\log n)$ block transpositions with $\Omega(n)$ redundant bits ([14] can also recover from block replications), but [23] has a binary alphabet while [14] has a constant size alphabet. However the work of Schulman and Zuckerman [23] also needs $\Omega(n)$ redundant bits even to correct one block transposition. We further note that by combining the techniques in [14] and [15], one can get an explicit binary code that corrects k block transpositions with $\widetilde{O}(\sqrt{kn})$ redundant bits. However to our knowledge this result has not appeared anywhere in the literature, and moreover it requires at least $\widetilde{\Omega}(\sqrt{n})$ redundant bits even to correct one block transposition. We note that however none of the previous works mentioned studied edit errors that can allow block insertions/deletions.

1.1 Our results

In this paper we construct explicit deterministic document exchange protocols, and error correcting codes for adversaries discussed above. We have the following theorems.

▶ **Theorem 1.** *There exist constants $\alpha, \beta \in (0,1)$ such that for every $n, k, t \in \mathbb{N}$ with $k \leq \alpha n/\log n, t \leq \beta n$, there exists an explicit binary document exchange protocol with sketch size $O((k\log n + t)\log^2 \frac{n}{k\log n+t})$, against a (k,t) block edit adversary.*

This is the first explicit, deterministic document exchange protocol for block edit errors. The sketch size matches the randomized protocols of [17] and [18] up to an additional $\log \frac{n}{k\log n+t}$ factor, and is optimal up to an additional $\log^2 \frac{n}{k\log n+t}$ factor. Using this protocol, we can construct the following error correcting code.

▶ **Theorem 2.** *There exist constants $\alpha, \beta \in (0,1)$ such that for every $n, k, t \in \mathbb{N}$ with $k \leq \alpha n/\log n, t \leq \beta n$, there exists an explicit binary error correcting code with message length n and codeword length $n + O((k\log n + t)\log^2 \frac{n}{k\log n+t})$, against a (k,t) block edit adversary.*

For small k, t we can actually achieve the following result, which gives better parameters.

▶ **Theorem 3.** *There exist constants $\alpha, \beta \in (0,1)$ such that for every $n, k, t \in \mathbb{N}$ with $k \leq \alpha n/\log n, t \leq \beta n$, there exists an explicit binary code with message length n and codeword length $n + O(k\log n \log\log\log n + t)$, against a (k,t) block edit adversary.*

In the case of small k, t, these results significantly improve the result of Schulman and Zuckerman [23], which needs $\Omega(n)$ redundant bits even to correct one block transposition, and the result obtained by combining the techniques in [14] and [15], which needs $\widetilde{\Omega}(\sqrt{n})$ redundant bits even to correct one block transposition. The redundancy here is also optimal up to an extra $\log\log\log n$ factor or $\log^2 \frac{n}{k\log n+t}$ factor.

As a special case, we obtain the following corollaries for standard edit errors with block transpositions.

▶ **Corollary 4.** *There exist a constant $\alpha \in (0,1)$ such that for every $n, k \in \mathbb{N}$ with $k \leq \alpha n/\log n$, there exists an explicit binary document exchange protocol with sketch size $O(k\log n \log^2 \frac{n}{k\log n})$, against an adversary who can perform k edit operations or block transpositions.*

▶ **Corollary 5.** *There exist a constant $\alpha \in (0,1)$ such that for every $n, k \in \mathbb{N}$ with $k \leq \alpha n/\log n$, there exists an explicit binary error correcting code with message length n and codeword length $\min\{n + O(k\log n \log^2 \frac{n}{k\log n}), n + O(k\log n \log\log\log n)\}$, against an adversary who can perform k edit operations or block transpositions.*

▶ Remark 6. As illustrated by our theorems and corollaries, the sketch size in our document exchange protocol or the number of redundant bits in our error correcting codes do *not* depend on the size of a block in block transpositions, they only depend on the number of such operations performed. In contrast, the sketch size or the number of redundant bits do depend on the size of a block in block insertions or deletions. This again shows that we cannot simply treat a block transposition as a block deletion followed by a block insertion, because that will lead to a sketch size dependent on the block size.

2 Document Exchange

This section describes the construction of our document exchange protocol and its proof in sketch. Details are deferred to the appendix.

▶ **Definition 7** (Collision free hash functions). *Given $n, p, q \in \mathbb{N}, p \leq n$ and a string $x \in \{0,1\}^n$, we say a hash function $h : \{0,1\}^p \to \{0,1\}^q$ is collision free (for x), if for every $i, j \in [n-p+1]$, $h(x[i, i+p)) = h(x[j, j+p))$ if and only if $x[i, i+p) = x[j, j+p)$.*

▶ **Theorem 8.** *There exists an algorithm which, on input $n, p, q \in \mathbb{N}, p \leq n, q = c_0 \log n$ for large enough constant c_0, $x \in \{0,1\}^n$, outputs a description of a hash function $h : \{0,1\}^p \to \{0,1\}^q$ that is collision free for x, in time $\mathsf{poly}(n)$, where the description length is $O(\log n)$.*

Also there is an algorithm which, given the description of h and any $u \in \{0,1\}^p$, can output $h(u)$ in time $\mathsf{poly}(n)$.

Proof Sketch. The construction uses almost κ-wise independence generator to get the hash functions, assuring no pair of blocks of x collides. The number of pairs is $O(n^2)$. The seed length is $O(\log n)$. So we can do an exhaustive search to find such sequence of collision free hash functions. For details, see the full version. ◀

▶ **Definition 9** (Matching). *Given $n, n', p, q \in \mathbb{N}, p \leq n, p \leq n'$, a function $h : \{0,1\}^p \to \{0,1\}^q$ and two strings $x \in \{0,1\}^n, y \in \{0,1\}^{n'}$, a matching (may not be monotone) between x and y under h is a sequence of matches (pairs of indices) $w = ((i_1, j_1), \ldots (i_{|w|}, j_{|w|}))$ s.t.*

- *for every $k \in [|w|]$,*
  - *$i_k = 1 + pl_k \in [n]$ for some l_k,*
  - *$j_k \in [n']$,*
  - *$h(x[i_k, i_k + p)) = h(y[j_k, j_k + p))$,*
- *$i_1, \ldots, i_{|w|}$ are distinct.*

A non-overlapping matching is a matching with one more restriction.

- *Intervals $[j_k, j_k + p), k \in [|w|]$, are disjoint.*

When considering overlaps, the matching has overlapping degree d, if each bit of y appears in at most d matched pairs for some small number d.

For a match (i, j), it matches two intervals, one from x, the other from y. When we say the y's interval (of the match (i, j)), we mean $[j, j + p)$, and similarly the x's interval is $[i, i + p)$. A match (i, j) in a matching is called a wrong match (or wrong pair) if $x[i, i+p) \neq y[j, j+p)$. Otherwise it is called a correct match (or correct pair). A pair of indices (i, j) is called a potential match between x and y if $h(x[i, i + p)) = h(y[j, j + p))$. It may be wrong because $x[i, i + p)$ may not be $y[j, j + p)$. When x, y are clear from the context we simply say (i, j) is a potential match.

To compute a monotone non-overlapping matching we can use the dynamic programming method in [7]. But our matching is not necessarily monotone. So this raises the question of how hard this problem is.

It seems difficult to find a polynomial algorithm which can exactly compute it. So instead we use constant approximation techniques. There're two difficulties at the first thought. One is that if we compute the non-overlapping matching over the entire strings, then a constant approximation is too bad since there will be $O(n)$ unmatched blocks. So for each level, we restrict our attention to blocks that are uncovered and wrongly recovered (but discovered by us). The other problem is that we need the approximation rate to be a large enough constant. To achieve this goal, we actually computing matchings with constant degree.

We start from a 1/3-approximation algorithm, which is greedy.

▶ **Construction 10.** *Given $n, n', p, q \in \mathbb{N}, p \leq n, p \leq n'$, a polynomial time computable function $h : \{0,1\}^p \to \{0,1\}^q$ and two strings $x \in \{0,1\}^n, y \in \{0,1\}^{n'}$, we have the following 1/3-approximation algorithm for computing the non-overlapping matching.*

1. *Let the sequence of matches w be empty;*
2. *Find $i = 1 + pl \in [n]$ and $j \in [n']$, where $l \in \mathbb{N}$, s.t.*
   - *$h(x[i, i+p)) = h([j, j+p))$,*
   - *i is not in any match (as the first entry) of the current w,*
   - *$[j, j+p)$ does not overlap with any $[j', j'+p)$ for any j' as the second entry in any matches of the current w;*
3. *If there is such a pair of indices i, j, then add the match (i, j) to w and go to step 2; Otherwise, output w and stop.*

▶ **Lemma 11.** *Construction 10 gives a 1/3-approximation algorithm for computing the non-overlapping matching.*

Proof deferred to the full version.

Next we give an explicit algorithm which computes a even larger matching (better approximation), but it allows overlaps.

▶ **Construction 12.** *Given $n, n', p, q \in \mathbb{N}, p \leq n, p \leq n'$, a (polynomial time computable) function $h : \{0,1\}^p \to \{0,1\}^q$ and two strings $x \in \{0,1\}^n, y \in \{0,1\}^{n'}$, we have the following algorithm.*

1. *Let the matching w be empty, set $S = \{i = 1 + pl \mid l \in \mathbb{N}, i \in [n]\}$, integer $c = 0$;*
2. *Conduct Construction 10 to compute a matching w' between x_S and y under h. Here x_S is the projection of x on intervals in set S;*
3. *Let $w = w \cup w'$;*
4. *Let $S = S \setminus \{u \mid \exists (u, v) \in w\}$;*
5. *$c = c + 1$;*
6. *If $c \geq 3$, output w; Otherwise go to step 2.*

Note that Construction 12 is in polynomial time since it simply conducts Construction 10 for 3 times and after each conduction it removes matched blocks of x and only considers the remaining blocks in the next iteration. So we only need to show its correctness.

▶ **Lemma 13.** *Construction 12 computes a degree 3 overlapping matching w between x and y under h, such that $|w| \geq 2/3|w^*|$, where w^* is the maximum non-overlapping matching between x and y under h.*

Proof. Let $w_i, i = 1, 2, 3$ be the matching the algorithm computes after round i. Also let $S_i, i = 1, 2, 3$ be the set S after the ith round.

By Lemma 11, $|w_1| \geq 1/3|w^*|$. The number of unmatched blocks is $\bar{n} - |w_1| \leq \bar{n} - 1/3|w^*|$, where $\bar{n} = \lfloor n/p \rfloor$ is the total number of blocks of x.

The maximum matching between x_{S_1} and y is at least $|w^*| - |w_1|$. This is because that, each of the matched blocks of x by w_1, should be among the x's blocks in the matches of w^*. There are at most $|w_1|$ of them. So there are still $|w^*| - |w_1|$ remaining matches in w^* which corresponds to blocks in x_{S_1}.

Again by Lemma 11, for $i \geq 2$, at least $1/3(|w^*| - |w_{i-1}|)$ blocks of $x_{S_{i-1}}$ will be matched in the ith round.

Thus

$$\begin{aligned}
|w_i| &\geq |w_{i-1}| + 1/3(|w^*| - |w_{i-1}|) && (1)\\
&= 1/3|w^*| + 2/3|w_{i-1}| && (2)\\
&\geq (1-(2/3)^{i-1})|w^*| + (2/3)^{i-1}|w_1| && (3)\\
&\geq (1-(2/3)^{i-1})|w^*| + (1/3)(2/3)^{i-1}|w^*| && (4)\\
&= (1-(2/3)^{i})|w^*|. && (5)
\end{aligned}$$

Inequality 1 is due to Lemma 11 as explained above. Equality 2 is due to a direct computation. 3 is by recursively applying 1 and 2 from $i-1$ to 2. 4 is because $|w_1| \geq 1/3|w^*|$.

As a result, $|w_3| \geq 19/27|w^*| \geq 2/3|w^*|$.

Note that we apply Construction 10 for 3 times, where in each time, it gives a non-overlapping matching. So each entry of y is in at most one of the matches in that round. So finally we get a degree 3 overlapping matching. ◀

We now give the following document exchange protocol.

▶ **Construction 14.** *The protocol works for every input length $n \in \mathbb{N}$, every (k_1, t) block-insertions/deletions k_2 block-transpositions, $k_1, k_2 \leq \alpha n/\log n, t \leq \beta n$, for some constant α, β. (If k_1 or $k_2 > \alpha n/\log n$, or $t > \beta n$, we simply let Alice send her input string.) Let $k = k_1 + k_2$.*

Both Alice's and Bob's algorithms have $L = O(\log \frac{n}{k\log n + t})$ levels.

For every $i \in [L]$, in the i-th level,

- *Let the block size be $b_i = \frac{n}{18\cdot 2^i(k+\frac{t}{\log n})}$, i.e., in each level, divide every block of x in the previous level evenly into two blocks. We choose L properly s.t. $b_L = O(\log n)$;*
- *The number of blocks $l_i = n/b_i$;*

Alice: On input $x \in \{0,1\}^n$,

1. *For the i-th level,*
   a. *Construct a hash function $h_i : \{0,1\}^{b_i} \to \{0,1\}^{b^* = \Theta(\log n)}$ for x by Theorem 8.*
   b. *Compute the sequence of hash values i.e. $v[i] = (h_i(x[1, 1+b_i)), h_i(x[1+b_i, 1+2b_i)), \ldots, h_i(x[1+(l_i-1)b_i, l_i b_i)))$;*
   c. *Compute the redundancy $z[i] \in (\{0,1\}^{b^*})^{\Theta((k+\frac{t}{\log n})i)}$ for $v[i]$ by using an algebraic geometry code[2], where the code has distance at least $180(k+\frac{t}{\log n})i$;*
2. *Compute the redundancy $z_{\text{final}} \in (\{0,1\}^{b_L})^{\Theta((k+\frac{t}{\log n})\log L)}$ for the blocks of the L-th level by using an algebraic geometry code[2]., where the code has distance at least $90(k+\frac{t}{\log n})L$;*
3. *Send $h = (h_1, \ldots, h_L)$, $z = (z[1], z[2], \ldots, z[L])$, $v[1]$, z_{final} to Bob.*

Bob: On input $y \in \{0,1\}^{O(n)}$ and received h, z, $v[1]$, z_{final},

1. *Create $\tilde{x} \in \{0,1,*\}^n$ (i.e. Bob's current version of Alice's x), initiating it to be $(*, *, \ldots, *)$;*
2. *For the i-th level where $1 \leq i \leq L-1$,*
   a. *Apply the decoding of the algebraic geometry code on $h_i(\tilde{x}'[1, 1+b_i)), h_i(\tilde{x}'[1+b_i, 1+2b_i)), \ldots, h_i(\tilde{x}'[1+(l_i-1)b_i, l_i b_i)), z[i]$ to get the sequence of hash values $v[i]$. Note that $v[1]$ is received directly, thus Bob does not need to compute it;*
   b. *Let $S = \{j \in [n] \mid h_i(\tilde{x}[1+(j-1)b_i, 1+jb_i)) \neq v[i][j]$ or $x[1+(j-1)b_i, 1+jb_i) = (*, \ldots, *)\}$;*

[2] See the full version `https://arxiv.org/abs/1809.00725`.

   c. *Compute the matching* $w_i = ((p_1, p'_1), \ldots, (p_{|w|}, p'_{|w|})) \in ([l_i] \times [|y|])^{|w_i|}$ *between* x_S *and* y *under* h_i*, using* $v[i]$*, by Lemma 12;*
   d. *Evaluate* $\tilde{x}$ *according to the matching, i.e. let* $\tilde{x}[p_j, p_j + b_i) = y[p'_j, p'_j + b_i)$*, where* $p_j, p'_j \in w_i, j \in [|w_i|]$*;*
3. *In the* L*'th level, apply the decoding of the algebraic geometry code on the blocks of* $\tilde{x}$ *and* z_{final} *to get* x*;*
4. *Return* x*.*

▶ **Lemma 15.** *For every* i*, the maximum non-overlapping matching between* x_S *and* y *under* h_i *has size at least* $|S| - (2k_1 + 3k_2 + t/\log n)$*.*

▶ **Lemma 16.** *For every* i*,* $|w_i| \geq 2/3(|S| - (2k_1 + 3k_2 + t/\log n))$*.*

▶ **Lemma 17.** *For every* i*, if* $v[1], \ldots, v[i]$ *are correctly recovered, then in the* i*-th level the number of wrongly recovered blocks of* x *is at most* $3i(2k_1 + 3k_2 + \frac{t}{\log n})$*.*

Proof. Consider the matching w^* corresponding to the current recovering of x after i levels, i.e., this matching is generated at level 1 and adjusted level by level. In level j, we first use hash values to test every block to see if it is correctly recovered. For wrongly recovered blocks we delete their corresponding matches. Then for remaining wrongly recovered blocks and unrecovered blocks, we compute a matching w_j for them, and add all matches in w_j to w^*.

For $w_j, j \leq i$, after level i, the number of wrongly recovered blocks in level i caused by (the remaining part of) w_j is at most $3(2k_1 + 3k_2 + \frac{t}{\log n})$.

This is because in w_j is constructed by Construction 12, which is a union of 3 matchings. Each matching of them is non-overlapping. We only need to show that w_j, after eliminating detected wrong pairs in these i levels, contains at most $2k_1 + 3k_2 + \frac{t}{\log n}$ wrong matches between x's and y's blocks in the i-th level. To see this, first note that these matches' y intervals are only from blocks which are modified from x's blocks or newly inserted. For each block-insertion of t_j bits, it can contribute at most $\lceil t_j/b_i \rceil + 1$ wrong matches. Each block-deletion can contribute at most 2 wrong matches. So totally block insertions/deletions can cause $\sum_{j=1}^{k_1}(\lceil t_j/b_i \rceil + 1) \leq 2k_1 + t/b_i$ wrong matches. On the other hand, k_2 block-transpositions can contribute at most $3k_2$ wrong matches, because 1 block-transposition can only cause 1 wrong match when deleting the block and inserting the block to its destination may contribute 2 wrong matches. Hence the total number wrong matches is at most $2k_1 + 3k_2 + t/b_i$.

Since there are i matchings $w_1, \ldots, w_i$, each containing 3 non-overlapping matchings, the number of wrongly recovered blocks remaining in w^* is at most $3i(2k_1 + 3k_2 + \frac{t}{\log n})$. ◀

▶ **Lemma 18.** *For every* i*, if* $v[1], \ldots, v[i]$ *are correctly recovered, then in level* i*, the number of unrecovered blocks is at most* $36i(k + \frac{t}{\log n})$*.*

Proof is in the full version.

▶ **Lemma 19.** *Bob can recover* x *correctly.*

Proof. We use induction to show that for every $i \in [L]$, $v[i]$ can be computed correctly by Bob.

For the first level, $v[1]$ is directly received from Alice.

Assume $v[1], \ldots, v[i-1]$ can be computed correctly. By Lemma 18, the number of unrecovered blocks after level $i-1$ is at most $36(i-1)(k + t/\log n)$. By Lemma 17, the number of wrongly recovered blocks is at most $9(i-1)(k + t/\log n)$. So the total number of wrongly recovered and unrecovered blocks is at most

$$2 \times (36(i-1)(k + t/\log n) + 9(i-1)(k + t/\log n)) \leq 90(i-1)(k + t/\log n) < 90i(k + t/\log n).$$

Note that with the redundancy $z[i]$, its corresponding code has distance at least $180(k+t/b_i)i$. So Bob can recover $v[i]$ correctly by the property of the algebraic geometry code.

As a result, at level L. By Lemma 17, the number of wrongly recovered blocks is at most $3L(2k_1 + 3k_2 + \frac{t}{b_L})$. By Lemma 18 the number of unrecovered blocks, is at most $36L(k + t/\log n)$. So the total number of wrongly recovered and unrecovered blocks is at most $45L(k + t/\log n)$. Note that the code distance corresponding to the redundancy z_{final} is at least $90(k + t/b_L)L$. So all blocks of x can be recovered correctly by using the decoding of the algebraic geometry code. ◀

Communication Complexity and running time computation are in the full version.

To this end, we showed Theorem 1.

3 Error Correcting Codes

We now briefly describe how to construct an error correcting code from a document exchange protocol for block edit errors. Similar to the construction in [7], our starting point is to first encode the sketch of the document exchange protocol using the code by Schulman and Zuckerman [23], which can resist edit errors and block transpositions. Then we concatenate the message with the encoding of the sketch. When decoding, we first decode the sketch, then apply the document exchange protocol on Bob's side to recover the message.

However, here we have an additional issue with this approach: a block transposition may move some parts of the encoding of the sketch to somewhere in the middle of the message, or vice versa. In this case, we won't be able to tell which part of the received string is the encoding of the sketch, and which part is the original message.

To solve this issue, we use a fixed string $\mathsf{buf} = 0^{\ell_{\mathsf{buf}}} \circ 1$ as a buffer to mark the encoding of the sketch, for some $\ell_{\mathsf{buf}} = O(\log n)$. More specifically, we evenly divide the encoding of the sketch into small blocks of length ℓ_{buf}, and insert buf before every block. Note that this only increases the length of the encoding of the sketch by a constant factor. The reason we use such a small block length is that, even if the adversary can forge or destroy some buffers, the total number of bits inserted or deleted caused by this is still small. In fact, we can bound this by $O(k)$ block insertions/deletions with at most $O(k \log n)$ bits inserted/deleted, for which both the sketch and the encoding of the sketch can handle. When decoding, we first recognize all the buf's. Then we take the ℓ_{buf} bits after each buf to form the decoding of the sketch, and take the remaining bits as the message.

Unfortunately, this approach introduces two additional problems here. The first problem is that the original message may contain buf as a substring. If this happens then in the decoding procedure again we will be taking part of the message to be in the encoding of the sketch. The second problem is that the small blocks of the encoding of the sketch may also contain buf. In this case we will be deleting information from the encoding of the sketch, which causes too many edit errors.

To address the first problem, we turn the original message into a *pseudorandom* string by computing the XOR of the message with the output of an appropriate pseudorandom generator that has seed length $O(\log n)$. We show that with high probability buf does not appear as a substring in the XOR. We can then exhaustively search for a seed that satisfies this requirement, and append the seed to the sketch of the document exchange protocol.

To address the second problem, we choose the length of the buffer to be longer than the length of each block in the encoding of the sketch, so that buf doesn't appear as a substring in any block. This is exactly why we choose the length of the buffer to be $\ell_{\mathsf{buf}} + 1$ while we choose the length of each block to be ℓ_{buf}.

If we directly apply our document exchange protocol to the construction above, we obtain an error correcting code with $O((k \log n + t) \log^2 \frac{n}{k \log n + t})$ redundant bits. Next we discuss how to achieve better redundancy for small k and t.

We first briefly describe the construction of the explicit binary code for k edit errors with redundancy $O(k \log n)$ in [7]. The construction in [7] starts by transforming the message into a string with the *B-distinct* property: any two substrings of length some $B = O(\log n)$ are distinct. This is obtained by computing the XOR of the message with the output of an appropriately designed pseudorandom generator. The construction then designs a document exchange protocol for such a string, and encodes the sketch of the document exchange protocol to give an error correcting code.

The document exchange protocol for a B-distinct string in [7] actually consists of two stages: in stage I, Alice uses a fixed pattern p to divide her string into blocks of size $\mathsf{poly}(\log n)$. Next, Alice sends a sketch of size $O(k \log n)$ to help Bob recover the partition of her string. To achieve this, Bob also divides his string into blocks in the same way that Alice does. Alice creates a vector V where each entry of V is indexed by a binary string of length B. Specifically, Alice looks at each block in her partition, and stores the B-prefix (the prefix of length B) of its next block and the length of the current block in the entry of V indexed by the B-prefix of the current block. This ensures each entry of the vector V has only $O(\log n)$ bits. Bob creates a vector V' in the same way. [7] shows that V and V' differ in at most $O(k)$ entries, thus Alice can send a sketch of size $O(k \log n)$ using the Reed-Solomon Code to help Bob recover V from V'. Once this is done, Bob can use V to obtain a guess of Alice's string.

Stage II consists of several levels. In each level, both parties divide each of their blocks evenly into $O(\log^{0.4} n)$ smaller blocks, and Alice generates a sequence of special hash functions called *ϵ-synchronization hash functions*. The nice properties of these hash functions guarantee that in each level Alice can send $O(k \log n)$ bits to Bob, so that Bob can recover all but $O(k)$ blocks of Alice's string. This stage ends in $O(\log_{\log^{0.4} n}(\mathsf{poly}(\log n))) = O(1)$ levels, where in the last level Alice can simply send a sketch of size $O(k \log n)$ for Bob to recover her string x.

Checking these two stages, it turns out that stage I can be modified to work for block edit errors as well. Intuitively, this is because it is still true that such errors won't cause too many different blocks between V and V'. On the other hand, stage II becomes problematic, since the use of ϵ-synchronization hash functions crucially relies on the monotone property of standard edit errors. Allowing block transpositions ruins this property, and it is not clear how to give suitable ϵ-synchronization hash functions to work in this case.

To solve the issue, in stage II, we can apply the deterministic document exchange protocol we developed earlier. This implies an error correcting code of redundancy $O(k \log n \log\log n + t)$. However, we show that we can further reduce the redundancy to $O(k \log n \log\log\log n + t)$ by using the string parsing idea in [8] to improve the partition in Stage I.

Given an input string, string parsing builds a tree where each leaf corresponds to a symbol of the input string, and each non-leaf node corresponds to a substring of the input string. Each node of the tree is associated with a label, which is the hash value of its corresponding substring under some hash function. The structure of the tree only depends locally on the input string, e.g., an edit error on the input string only affects $O(\log n \log^* n)$ nodes.

More specifically, string parsing builds the tree bottom-up from one level to another. The labels in the bottom level are obtained by directly applying the hash function to the symbols. Then, the algorithm builds one level of the tree as follows. The labels of the nodes in the previous level form a string of alphabet size $\mathsf{poly}(n)$. The algorithm first finds all repetitive substrings in this string (we say a substring is repetitive, if it's of the form a^l, for some $l \geq 2$). The remaining substrings satisfy the property that any two adjacent symbols are different, and we say such substrings are *non-repetitive*. [8] then applies an alphabet reduction algorithm to the non-repetitive substrings, and obtains a new non-repetitive string

for each substring over the alphabet $\{0,1,2\}$. The alphabet reduction works in $\log^* n$ steps, where in each step the alphabet size is reduced from the current size a to $\log a$. Thus in $\log^* n$ steps the alphabet size becomes a constant. Now for all the new strings obtained, the algorithm finds local maximums and local minimums that are not adjacent to any local maximum as *landmarks*, and partition the strings into small blocks of length 2 or 3 by using the landmarks. Finally, for each block, the algorithm builds a new node in this level, whose children are the nodes in the block and whose label is the hash value of the subtree.

Here, in our construction of error correcting codes, we use the idea of string parsing in stage I to partition Alice's string x into small blocks. Our goal is to partition the string into blocks of length roughly $\Theta(\log n \cdot \mathsf{poly}(\log\log n))$, while an edit error on the string can only affect a small number of contiguous blocks. In this way, stage II only takes $O(\log\log\log n)$ levels and the sketch size in stage II is $O(k\log n\log\log\log n + t)$. Note that each node in the parsing tree depends only locally on the input string. We use this property to bound the number of errors among the small blocks obtained in stage I.

To achieve our goal, instead of building a full parsing tree, we only build a partial parsing tree. That is, in each level of the parsing tree, we check the number of leaves under each node. If a node has more than T leaves for some threshold T, we mark the node as "finish". We also mark a node as "frozen", if all its adjacent nodes are marked as "finish". For each "finish" node, we build a new node in the next level, with the only child being this "finish" node. We then use these "finish" nodes to divide the string into several substrings, and apply alphabet reduction to the substrings, choose the landmarks, and partition each substring into small blocks according to the landmarks. Then for each small block, we build a new node in the next level, and set the children of the new node to be all nodes in the same block. We keep doing this until each node is either marked as "finish" or "frozen". Finally, we merge each "frozen" node to the "finish" node on its left or right. At the end of this process, we obtain several trees, and we partition the string x into small blocks, where each block consists of all the leaves in a tree. To remove the $O(\log^* n)$ factor, we only do two levels of alphabet reduction in each level of the tree. However, this will result in an alphabet size of $O(\log\log n)$, which means the tree may have $O(\log\log n)$ children. Hence, the block size may be as large as $O(T\log\log n)$. Note that each block depends on $O(\log T)$ blocks on its left and right, since in each level of the partial parsing tree, each node depends locally on a constant number of adjacent nodes. We prove that, if y is obtained from x by (k,t) block edit errors, then the partition of y can be obtained from the partition of x by $(k, O(t/T + k\log T))$ block edit errors over a larger alphabet. If we set $T = \log n$, then in stage I Alice still needs to send a sketch of $O(k\log n\log\log n + t)$ bits. To further reduce the redundancy, we apply the partial parsing tree method again with another threshold $T' = \Theta(\log\log n)$. Now the errors are reduced to $(k, O(\frac{t}{TT'} + k\log T'))$ block edit errors over a larger alphabet, and the block size increases by a $O(T'\log\log n\log\log\log n)$ factor, and becomes $O(\log n(\log\log n)^2)$.

We show that now in stage I, Alice can send a sketch with $O(\frac{t}{TT'} + k\log T')\cdot O(\log n) = O(k\log n\log\log\log n + t)$ bits; and in stage II, Alice can send a sketch with $O(k\log n\log\log\log n + t)$ bits. So the total sketch size is still $O(k\log n\log\log\log n + t)$. By using the encoding of Schulman and Zuckerman [23] and the buffer buf, the final redundancy of the error correcting code is also $O(k\log n\log\log\log n + t)$.

References

1 Djamal Belazzougui. Efficient Deterministic Single Round Document Exchange for Edit Distance. *CoRR*, abs/1511.09229, 2015. `arXiv:1511.09229`.

2 Djamal Belazzougui and Qin Zhang. Edit Distance: Sketching, Streaming, and Document Exchange. In *Proceedings of the 57th IEEE Annual Symposium on Foundations of Computer Science*, pages 51–60. IEEE, 2016.

3 J. Brakensiek, V. Guruswami, and S. Zbarsky. Efficient Low-Redundancy Codes for Correcting Multiple Deletions. *IEEE Transactions on Information Theory*, PP(99):1–1, 2017. doi:10.1109/TIT.2017.2746566.

4 Boris Bukh and Venkatesan Guruswami. An improved bound on the fraction of correctable deletions. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 1893–1901. ACM, 2016.

5 Diptarka Chakraborty, Elazar Goldenberg, and Michal Koucký. Low Distortion Embedding from Edit to Hamming Distance using Coupling. In *Proceedings of the 48th IEEE Annual Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 2016.

6 K. Cheng, B. Haeupler, X. Li, A. Shahrasbi, and K. Wu. Synchronization Strings: Efficient and Fast Deterministic Constructions over Small Alphabets. *ArXiv e-prints*, March 2018. arXiv:1803.03530.

7 Kuan Cheng, Zhengzhong Jin, Xin Li, and Ke Wu. Deterministic Document Exchange Protocols, and Almost Optimal Binary Codes for Edit Errors. In *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2018.

8 Graham Cormode and S. Muthukrishnan. The string edit distance matching problem with moves. *ACM Transactions on Algorithms*, 3(1), 2007.

9 Graham Cormode, Mike Paterson, Suleyman Cenk Sahinalp, and Uzi Vishkin. Communication complexity of document exchange. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 197–206. ACM, 2000.

10 V. Guruswami and R. Li. Efficiently decodable insertion/deletion codes for high-noise and high-rate regimes. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 620–624, July 2016. doi:10.1109/ISIT.2016.7541373.

11 V. Guruswami and C. Wang. Deletion Codes in the High-Noise and High-Rate Regimes. *IEEE Transactions on Information Theory*, 63(4):1961–1970, April 2017. doi:10.1109/TIT.2017.2659765.

12 Bernhard Haeupler. Optimal Document Exchange and New Codes for Small Number of Insertions and Deletions. *arXiv preprint*, 2018. arXiv:1804.03604.

13 Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization strings: codes for insertions and deletions approaching the Singleton bound. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 33–46. ACM, 2017.

14 Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization Strings: Explicit Constructions, Local Decoding, and Applications. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing*, 2018.

15 Bernhard Haeupler, Amirbehshad Shahrasbi, and Ellen Vitercik. Synchronization Strings: Channel Simulations and Interactive Coding for Insertions and Deletions. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming*, 2018.

16 Tom Høholdt, Jacobus H Van Lint, and Ruud Pellikaan. Algebraic geometry codes. *Handbook of coding theory*, 1(Part 1):871–961, 1998.

17 Utku Irmak, Svilen Mihaylov, and Torsten Suel. Improved single-round protocols for remote file synchronization. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1665–1676. IEEE, 2005.

18 Hossein Jowhari. Efficient Communication Protocols for Deciding Edit Distance. In *ESA*, 2012.

19 V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, February 1966.

20 H. Mercier, V. K. Bhargava, and V. Tarokh. A survey of error-correcting codes for channels with symbol synchronization errors. *IEEE Communications Surveys Tutorials*, 12(1):87–96, First 2010. doi:10.1109/SURV.2010.020110.00079.

21 A. Orlitsky. Interactive communication: balanced distributions, correlated files, and average-case complexity. In *[1991] Proceedings 32nd Annual Symposium of Foundations of Computer Science*, pages 228–238, October 1991. doi:10.1109/SFCS.1991.185373.

22 G. M. Tenengol'ts R. R. Varshamov. Code Correcting Single Asymmetric Errors. *Avtomat. i Telemekh*, 26:288–292, 1965.

23 L. J. Schulman and D. Zuckerman. Asymptotically good codes correcting insertions, deletions, and transpositions. *IEEE Transactions on Information Theory*, 45(7):2552–2557, November 1999. doi:10.1109/18.796406.

24 D Shapira and J. A. Storer. Edit distance with move operations. In *Proceedings of the 13th Symposium on Combinatorial Pattern Matching*, pages 85–98, 2002.

Restricted Max-Min Allocation: Approximation and Integrality Gap

Siu-Wing Cheng
Department of Computer Science and Engineering, HKUST, Hong Kong
scheng@cse.ust.hk

Yuchen Mao
Department of Computer Science and Engineering, HKUST, Hong Kong
ymaoad@cse.ust.hk

Abstract

Asadpour, Feige, and Saberi proved that the integrality gap of the configuration LP for the restricted max-min allocation problem is at most 4. However, their proof does not give a polynomial-time approximation algorithm. A lot of efforts have been devoted to designing an efficient algorithm whose approximation ratio can match this upper bound for the integrality gap. In ICALP 2018, we present a $(6+\delta)$-approximation algorithm where δ can be any positive constant, and there is still a gap of roughly 2. In this paper, we narrow the gap significantly by proposing a $(4+\delta)$-approximation algorithm where δ can be any positive constant. The approximation ratio is with respect to the optimal value of the configuration LP, and the running time is $poly(m,n) \cdot n^{poly(\frac{1}{\delta})}$ where n is the number of players and m is the number of resources. We also improve the upper bound for the integrality gap of the configuration LP to $3+\frac{21}{26} \approx 3.808$.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases fair allocation, configuration LP, approximation, integrality gap

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.38

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1905.06084`.

1 Introduction

Background

In the max-min fair allocation problem, we are given a set P of n players, a set R of m indivisible resources, and a set of non-negative values $\{v_{pr}\}_{p\in P, r\in R}$. For each $r \in R$ and each $p \in P$, resource r is worth a value of v_{pr} to player p. An allocation is a partition of R into disjoint subsets $\{D_p\}_{p\in P}$ so that each player p is assigned the resources in D_p. The goal is to find an allocation that maximizes the welfare of the least lucky player, that is, we want to maximize $\min_{p\in P} \sum_{r\in D_p} v_{pr}$. Unfortunately, unless P = NP, no polynomial-time algorithm can achieve an approximation ratio smaller than 2 [6].

Bezáková and Dani [6] tried to solve the problem using the assignment LP – a technique for the classic scheduling problem of makespan minimization [16]. However, they showed that the integrality gap of the assignment LP is unbounded, so rounding the assignment LP gives no guarantee on the approximation ratio. Later, Bansal and Sviridenko [4] proposed a stronger LP relaxation, the configuration LP, for the max-min allocation problem. Asadpour and Saberi [3] developed a polynomial-time rounding scheme for the configuration LP that gives an approximation ratio of $O(\sqrt{n}\log^3 n)$. Saha and Srinivasan [18] improved it to $O(\sqrt{n\log n})$. These approximation ratios almost match the lower bound of $\Omega(\sqrt{n})$ for the integrality gap of the configuration LP proved by Bansal and Svirodenko [4]. Bateni et al. [5]


and Chakrabarty et al. [7] established a trade-off between the approximation ratio and the running time. For any $\delta > 0$, they can achieve an approximation ratio of $O(n^{\delta})$ with $O(n^{1/\delta})$ running time.

In this paper, we study the restricted max-min allocation problem. In the restricted case, we have $v_{pr} \in \{v_r, 0\}$. That is, each resource r has an intrinsic value v_r, and it is worth value v_r to those players who desire it and value 0 to those who do not. Assuming P $\neq$ NP, the restricted case has a lower bound of 2 for the approximation ratio. The integrality gap of configuration LP for the restricted case also has a lower bound of 2. Bansal and Sviridenko [4] proposed an $O\big(\frac{\log\log n}{\log\log\log n}\big)$-approximation algorithm by rounding the configuration LP. Feige [11] proved that the integrality gap of the configuration LP is bounded by a constant, albeit large and unspecified. His proof was later made constructive by Haeupler et al. [12], and hence a constant approximation can be found in polynomial time. Asadpour et al. [2] viewed the restricted max-min allocation problem as a bipartite hyper-graph matching problem. Let T^* be the optimal value of the configuration LP. By adapting Haxell's [13] alternating tree technique for bipartite hyper-graph matchings, they proposed a local search algorithm that returns an allocation where every player receives at least $T^*/4$ worth of resources, and hence proved that the integrality gap of the configuration LP is at most 4. However, their algorithm is not known to run in polynomial time. A lot of efforts have been devoted to making their algorithm run in polynomial time. Polacek and Svensson [17] showed that the local search can be done in quasi-polynomial time by building the alternating tree in a more careful way. Annamalai, Kalaitzis and Svensson [1] carried out the local search in a more structured way. Together with two new *greedy* and *lazy update* strategies, they can find in polynomial time an allocation in which every player receives a value of at least $T^*/(6 + 2\sqrt{10} + \delta)$. Recently, we proposed a more flexible, aggressive greedy strategy that improves the approximation ratio to $6 + \delta$ [9]. Davies et al. [10] claimed a $(6 + \delta)$-approximation algorithm for the restricted max-min allocation problem by reducing it to the fractional matroid max-min allocation problem.

Our Contribution

We adapt the framework in [1] by introducing two new strategies: *layer-level node-disjoint paths* and *limited blocking*. The performance of our framework is determined by three parameters, and a trade-off between the running time and the quality of solution can be achieved by tuning these parameters. On one extreme, our framework acts exactly the same as the original local search in [2], which achieves a ratio of 4 but not necessarily run in polynomial time. On the other extreme, it becomes something like the algorithm in [1], which achieves a polynomial running time but a much worse ratio. We show that, in order to achieve a polynomial running time, one doesn't have to go from one extreme to the other – a marginal movement is sufficient. As a result, a ratio slightly worse than 4 can be achieved in polynomial time.

▶ **Theorem 1.** *For any constant $\delta > 0$, there is a $(4 + \delta)$-approximation algorithm for the restricted max-min allocation problem that runs in poly$(m, n) \cdot n^{poly(\frac{1}{\delta})}$ time.*

Although the algorithm we present takes the optimal value of the configuration LP as its input, one can avoid solving the configuration LP by combining our algorithm with binary search to zoom into the optimal value of configuration LP. The binary search technique is similar to that in [1, 9].

We also show that the integrality gap of the configuration LP is at most $3 + \frac{21}{26} \approx 3.808$ by giving a better analysis of the AFS algorithm. This improves the bound of $3 + \frac{5}{6} \approx 3.833$ recently obtained in [8, 15].

Primal

$$\sum_{C \in \mathcal{C}_p(T)} x_{p,C} \geqslant 1 \forall p \in P$$

$$\sum_{p \in P} \sum_{C \in \mathcal{C}_p(T): r \in C} x_{p,C} \leqslant 1 \forall r \in R$$

$$x_{p,C} \geqslant 0$$

Dual

$$\max \quad \sum_{p \in P} y_p - \sum_{r \in R} z_r$$

$$s.t. \quad y_p \leqslant \sum_{r \in C} z_r \qquad \forall p \in P, \forall C \in \mathcal{C}_p(T)$$

$$y_p \geqslant 0 \qquad \forall p \in P$$

$$z_r \geqslant 0 \qquad \forall r \in R$$

Figure 1 The configuration LP and its dual.

▶ **Theorem 2.** *The integrality gap of the configuration LP for the restricted max-min allocation problem is at most* $3 + \frac{21}{26} \approx 3.808$.

We focus on only the proof of Theorem 1 in the main text. The proof of Theorem 2 can be found in the full version of the paper. Other omitted proofs can also be found in the full version of the paper.

2 Preliminaries

2.1 The Configuration LP

Suppose that we hope to find an allocation where every player receives at least T worth of resources. A *configuration* for a player p is a subset D of the resources desired by p such that $\sum_{r \in D} v_r \geqslant T$. Let $\mathcal{C}_p(T)$ denote the set of all configurations for p.

The configuration LP is given on the left of Figure 1. Given a target T, the configuration LP, denoted as $CLP(T)$, associates a variable $x_{p,C}$ with each player p and each configuration C in $\mathcal{C}_p(T)$. Its first constraint ensures that each player receives at least 1 unit of configurations, and the second constraint guarantees that every resource r is used in at most 1 unit of configurations. The optimal value of the configuration LP is the largest T for which $CLP(T)$ is feasible. We denote this optimal value by T^*. Without loss of generality, we assume that $T^* = 1$ for the rest of the paper. Although the configuration LP may have an exponential number of variables, it can be solved within any constant relative error in polynomial time [4]. Viewing the objective function of the configuration LP as a minimization of a constant, one can get the dual LP on the right of the Figure 1.

2.2 Fat and thin edges

Our goal is to find an allocation in which every player receives at least λ worth of resources for some $\lambda \in (0, 1)$. In particular, our approximation algorithm sets $\lambda = \frac{1}{4+\delta}$ where δ is a positive constant. For each resource $r \in R$, we call r *fat* if $v_r \geqslant \lambda$, and *thin* otherwise. To find the target allocation, it suffices to assign each player p either a fat resource desired by p or a subset D of the thin resources desired by p with $\sum_{r \in D} v_r \geqslant \lambda$.

For every $p \in P$ and every fat resource r desired by p, we call $\{p, r\}$ a *fat edge*. For every $p \in P$ and every subset D of the thin resources desired by p, we call (p, D) a *thin edge* if $\sum_{r \in D} v_r \geqslant \lambda$. Two edges are *compatible* if they share no common resource. We say that a fat edge $\{p, r\}$ *covers* p and r. Similarly, a thin edge (p, D) *covers* p and the resources in D. A player or a resource is covered by a set of edges if it is covered by some edge in the set. For any $w \geqslant 0$, a thin edge (p, D) is a *w-minimal* if $\sum_{r \in D} v_r \geqslant w$ and $\sum_{r \in D'} v_r < w$ for any $D' \subsetneq D$. For a w-minimal thin edge (p, D), it is not hard to see that $w \leqslant \sum_{r \in D} v_r < w + \lambda$.

Given the above definitions of fat and thin edges, finding the target allocation is equivalent to finding a set of mutually compatible edges that covers all the players.

2.3 A local search idea

The following local search idea is initially proposed by Asadpour et al. [2], and is also used in [1, 9].

Let G be the bipartite graph formed by the players, the fat resources, and the fat edges. We maintain a set M of fat edges and a set $\mathcal{E}$ of thin edges such that: (i) M is a maximum matching of G, (ii) edges in $\mathcal{E}$ are λ-*minimal* and are mutually compatible, and (iii) each player is covered by at most one edge in $M \cup \mathcal{E}$. We call such M and $\mathcal{E}$ a partial allocation. Initially, M is an arbitrary maximum matching of G, and $\mathcal{E}$ is empty. The set $M \cup \mathcal{E}$ is updated and grown iteratively so that one more player is covered in each iteration. The final set $M \cup \mathcal{E}$ covers all the players and induces our target allocation.

Let p_0 be a player not yet covered by $M \cup \mathcal{E}$. We need to update $M \cup \mathcal{E}$ to cover p_0 without losing any player that are already covered. The simplest case is that we can find a player q_0 such that q_0 is covered by a thin edge a compatible with $\mathcal{E}$ and there is an alternating path [14] with respect to M from p_0 to q_0. Let π be this alternating path. We first update M by taking the symmetric difference $M \oplus \pi$, i.e., remove the edges in $\pi \cap M$ from the matching and add the edges in $\pi \setminus M$ to the matching. $M \oplus \pi$ is also a maximum matching of G. After the update, p_0 becomes matched while q_0 becomes unmatched. Then we add a to $\mathcal{E}$ to cover q_0 again. Here we slight abuse the notion of alternating paths in the sense that wen allow an alternating path with no edge. The $\oplus$ can easily extend to alternating paths with no edge.

It is possible that no edge covering q_0 is compatible with $\mathcal{E}$. Let a be an edge covering q_0. Suppose that b is an edge in $\mathcal{E}$ that is not compatible with a. We say b *blocks* a. Let p_1 be the player covered by b. In order to add a to $\mathcal{E}$, we have to release b from $\mathcal{E}$. But we cannot lose p_1, so before we release b, we need to find another edge to cover p_1. Now p_1 has a similar role as p_0.

2.4 Node-disjoint alternating paths

In order to achieve a polynomial running time, our algorithm updates M using multiple node-disjoint alternating paths from unmatched players to players . In this section, we define a problem of finding a largest set of node-disjoint paths. We also extend the $\oplus$ operation to a set of node-disjoint paths.

For any maximum matching M of G, we define G_M to be the directed graph obtained from G by orienting edges of G from r to p if $\{p, r\} \in M$, and from p to r if $\{p, r\} \notin M$. Let S be a subset of the players not matched by M. Let T be a subset of the players. Finding the largest set of node-disjoint alternating paths from S to T is equivalent to finding the largest set of node-disjoint paths in G_M from S to T. Let $G_M(S, T)$ denote the problem of finding the largest set of node-disjoint paths from S to T in G_M. Let $f_M(S, T)$ denotes the maximum number of such paths. Note that when $S \cap T \neq \emptyset$, a path consisting of a single node is allowed. Such path is called a trivial path. Paths with at least one edge is non-trivial. Let Π be a feasible solution for $G_M(S, T)$. The paths in Π originate from a subset of S, which we call the *sources* and denote as src_Π, and terminate in a subset of T, which we call the *sinks* and denote as $sink_\Pi$. We extend the $\oplus$ operation to Π. Viewing Π as a set of edges, $M \oplus \Pi$ stands for removing the edges in $\Pi \cap M$ from the matching and adding the edges in $\Pi \setminus M$ to the matching. One can see that $M \oplus \Pi$ is a maximum matching of G.

The problem $G_M(S, T)$ can be solved in polynomial time. Please refer to the full version of the paper for more details.

3 An Approximation Algorithm

We discuss below a few techniques used by our algorithm. Some of them are used in [1, 9, 10]. The *limited blocking* strategy is brand new, and is crucial to achieving an approximation ratio of $4 + \delta$. In the following discussion, one can interpret addable edges as thin edges that we hope to add to $\mathcal{E}$, and blocking edges as edges in $\mathcal{E}$ that are not compatible with addable edges. The precise definition will be given later.

Layers. As in [1, 9], we maintain a stack of layers, where each layer consists of addable edges and their blocking edges. The key to achieving a polynomial running time is to guarantee a geometric growth in the number of blocking edges from the bottom to the top of the stack.

Layer-level node-disjoint paths. We require that the players covered by the addable edges in a layer can be simultaneously reached via node-disjoint paths in G_M from the players covered by the blocking edges in the previous layers [10]. It has the same effect as the globally node-disjoint path used in [1]: if lots of addable edges in a layer become unblocked, then a significant update can be made. The advantage of our strategy is that it offers more flexibility when building a new layer.

Lazy update. When having an unblocked addable edge, one may be tempted to update M and $\mathcal{E}$ immediately. However, as in [1], in order to achieve a polynomial running time, we should wait until there are lots of unblocked addable edges, and then a significant update can be made in one step. The laziness is controlled by a small constant μ that will be defined later.

Greedy and Limited Blocking. Recall that the key to achieving a polynomial running time is to guarantee a geometric growth in the number of blocking edges from the bottom to the top of the stack. In [2], every addable edge is λ-minimal, and each blocking edge blocks exactly one addable edge. If using this strategy, in worst case, one may get a layer consisting of one addable edge that is blocked by many blocking edges. After some of these blocking edges are released from $\mathcal{E}$, we may be left with a layer of a single addable edge that is blocked by a single blocking edge, which breaks the geometric growth in the number of blocking edges. To resolve this issue, Annamalai et al. [1] allow a blocking edge to block as many addable edges as possible. However, it brings a new trouble: one may get a layer consisting of many addable edges that are blocked by one blocking edge. As a consequence, they have to introduce another strategy *Greedy*. They require every addable edge to be $\frac{1}{2}$-minimal. If such an addable edge is blocked, at least $\frac{1}{2} - \lambda$ worth of its resources must be occupied by blocking edges. Provided that a blocking edge is λ-minimal and covers at most 2λ worth of resources, the greedy strategy ensures that, in a layer, the number of blocking edges cannot be too small comparing with the number of addable edges. Analysis shows that although the greedy strategy makes the algorithm faster, it deteriorates the approximation ratio. Our strategy is a generalization of those used in [2] and [1]. We allow a blocking edge to block more than one addable edge, but once it shares strictly more than $\beta\lambda$ worth of resources with the addable edges blocked by it, we stop it from blocking more edges. We use greedy too. In our algorithm, addable edges are $(1 + \gamma)\lambda$-minimal for some constant γ.

If we set β, γ, μ to be 0, then our algorithm acts exactly the same as the local search in [2], which achieves a ratio of 4 but may not run in polynomial time. If β, γ are set to be some large constant, then our algorithm acts like the algorithm in [1] which achieves a polynomial running time but a much worse ratio. We show that carefully selected tiny β and tiny γ guarantee a polynomial running time but barely hurt the approximation ratio.

3.1 The algorithm

Let $M \cup \mathcal{E}$ be the current partial allocation. Let p_0 be a player that is not yet covered by $M \cup \mathcal{E}$. The algorithm alternates between two phases to update and extend $M \cup \mathcal{E}$ so that the partial allocation covers p_0 eventually without losing any covered player. In the building phase, it pushes new layers onto a stack, where each layer stores some addable edges and their blocking edges. In the collapse phase, it uses unblocked addable edges to release some blocking edges in some layer from $\mathcal{E}$.

Since we frequently talk about resources covered by thin edges and take sum of values over a set of resources, we define the following notations. Given a thin edge e, R_e denotes the set of resources covered by e. Given a set $\mathcal{S}$ of thin edges, $R(\mathcal{S})$ denotes the set of thin resources covered by $\mathcal{S}$. Given a set D of resources, define $v[D] = \sum_{r \in D} v_r$.

3.1.1 Building phase

The algorithm maintains a stack of layers. The layer index starts with 1 from the bottommost layer in the stack. The i-th layer L_i is a tuple $(\mathcal{A}_i, \mathcal{B}_i, d_i, z_i)$, where $\mathcal{A}_i$ is a set of addable edges that we want to add to $\mathcal{E}$, $\mathcal{B}_i$ is a set of blocking edges that prevent us from doing so, and d_i and z_i are two values maintained for the sake of analysis. The algorithm also maintains a set $\mathcal{I}$ of addable edges that are compatible with $\mathcal{E}$. We will define addable edges and blocking edges later. We use ℓ to denote the number of layers in the current stack. The state of the algorithm is specified by $(M, \mathcal{E}, \mathcal{I}, (L_1, \ldots, L_\ell))$.

For each $\mathcal{A}_i$, we use A_i to denote the set of players covered by $\mathcal{A}_i$. Similarly, B_i and I denote the set of players covered by $\mathcal{B}_i$ and $\mathcal{I}$, respectively. For $i \in [1, \ell]$, define $\mathcal{B}_{\leqslant i} = \bigcup_{j=1}^{i} \mathcal{B}_j$, $B_{\leqslant i} = \bigcup_{j=1}^{i} B_j$, and $\mathcal{A}_{\leqslant i} = \bigcup_{j=1}^{i} \mathcal{A}_j$. We do not define $A_{\leqslant i}$ because although two addable edges in different layers do not share any resource, they may cover the same player. This is a consequence of the layer-level node-disjoint paths technique.

For simplicity, we define the first layer L_1 to be $(\emptyset, \{(p_0, \emptyset)\}, 0, 0)$. That is, $\mathcal{A}_1 = \emptyset$, $\mathcal{B}_1 = \{(p_0, \emptyset)\}$, and $d_1 = z_1 = 0$.

The layers are built inductively. Initially, there is only the layer L_1 and $\mathcal{I} = \emptyset$. Let ℓ be the number of layers in the current stack. Consider the construction of the $(\ell + 1)$-th layer.

▶ **Definition 3.** *Let $\beta \geqslant 0$ be a constant to be specified later. A thin resource r is **inactive** if* (i) *$r \in R(\mathcal{A}_{\leqslant \ell} \cup \mathcal{B}_{\leqslant \ell})$, or* (ii) *$r \in R(\mathcal{A}_{\ell+1} \cup \mathcal{I})$, or* (iii) *$r \in R_b$ for some $b \in \mathcal{B}_{\ell+1}$ and $v[R_b \cap R(\mathcal{A}_{\ell+1})] > \beta\lambda$. If a thin resource is not inactive, then it is **active**.*

We will define addable edges so that they use only active thin resources.

▶ **Definition 4.** *A player p is **addable** if $f_M(B_{\leqslant \ell}, A_{\ell+1} \cup I \cup \{p\}) = f_M(B_{\leqslant \ell}, A_{\ell+1} \cup I) + 1$.*

The activeness of thin resources and the addability of the players depend on $\mathcal{A}_{\ell+1}$ and $\mathcal{I}$ ($A_{\ell+1}$ and I), so they may be affected as we add edges to $\mathcal{A}_{\ell+1}$ and $\mathcal{I}$.

▶ **Definition 5.** *A thin edge (p, D) is **addable** if p is addable and D is a set of **active** thin resources desired by p with $v[D] \geqslant \lambda$. The **blocking edges** of an addable edge (p, D) are $\{e \in \mathcal{E} : R_e \cap D \neq \emptyset\}$. An addable edge (p, D) is **unblocked** if $v[D \setminus R(\mathcal{E})] \geqslant \lambda$.*

Recall that, for any $w > 0$, a thin edge (p, D) is a *w-minimal* if $v[D] \geqslant w$ and $v[D'] < w$ for any $D' \subsetneq D$. Our algorithm considers two kinds of addable edges. The first kind is unblocked addable edges that are λ-minimal. It is easy to see that if an unblocked addable edge is λ-minimal, then it must be compatible with $\mathcal{E}$. We use $\mathcal{I}$ to keep such addable edges. The second kind is blocked addable edges that are $(1 + \gamma)\lambda$-minimal, where γ is a constant

Table 1 Some facts about $\mathcal{I}$ and the layers in the stack.

Fact 1	Edges in $\mathcal{A}_{\leqslant \ell} \cup \mathcal{I}$ are mutually compatible.
Fact 2	Edges in $\mathcal{I}$ are compatible with edges in $\mathcal{E}$.
Fact 3	For any $i \in [1, \ell]$, no two edges in $\mathcal{A}_i \cup \mathcal{I}$ cover the same player.
Fact 4	$\{\mathcal{B}_2, \ldots, \mathcal{B}_\ell\}$ are disjoint subsets of $\mathcal{E}$. Note that $\mathcal{B}_1 = \{(p_0, \emptyset)\}$ does not share any resource with $\mathcal{B}_i$ for $i \in [2, \ell]$.

to be specified later. Such edges will be added to $\mathcal{A}_{\ell+1}$. Once a $(1+\gamma)\lambda$-minimal addable edge (p, D) becomes unblocked, we can easily extract a λ-minimal unblocked addable edge (p, D') with $D' \subseteq D$.

Consider condition (iii) in Definition 3. Let b be a blocking edge in $\mathcal{B}_{\ell+1}$. When R_b and $R(\mathcal{A}_{\ell+1})$ share strictly more than $\beta\lambda$ worth of resources, all resources in R_b become inactive. Any addable edge to be added to $\mathcal{A}_{\ell+1}$ in the future cannot use these inactive resources, and hence, will not be blocked by b. This is how we achieve "limited blocking" mentioned before.

We call BUILD below to construct the $(\ell+1)$-th layer. Note that after adding an addable edge to $\mathcal{A}_{\ell+1}$, we immediately add its blocking edges to $\mathcal{B}_{\ell+1}$ in order to keep the inactive/active status of thin resources up-to-date.

BUILD$(M, \mathcal{E}, \mathcal{I}, (L_1, \cdots, L_\ell))$

1. Initialize $\mathcal{A}_{\ell+1} = \emptyset$ and $\mathcal{B}_{\ell+1} = \emptyset$.
2. While there is an unblocked addable edge that is λ-minimal, add it to $\mathcal{I}$.
3. While there is an addable edge (p, D) that is $(1+\gamma)\lambda$-minimal
   - 3.1 add (p, D) to $\mathcal{A}_{\ell+1}$. (Note that (p, D) must be blocked; otherwise, we could extract from it a λ-minimal unblocked addable edge, which should be added to $\mathcal{I}$ in step 2.)
   - 3.2 add to $\mathcal{B}_{\ell+1}$ the edges in $\mathcal{E}$ that block (p, D).
4. Set $d_{\ell+1} := f_M(B_{\leqslant \ell}, A_{\ell+1} \cup I)$, $z_{\ell+1} := |A_{\ell+1}|$, and $L_{\ell+1} := (\mathcal{A}_{\ell+1}, \mathcal{B}_{\ell+1}, d_{\ell+1}, z_{\ell+1})$.
5. Update $\ell := \ell + 1$

Table 1 lists a few facts about the layers.

3.1.2 Collapse phase

When some layer becomes collapsible, the algorithm enters the collapse phase. Let $(M, \mathcal{E}, \mathcal{I}, (L_1, \cdots, L_\ell))$ be the current state of the algorithm. In order to determine whether a layer is collapsible or not, we need to compute the following decomposition of $\mathcal{I}$. Let $(\mathcal{I}_1, \ldots, \mathcal{I}_{\ell-1})$ be some disjoint subsets of $\mathcal{I}$. Let I_i denote the set of players covered by $\mathcal{I}_i$. For $i \in [1, \ell-1]$, we use $\mathcal{I}_{\leqslant i}$ and $I_{\leqslant i}$ to denote $\bigcup_{j=1}^{i} \mathcal{I}_j$ and $\bigcup_{j=1}^{i} I_j$, respectively.

▶ **Definition 6.** *A collection of disjoint subsets $(\mathcal{I}_1, \ldots, \mathcal{I}_{\ell-1})$ of $\mathcal{I}$ is a **canonical decomposition** of $\mathcal{I}$ if for all $i \in [1, \ell-1]$, $f_M(B_{\leqslant i}, I_{\leqslant i}) = f_M(B_{\leqslant i}, I) = |I_{\leqslant i}|$. A solution Γ for $G_M(B_{\leqslant \ell-1}, I)$ is a **canonical solution** with respect to the canonical decomposition $(\mathcal{I}_1, \ldots, \mathcal{I}_{\ell-1})$ if Γ can be partitioned into disjoint subsets $(\Gamma_1, \ldots, \Gamma_{\ell-1})$ such that for every $i \in [1, \ell-1]$, Γ_i is a set of $|I_i|$ paths from B_i to I_i in G_M.*

Although it is not clear from the definition, invariant 1 in Table 2 implies that $(\mathcal{I}_1, \ldots, \mathcal{I}_{\ell-1})$ is indeed a partition of $\mathcal{I}$. The following lemma is analogous to its counterpart in [1, 9]. Its proof is omitted.

▶ **Lemma 7.** *Let ℓ be the number of layers in the stack. A canonical decomposition of $\mathcal{I}$ and a corresponding canonical solution for $G_M(B_{\leqslant \ell-1}, I)$ can be computed in poly(ℓ, m, n) time.*

All the edges in $\mathcal{I}_i$ are compatible with $\mathcal{E}$, and all the players in I_i can be reached from B_i by node-disjoint paths Γ_i in G_M, so every edge in $\mathcal{I}_i$ can be used to release one blocking edge in $\mathcal{B}_i$ from $\mathcal{E}$. A layer is collapsible if a certain fraction of its blocking edges can be released.

▶ **Definition 8.** *Let μ be a constant to be specified later. A layer L_i is **collapsible** if there is a canonical decomposition $(\mathcal{I}_1, \ldots, \mathcal{I}_{\ell-1})$ of $\mathcal{I}$ such that $|I_i| > \mu |B_i|$.*

Note that although there can be more than one canonical decomposition, the collapsibility of a layer is independent of the choice of canonical decompositions, because by definition, we always have $|I_i| = f_M(B_{\leqslant i}, I) - f_M(B_{\leqslant i-1}, I)$.

When some layer is collapsible, we enter the collapse phase, call Collapse to shrink collapsible layers until no layer is collapsible, and then return to the build phase.

Collapse$(M, \mathcal{E}, \mathcal{I}, (L_1, \cdots, L_\ell))$

1. Compute a canonical decomposition $(\mathcal{I}_1, \ldots, \mathcal{I}_{\ell-1})$ and a corresponding canonical solution $\Gamma_1 \cup \cdots \cup \Gamma_{\ell-1}$ for $G_M(B_{\leqslant \ell-1}, I)$. If no layer is collapsible, go to the build phase; otherwise, let L_t be the collapsible layer with the smallest index.
2. Remove all the layers above L_t from the stack. Set $\mathcal{I} := \mathcal{I}_{\leqslant t-1}$.
3. Recall that $src_{\Gamma_t} \subseteq B_t$ by Definition 6. Let $\mathcal{B}^\Gamma$ denote the set of edges in $\mathcal{B}_t$ that are incident to players in src_{Γ_t}. We use $\mathcal{I}_t$ and Γ_t to release the edges in $\mathcal{B}^\Gamma$.
   - 3.1 Update M by flipping the paths in Γ_t, i.e., set $M := M \oplus \Gamma_t$.
   - 3.2 Add to $\mathcal{E}$ the edges in $\mathcal{I}_t$, i.e., set $\mathcal{E} := \mathcal{E} \cup \mathcal{I}_t$.
   - 3.3 Each player in src_{Γ_t} is now covered by either a fat resource or a thin edge from $\mathcal{I}_t$. If $t = 1$, then p_0 is already covered, and the algorithm terminates. Assume that $t \geqslant 2$. Edges in $\mathcal{B}^\Gamma$ can be safely released from $\mathcal{E}$. Set $\mathcal{E} := \mathcal{E} \setminus \mathcal{B}^\Gamma$ and $\mathcal{B}_t := \mathcal{B}_t \setminus \mathcal{B}^\Gamma$.
4. If $t \geqslant 2$, we need to update $\mathcal{A}_t$ because some edges in $\mathcal{A}_t$ may become unblocked due to the release of blocking edges. For every edge (p, D) in $\mathcal{A}_t$ that becomes unblocked,
   - 4.1 Remove (p, D) from $\mathcal{A}_t$,
   - 4.2 if $f_M(B_{\leqslant t-1}, I \cup \{p\}) = f_M(B_{\leqslant t-1}, I) + 1$, then extract a λ-minimal unblocked addable edge (p, D') from (p, D), and add (p, D') to $\mathcal{I}$.
5. Update $\ell := t$. Go to step 1.

4 Analysis of the approximation algorithm

4.1 Some invariants

Table 2 lists a few invariants, where ℓ is the number of layers in the stack. Lemmas 9 and 10 below have analogous versions in [1, 9] and can be proved similarly. We omit their proofs.

▶ **Lemma 9.** Build *and* Collapse *maintain the invariants in Table 2.*

▶ **Lemma 10.** *Let $(L_1, \ldots, L_\ell)$ be the stack of layers. If no layer is collapsible, then* (i) *$|I| \leqslant \mu |B_{\leqslant \ell-1}|$, and* (ii) *for all $i \in [1, \ell]$, $|A_i| \geqslant z_i - \mu |B_{\leqslant i-1}|$.*

Table 2 Invariants maintained by the algorithm.

Invariant 1	$f_M(B_{\leqslant \ell-1}, I) = \lvert I\rvert$.
Invariant 2	For all $i \in [1, \ell-1]$, $f_M(B_{\leqslant i}, A_{i+1} \cup I) \geqslant d_{i+1}$.
Invariant 3	For all $i \in [1, \ell]$, $\lvert A_i\rvert \leqslant z_i$.
Invariant 4	For all $i \in [1, \ell]$, $d_i \geqslant z_i$.

4.2 Bounding the number of blocking edges

Lemma 11 – 13 are consequences of our greedy and limited blocking strategies. Basically, they bound the number of blocking edges in a layer in terms of the number of addable edges.

▶ **Lemma 11.** *Let $L_i = (\mathcal{A}_i, \mathcal{B}_i, d_i, z_i)$ be an arbitrary layer in the stack. For each edge $b \in \mathcal{B}_i$, there is an edge $a \in \mathcal{A}_i$ such that $v[R_b \cap R(\mathcal{A}_i \setminus \{a\})] \leqslant \beta\lambda$.*

Proof. Let b be an edge in $\mathcal{B}_i$. Sort the edges in $\mathcal{A}_i$ in chronological order of their additions into $\mathcal{A}_i$. Let a be the last edge in $\mathcal{A}_i$ that is blocked by b. By our choice of a, the edges in $\mathcal{A}_i$ that are added after a cannot be blocked by b, so they do not share any common resource with b. Among the edges added before a, let S be the subset of their resources that are also covered by b. We claim that $v[S] \leqslant \beta\lambda$. If not, all resources in R_b would be inactive before the addition of a by definition of inactive resources. So no resource in R_b could be included in a. But $R_a \cap R_b$ must be non-empty as b blocks a, a contradiction. ◀

▶ **Lemma 12.** *Let $L_i = (\mathcal{A}_i, \mathcal{B}_i, d_i, z_i)$ be an arbitrary layer in the stack. We have $\lvert A_i\rvert < (1 + \frac{\beta}{\gamma})\lvert B_i\rvert$.*

Proof. By Lemma 11, for each $b \in \mathcal{B}_i$, we can identify an edge $a_b \in \mathcal{A}_i$ so that $v[R_b \cap R(\mathcal{A}_i \setminus \{a_b\})] \leqslant \beta\lambda$. Let $\mathcal{A}_i^0 = \{a_b : b \in \mathcal{B}_i\}$ be the set of edges identified. $\lvert\mathcal{A}_i^0\rvert \leqslant \lvert B_i\rvert$.

Let $\mathcal{A}_i^1 = \mathcal{A}_i \setminus \mathcal{A}_i^0$. For every $b \in \mathcal{B}_i$, $v[R_b \cap R(\mathcal{A}_i^1)] \leqslant v[R_b \cap R(\mathcal{A}_i \setminus \{a_b\})] \leqslant \beta\lambda$. Taking sum over all edges in $\mathcal{B}_i$, we get $v[R(\mathcal{B}_i) \cap R(\mathcal{A}_i^1)] \leqslant \beta\lambda\lvert B_i\rvert$. On the other hand, each edge a in $\mathcal{A}_i^1$ is $(1+\gamma)\lambda$-minimal and is blocked, so it must have more than $\gamma\lambda$ worth of its resources occupied by edges in $\mathcal{B}_i$, i.e., $v[R(\mathcal{B}_i) \cap R_a] > \gamma\lambda$. Taking sum over all the edges in $\mathcal{A}_i^1$ gives $v[R(\mathcal{B}_i) \cap R(\mathcal{A}_i^1)] > \gamma\lambda\lvert\mathcal{A}_i^1\rvert$. Hence, $\beta\lambda\lvert B_i\rvert \geqslant v[R(\mathcal{B}_i) \cap R(\mathcal{A}_i^1)] > \gamma\lambda\lvert\mathcal{A}_i^1\rvert$.

Finally we get $\lvert B_i\rvert = \lvert\mathcal{A}_i^0\rvert + \lvert\mathcal{A}_i^1\rvert \leqslant \lvert B_i\rvert + \frac{\beta}{\gamma}\lvert B_i\rvert < (1 + \frac{\beta}{\gamma})\lvert B_i\rvert$. ◀

▶ **Lemma 13.** *Let $L_i = (\mathcal{A}_i, \mathcal{B}_i, d_i, z_i)$ be an arbitrary layer in the stack. Let $\mathcal{B}_i'$ be the set of edges in $\mathcal{B}_i$ that share strictly more than $\beta\lambda$ resources with edges in $\mathcal{A}_i$. More precisely, $\mathcal{B}_i' = \{e \in \mathcal{B}_i : v[R_e \cap R(\mathcal{A}_i)] > \beta\lambda\}$. We have $\lvert\mathcal{B}_i'\rvert < \frac{2+\gamma}{\beta}\lvert A_i\rvert$.*

Proof. Taking the sum of $v[R_e \cap R(\mathcal{A}_i)]$ over all edges e in $\mathcal{B}_i'$, we obtain $v[R(\mathcal{B}_i') \cap R(\mathcal{A}_i)] > \beta\lambda\lvert\mathcal{B}'_i\rvert$. On the other hand, $v[R(\mathcal{B}_i') \cap R(\mathcal{A}_i)] \leqslant v[R(\mathcal{A}_i)] < (2+\gamma)\lambda\lvert A_i\rvert$. The last inequality is because that edges in $\mathcal{A}_i$ are $(1+\gamma)\lambda$-minimal. Combining the above two inequality, we obtain $\beta\lambda\lvert\mathcal{B}'_i\rvert < (2+\gamma)\lambda\lvert A_i\rvert \Rightarrow \lvert\mathcal{B}_i'\rvert < \frac{2+\gamma}{\beta}\lvert A_i\rvert$. ◀

4.3 Geometric growth in the number of blocking edges

Now we are ready to prove that the number of blocking edges grow geometrically from bottom to top. Lemma 14 states that there are lots of addable edges in a layer *immediately after* its construction. Previous Lemma 10(ii) ensures that as long as there is no collapsible layer, every layer cannot lose too many addable edges. Therefore, Lemma 14 implies that

when no layer is collapsible, then every layer must have lots of addable edges, even if they are not newly constructed. Then since the number of blocking edges in a layer is lower bounded in terms of the number of addable edges by Lemma 12, we can conclude that there must be lots of blocking edges in a layer when no layer in the stack is collapsible (Lemma 17).

▶ **Lemma 14.** *Let $(M, \mathcal{E}, \mathcal{I}, (L_1, \ldots, L_{\ell+1}))$ be the state of the algorithm **immediately after** the construction of $L_{\ell+1}$. If no layer is collapsible, then $z_{\ell+1} = |\mathcal{A}_{\ell+1}| \geqslant 2\mu|\mathcal{B}_{\leqslant\ell}|$.*

Proof. We give a proof by contradiction. Suppose that $z_{\ell+1} = |\mathcal{A}_{\ell+1}| < 2\mu|\mathcal{B}_{\leqslant\ell}|$. We will show that the dual of $CLP(1)$ is unbounded, which implies that $CLP(1)$ is infeasible, contradicting the assumption that the configuration LP has optimal value $T^* = 1$.

Consider the moment immediately after we finish adding edges to $\mathcal{A}_{\ell+1}$ during the construction of $L_{\ell+1}$. At this moment, there is no $(1+\gamma)\lambda$-minimal addable edge left. The rest of the proof is with respect to this moment.

Let Π be an optimal solution for $G_M(B_{\leqslant\ell}, A_{\ell+1} \cup I)$. Note that $M \oplus \Pi$ is a maximum matching of G. Consider the directed graph $G_{M\oplus\Pi}$ obtained by orienting the edges of G according to whether they are in $M \oplus \Pi$ or not. Let P^+ be the set of players that can be reached in $G_{M\oplus\Pi}$ from $B_{\leqslant\ell} \setminus src_\Pi$. Let R_f^+ be the set of fat resources that can be reached in $G_{M\oplus\Pi}$ from $B_{\leqslant\ell} \setminus src_\Pi$. Let R_t^+ be the set of inactive thin resources.

▷ Claim 15. (i) Players in P^+ are still addable after we finish adding edges to $\mathcal{A}_{\ell+1}$ (ii) Players in P^+ have in-degree at most 1 in $G_{M\oplus\Pi}$. (iii) Resources in R_f^+ have out-degree exactly 1 in $G_{M\oplus\Pi}$.

We define a dual solution $(\{y_p^*\}_{p\in P}, \{z_r^*\}_{r\in R})$ as follows.

$$y_p^* = \begin{cases} 1-(1+\gamma)\lambda & \text{if } p \in P^+, \\ 0 & \text{otherwise.} \end{cases} \qquad z_r^* = \begin{cases} 1-(1+\gamma)\lambda & \text{if } r \in R_f^+, \\ v_r & \text{if } r \in R_t^+, \\ 0 & \text{otherwise.} \end{cases}$$

▷ Claim 16. $(\{y_p^*\}_{p\in P}, \{z_r^*\}_{r\in R})$ is a feasible solution, and it has a positive objective function value.

Suppose that Claim 16 holds. Then $(\{\alpha y_p^*\}_{p\in P}, \{\alpha z_r^*\}_{r\in R})$ is also a feasible solution for any $\alpha > 0$. As α goes to infinity, the objective function value goes to infinity, yielding the contradiction that we look for. ◀

We omit the proof of Claim 15. We give the proof of Claim 16 below.

Feasibility. We need to show that $\forall p \in P$, $\forall C \in \mathcal{C}_p(1)$, $y_p^* \leqslant \sum_{r\in C} z_r^*$. If $p \notin P^+$, then $y_p^* = 0$, and the inequality holds since z_r^* is non-negative. Assume that $p \in P^+$. So $y_p^* = 1-(1+\gamma)\lambda$. Let C be any configuration for p. We show that $\sum_{r\in C} z_r^* \geqslant 1-(1+\gamma)\lambda$.

Case 1: C contains a fat resource r_f. Since p desires r_f, $G_{M\oplus\Pi}$ has either an edge (p, r_f) or an edge (r_f, p). By the definition of P^+, there is a path π in $G_{M\oplus\Pi}$ from $B_{\leqslant\ell} \setminus src_\Pi$ to p. If $G_{M\oplus\Pi}$ has an edge (p, r_f), we can reach r_f from $B_{\leqslant\ell} \setminus src_\Pi$ by following π and then (p, r_f). So $r_f \in R_f^+$. If $G_{M\oplus\Pi}$ has an edge (r_f, p), then p is matched by $M \oplus \Pi$. Since players in $B_{\leqslant\ell} \setminus src_\Pi$ are not matched by $M \oplus \Pi$, $p \notin (B_{\leqslant\ell} \setminus src_\Pi)$. By Claim 15, in $G_{M\oplus\Pi}$, the in-degree of p is at most one, so (r_f, p) is the only edge entering p. To reach p, π must reach r_f first. Hence, we can follow π to reach r_f from $B_{\leqslant\ell} \setminus src_\Pi$, which implies $r_f \in R_f^+$. In both cases, we have $\sum_{r\in C} z_r^* \geqslant z_{r_f}^* = 1-(1+\gamma)\lambda$.

Case 2: C contains only thin resources. By Claim 15, p is still addable after we finish adding edges to $\mathcal{A}_{\ell+1}$. However, when we finish adding edges to $\mathcal{A}_{\ell+1}$, there is no $(1+\gamma)\lambda$-minimal addable edge left. It must be that p does not have enough active resources to form an addable edge. The active thin resources in C must have a total value less than $(1+\gamma)\lambda$. Recall that $v[C] \geqslant 1$. At least $1-(1+\gamma)\lambda$ worth of thin resources in C are inactive. Since $z^*_r = v_r$ for inactive thin resources, $\sum_{r\in C} z^*_r \geqslant 1-(1+\gamma)\lambda$. ◁

Positive Objective Function Value. We need to show that $\sum_{p\in P} y^*_p - \sum_{r\in R} z^*_r > 0$. By our setting of y^*_p and z^*_r, $\sum_{p\in P} y^*_p - \sum_{r\in R} z^*_r = \sum_{p\in P^+} y^*_p - \sum_{r\in R^+_f} z^*_r - \sum_{r\in R^+_t} z^*_r$.

First consider $\sum_{p\in P^+} y^*_p - \sum_{r\in R^+_f} z^*_r$. Since y^*_p and z^*_r have the same value $1-(1+\gamma)\lambda$ for $p\in P^+$ and $r\in R^+_f$, it suffices to bound $|P^+|-|R^+_f|$ from below. For each $r_f\in R^+_f$, by Claim 15, r_f has exactly one out-going edge to some player p in $G_{M\oplus\Pi}$. Since r_f is reachable from $B_{\leqslant\ell}\setminus src_\Pi$, so is p. That is, $p\in P^+$. We charge r_f to p. By Claim 15, each player in P^+ has in-degree at most 1 in $G_{M\oplus\Pi}$, so each of them is charged at most once. Note that players in $B_{\leqslant\ell}\setminus src_\Pi$ obviously belong to P^+ because they can be reached by themselves. Moreover, they have zero in-degree in $G_{M\oplus\Pi}$ as they are not matched by $M\oplus\Pi$, so they are not charged. Therefore, $|P^+|-|R^+_f| \geqslant |B_{\leqslant\ell}\setminus src_\Pi| \geqslant |B_{\leqslant\ell}|-|A_{\ell+1}|-|I|$. The last inequality is because Π is an optimal solution for $G_M(B_{\leqslant\ell}, A_{\ell+1}\cup I)$. In summary,

$$\sum_{p\in P^+} y^*_p - \sum_{r\in R^+_f} z^*_r \geqslant (1-(1+\gamma)\lambda)\left(|B_{\leqslant\ell}|-|A_{\ell+1}|-|I|\right). \tag{1}$$

Now consider $\sum_{r\in R^+_t} z^*_r$. By definition of inactive resources, R^+_t can be divided into three parts: those covered by $\mathcal{A}_{\leqslant\ell}\cup\mathcal{B}_{\leqslant\ell}$, those covered by $\mathcal{A}_{\ell+1}\cup\mathcal{I}$, and those covered by $\mathcal{B}'_{\ell+1} = \{e\in\mathcal{B}_{\ell+1} : v[R_e\cap R(\mathcal{A}_{\ell+1})] > b\lambda\}$. We handle these three parts separately.

Every edge in $\mathcal{A}_{\leqslant\ell}$ is blocked by some edges in $\mathcal{B}_{\leqslant\ell}$, so it has less than λ worth of thin resources not used by $\mathcal{B}_{\leqslant\ell}$. Every edge in $\mathcal{B}_{\leqslant\ell}$ is λ-minimal, so it covers less than 2λ worth of thin resources. Thus, $v[R(\mathcal{A}_{\leqslant\ell}\cup\mathcal{B}_{\leqslant\ell})] < \lambda|\mathcal{A}_{\leqslant\ell}| + 2\lambda|\mathcal{B}_{\leqslant\ell}| \leqslant \left(3+\frac{\beta}{\gamma}\right)\lambda|B_{\leqslant\ell}|$. The last inequality is by Lemma 12. Edges in $\mathcal{A}_{\ell+1}$ are $(1+\gamma)\lambda$-minimal, so each of them covers less than $(2+\gamma)\lambda$ worth of resources. Edges in $\mathcal{I}$ are λ-minimal, so each of them covers less than 2λ worth of thin resources. Therefore, $v[R(\mathcal{A}_{\ell+1}\cup\mathcal{I})] < (2+\gamma)\lambda|A_{\ell+1}| + 2\lambda|I|$. Edges in $\mathcal{B}'_{\ell+1}$ are λ-minimal, so $v[R(\mathcal{B}'_{\ell+1})] < 2\lambda|\mathcal{B}'_{\ell+1}| < \frac{4+2\gamma}{\beta}\lambda|A_{\ell+1}|$. The second inequality is by Lemma 13. Combining the above three parts gives

$$\sum_{r\in R^+_t} z^*_r = \sum_{r\in R^+_t} v_r < \left(3+\frac{\beta}{\gamma}\right)\lambda|B_{\leqslant\ell}| + 2\lambda|I| + \left(2+\gamma+\frac{4+2\gamma}{\beta}\right)\lambda|A_{\ell+1}|. \tag{2}$$

Combining (1) and (2) gives that

$$\begin{aligned}&\sum_{p\in P^+} y^*_p - \sum_{r\in R^+_f} z^*_r - \sum_{r\in R^+_t} z^*_r\\ &> \left(1-\left(4+\gamma+\frac{\beta}{\gamma}\right)\lambda\right)|B_{\leqslant\ell}| - \left(1+\left(1+\frac{4+2\gamma}{\beta}\right)\lambda\right)|A_{\ell+1}| - (1+(1-\gamma)\lambda)\,|I|.\end{aligned}$$

By the contrapositive assumption at the beginning of the proof of Lemma 14, $|A_{\ell+1}| < 2\mu|B_{\leqslant\ell}|$. Moreover, since no layer is collapsible, by Lemma 10(i), $|I| \leqslant \mu|B_{\leqslant\ell}|$. Substituting

these two inequalities into the above gives

$$\begin{aligned}&\sum_{p\in P^+} y_p^* - \sum_{r\in R_f^+} z_r^* - \sum_{r\in R_t^+} z_r^*\\ &> \left((1-3\mu) - \left(4+\gamma+\frac{\beta}{\gamma}+3\mu+\frac{(8+4\gamma)\mu}{\beta}-\gamma\mu\right)\lambda\right)|B_{\leqslant\ell}|.\end{aligned}$$

Let $\beta=\gamma^2$ and $\mu=\gamma^3$. We have

$$\sum_{p\in P^+} y_p^* - \sum_{r\in R_f^+} z_r^* - \sum_{r\in R_t^+} z_r^* > \left(\left(1-3\gamma^3\right) - \left(4+10\gamma+4\gamma^2+3\gamma^3-\gamma^4\right)\lambda\right)|B_{\leqslant\ell}|.$$

Recall that $\lambda=\frac{1}{4+\delta}$ for some $\delta>0$. As $\gamma\to 0$, $\frac{1-3\gamma^3}{4+10\gamma+4\gamma^2+3\gamma^3-\gamma^4}\to\frac{1}{4}$. Hence, there is a sufficiently small γ that makes $\frac{1-3\gamma^3}{4+10\gamma+4\gamma^2+3\gamma^3-\gamma^4}>\frac{1}{4+\delta}=\lambda$, thereby proving $\sum_{p\in P^+} y_p^* - \sum_{r\in R_f^+} z_r^* - \sum_{r\in R_t^+} z_r^* > 0$. Moreover, one can verify that $\frac{1}{\gamma}=O(\frac{1}{\delta})$. ◁

▶ **Lemma 17.** *Let $(M,\mathcal{E},\mathcal{I},(L_1,\ldots,L_\ell))$ be a state of the algorithm. If no layer is collapsible, then for $i\in[1,\ell-1]$, $|B_{i+1}|\geqslant\frac{\gamma^3}{1+\gamma}|B_{\leqslant i}|$.*

Proof. Fix an $i\in[1,\ell-1]$. Consider the period from the *most recent* construction of layer L_{i+1} until now. During this period, none of the layers below L_{i+1} has ever been collapsed; otherwise, L_{i+1} would be removed, contradiction. Hence, blocking edges in the layers below L_{i+1} have never been touched during this period. In other words, at the time L_{i+1} was constructed, the set of blocking edges in the layers below L_{i+1} was exactly $\mathcal{B}_{\leqslant i}$. Also the constant z_{i+1} is unchanged. By Lemma 14, $z_{i+1}\geqslant 2\mu|\mathcal{B}_{\leqslant i}|$. Although addable edges may be removed from the L_{i+1} during this period, there are still lots of addable edges left. By Lemma 10(ii), $|A_{i+1}|\geqslant z_{i+1}-\mu|B_{\leqslant i}|\geqslant\mu|B_{\leqslant i}|$. By Lemma 12, $|B_{i+1}|>\frac{1}{(1+\beta/\gamma)}|A_{i+1}|\geqslant\frac{\mu}{(1+\beta/\gamma)}|\mathcal{B}_{\leqslant i}|$. Recall that we set $\beta=\gamma^2$ and $\mu=\gamma^3$ in the proof of Claim 16. Replacing β by γ^2 and μ by γ^3 proves the lemma. ◀

▶ **Lemma 18.** *In $poly(m,n)\cdot n^{poly(\frac{1}{\delta})}$ time, the algorithm extends $M\cup\mathcal{E}$ to cover one more player.*

Given Lemma 17, Lemma 18 can be proved in a way similar to that of [1, 9]. We sketch the proof here. Consider all non-collapsible states ever reached by the algorithm. By non-collapsible, we mean that no layer is collapsible in this state. Let $h=\frac{\gamma^3}{1+\gamma}$. For each non-collapsible state $(M,\mathcal{E},\mathcal{I},(L_1,\ldots,L_\ell))$, we define its signature vector $(s_1,\ldots,s_\ell,\infty)$ where $s_i=\log_{1/(1-\mu)}\frac{|B_i|}{h^{i+1}}$. One can verify that the coordinates of the signature vector are non-decreasing, and that as the algorithm goes from one non-collapsible state to another, the signature vector decreases lexicographically. Moreover, the sum of the coordinates is bounded by U^2 where $U=\log n\cdot O(\frac{1}{\mu h}\log\frac{1}{h})$. Each signature can be regarded as a partition of an integer less than or equal to U^2. Summing up the number of partitions of an integer i over all $i\in[1,U^2]$, we get the upper bound of $n^{O(\frac{1}{uh}\log\frac{1}{h})}$ on the number of distinct signatures. Recall that $u=\gamma^3$, $h=\frac{\gamma^3}{1+\gamma}$, and $\frac{1}{\gamma}=O(\frac{1}{\delta})$. As a consequence, the number of non-collapsible states ever reached by the algorithm is bounded by $n^{poly(\frac{1}{\delta})}$. Between two consecutive non-collapsible states, there is one Build and at most $\log_{h+1} n$ Collapse, which take $poly(m,n)\cdot n^{poly(\frac{1}{\delta})}$ time in total. The total running time is thus $poly(m,n)\cdot n^{poly(\frac{1}{\delta})}$.

References

1. C. Annamalai, C. Kalaitzis, and O. Svensson. Combinatorial Algorithm for Restricted Max-Min Fair Allocation. *ACM Transactions on Algorithms*, 13(3):37:1–37:28, 2017.
2. A. Asadpour, U. Feige, and A. Saberi. Santa Claus meets hypergraph matchings. *ACM Transactions on Algorithms*, 8(3):24:1–24:9, 2012.
3. A. Asadpour and A. Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. In *Proceedings of the 39th ACM Symposium on Theory of Computing*, pages 114–121, 2007.
4. N. Bansal and M. Sviridenko. The Santa Claus problem. In *Proceedings of the 38th ACM Symposium on Theory of Computing*, pages 31–40, 2006.
5. M. Bateni, M. Charikar, and V. Guruswami. Max-Min Allocation via Degree Lower-bounded Arborescences. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 543–552, 2009.
6. I. Bezáková and Varsha Dani. Allocating indivisible goods. *SIGecom Exchanges*, 5(3):11–18, 2005.
7. D. Chakrabarty, J. Chuzhoy, and S. Khanna. On Allocating Goods to Maximize Fairness. In *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science*, pages 107–116, 2009.
8. S.-W. Cheng and Y. Mao. Integrality Gap of the Configuration LP for the Restricted Max-Min Fair Allocation. *CoRR*, abs/1807.04152, 2018. `arXiv:1807.04152`.
9. S.-W. Cheng and Y. Mao. Restricted Max-Min Fair Allocation. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming*, pages 37:1–37:13, 2018.
10. S. Davies, T. Rothvoss, and Y. Zhang. A Tale of Santa Claus, Hypergraphs and Matroids. *CoRR*, abs/1807.07189, 2018. `arXiv:1807.07189`.
11. U. Feige. On allocations that maximize fairness. In *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms*, pages 287–293, 2008.
12. B. Haeupler, B. Saha, and A. Srinivasan. New Constructive Aspects of the Lovász Local Lemma. *Journal of the ACM*, 58(6):28:1–28:28, 2011.
13. P.E. Haxell. A condition for matchability in hypergraphs. *Graphs and Combinatorics*, 11(3):245–248, 1995.
14. J.E. Hopcroft and R.M. Karp. An $n^{5/2}$ algorithm for maximum matching in bipartitie graphs. *SIAM Journal on Computing*, 2:225–231, 1973.
15. K. Jansen and L. Rohwedder. A note on the integrality gap of the configuration LP for restricted Santa Claus. *CoRR*, abs/1807.03626, 2018. `arXiv:1807.03626`.
16. J.K. Lenstra, D.B. Shmoys, and É. Tardos. Approximation Algorithms for Scheduling Unrelated Parallel Machines. In *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, pages 217–224, 1987.
17. L. Polacek and O. Svensson. Quasi-polynomial Local Search for Restricted Max-Min Fair Allocation. In *39th International Colloquium on Automata, Languages, and Programming*, pages 726–737, 2012.
18. B. Saha and A. Srinivasan. A New Approximation Technique for Resource-Allocation Problems. In *Proceedings of the 1st Symposium on Innovations in Computer Science*, pages 342–357, 2010.

Circuit Lower Bounds for MCSP from Local Pseudorandom Generators

Mahdi Cheraghchi
Department of Computing, Imperial College London, London, UK
http://mahdi.ch
m.cheraghchi@imperial.ac.uk

Valentine Kabanets
School of Computing Science, Simon Fraser University, Burnaby, BC, Canada
https://www.cs.sfu.ca/~kabanets/
kabanets@cs.sfu.ca

Zhenjian Lu
School of Computing Science, Simon Fraser University, Burnaby, BC, Canada
zhenjian_lu@sfu.ca

Dimitrios Myrisiotis
Department of Computing, Imperial College London, London, UK
d.myrisiotis17@imperial.ac.uk

Abstract

The Minimum Circuit Size Problem (MCSP) asks if a given truth table of a Boolean function f can be computed by a Boolean circuit of size at most θ, for a given parameter θ. We improve several circuit lower bounds for MCSP, using pseudorandom generators (PRGs) that are local; a PRG is called *local* if its output bit strings, when viewed as the truth table of a Boolean function, can be computed by a Boolean circuit of small size. We get new and improved lower bounds for MCSP that almost match the best-known lower bounds against several circuit models. Specifically, we show that computing MCSP, on functions with a truth table of length N, requires

- $N^{3-o(1)}$-size de Morgan formulas, improving the recent $N^{2-o(1)}$ lower bound by Hirahara and Santhanam (CCC, 2017),
- $N^{2-o(1)}$-size formulas over an arbitrary basis or general branching programs (no non-trivial lower bound was known for MCSP against these models), and
- $2^{\Omega\left(N^{1/(d+2.01)}\right)}$-size depth-$d$ AC^0 circuits, improving the superpolynomial lower bound by Allender et al. (SICOMP, 2006).

The AC^0 lower bound stated above matches the best-known AC^0 lower bound (for PARITY) up to a small *additive* constant in the depth. Also, for the special case of depth-2 circuits (i.e., CNFs or DNFs), we get an almost optimal lower bound of $2^{N^{1-o(1)}}$ for MCSP.

2012 ACM Subject Classification Theory of computation → Circuit complexity; Theory of computation → Pseudorandomness and derandomization

Keywords and phrases minimum circuit size problem (MCSP), circuit lower bounds, pseudorandom generators (PRGs), local PRGs, de Morgan formulas, branching programs, constant depth circuits

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.39

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at https://eccc.weizmann.ac.il/report/2019/022/.

Acknowledgements We thank the anonymous ICALP'19 reviewers for their excellent comments.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 39; pp. 39:1–39:14

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

1 Introduction

Given the truth table of some Boolean function f and a size parameter θ, the minimum circuit size problem (MCSP) asks whether f can be computed by a circuit of size at most θ. Understanding the exact complexity of MCSP is an important open problem in computational complexity theory, dating back to the 1950s [20].

It is easy to see that MCSP is in NP. A popular conjecture is that MCSP is also NP-hard. However, despite serious efforts over the years, such a proof is still unknown. Given that it is difficult to show that MCSP is hard, perhaps the problem is easy? It turns out that this cannot be the case under some plausible cryptographic assumptions. More specifically, it is known that if one-way functions exist, then MCSP is not in P [11]. As proving an *unconditional* lower bound for MCSP seems far beyond the reach of currently known techniques, can we at least prove unconditional lower bounds for MCSP against some restricted computational models?[1]

Two of the most studied restricted computational models in complexity theory are constant-depth circuits (AC^0) and de Morgan formulas. For AC^0 circuits, the best-known lower bound is about PARITY: PARITY on N variables requires depth-d AC^0 circuits of size $2^{\Omega\left(N^{1/(d-1)}\right)}$ [6]. For de Morgan formulas, the state-of-the-art lower bound is almost cubic, namely $N^{3-o(1)}$, for some polynomial-time computable function [7, 18, 19, 5].

Notably, there are also lower bounds against these models for MCSP. Allender et al. [2] showed that MCSP, on functions represented as a truth table of length N, cannot be computed by polynomial-size constant-depth AC^0 circuits. In fact, by a more careful analysis of their argument, one can get a lower bound of $2^{N^{1/(c\cdot d+O(1))}}$, for a constant $c \geq 2$. However, such a lower bound still has a worse dependence on the depth compared to the PARITY lower bound. For de Morgan formulas, Hirahara and Santhanam [9] showed that computing MCSP requires de Morgan formulas of size $N^{2-o(1)}$.

Given these two MCSP lower bounds and the best-known lower bounds against these two models, it is natural to ask whether we can get MCSP lower bounds against small-depth circuits and de Morgan formulas that match the state-of-the-art lower bounds against these models. More specifically, can we show that computing MCSP requires depth-d AC^0 circuits of size $2^{N^{1/(d+O(1))}}$ and de Morgan formulas of size $N^{3-o(1)}$? Furthermore, can we show lower bounds for MCSP against some other restricted models that match their state-of-the-art lower bounds? In this paper, we answer these questions in the affirmative.

1.1 Our results

Our first result is an almost-cubic de Morgan formula lower bound for MCSP.

▶ **Theorem 1.** *Any de Morgan formula computing* MCSP *on truth tables of length N must have size at least $N^3/2^{O\left(\log^{2/3} N\right)}$.*

We also get almost-quadratic lower bounds against formulas over an arbitrary basis as well as general branching programs; these almost match the best-known lower bounds against these models [12].

▶ **Theorem 2.** *Let C be either a formula over any basis or a branching program that computes* MCSP *on truth tables of length N. Then C must have size at least $N^2/2^{O\left(\sqrt{\log N}\right)}$.*

[1] A recent line of research on *hardness magnification* [16, 14] provides another motivation for proving relatively weak lower bounds for restricted circuit models against certain "gap variants" of MCSP. Such lower bounds are shown to imply much stronger (superpolynomial) lower bounds.

For small-depth circuits, we have the following improved lower bound for MCSP, which its dependence on the depth matches the one in the PARITY lower bound, up to a small additive constant.

▶ **Theorem 3.** *For every $d > 2$ and every constant $\gamma > 0$, any depth-d AC^0 circuit computing* MCSP *on truth tables of length N must have size $2^{\Omega\left(N^{1/(d+2+\gamma)}\right)}$.*

For the special case of depth-2 circuits, we can have an almost optimal lower bound.

▶ **Theorem 4.** *Any CNF or DNF computing* MCSP *on truth tables of length N must have size $2^{N/\widetilde{O}\left(\log^2 N\right)}$.*

Also, in this paper (in the full version), we give a fine-grained analysis of the approach of obtaining MCSP lower bounds from average-case hardness via the Nisan-Wigderson framework.

1.2 Our techniques

For a class $\mathfrak{C}$ of N-variate Boolean functions, a pseudorandom generator (PRG) against $\mathfrak{C}$ is a deterministic efficiently-computable function G mapping short binary strings (seeds) to longer binary strings so that every function in $\mathfrak{C}$ accepts G's output on a uniformly random seed with about the same probability as that for an actual uniformly random string. A key notion in this work is that of a local PRG. We say that a PRG is *local* if its N-bit output (viewed as the truth table of some function) has small circuit complexity. More precisely, for any fixed seed to the PRG, there exists a small circuit such that, given $j \in [N]$ as an input, the circuit computes the j-th bit of the PRG output, where the size of the circuit is measured relative to its input length, namely $\log N$.

Local PRGs in the context of MCSP (and related problems) have been studied in previous works (see, e.g., [2, 15, 9, 8]). In this work, we refine the previous approaches, and obtain stronger circuit lower bounds by establishing strong locality properties of certain PRGs.[2]

MCSP lower bounds from local PRGs. Suppose we have a local PRG against some class of circuits $\mathfrak{C}$ of size s, and we want to show that MCSP cannot be computed by any size-s circuit in $\mathfrak{C}$. Suppose some size-s circuit C in $\mathfrak{C}$ computes MCSP. Using the fact that a random function has almost maximum circuit complexity, we have that C will output FALSE on most of its inputs (by setting the size parameter θ to be a non-trivial quantity that is asymptotically smaller than $2^n/n$, where n is the input length of the function). If we replace the uniformly random inputs with the outputs of the local PRG, then, by the definition of PRGs, C will still output FALSE with large probability. However, since the PRG is local, all of its outputs have circuit complexity smaller than the size parameter θ, and hence must be accepted by C. A contradiction.

To get a strong lower bound, we would like to make the above argument to work for large s. Note that the local complexity of the PRG, $\lambda(s)$, is a function on the size of the circuit C, and we need this local complexity to be "non-trivial" in order to reach a contradiction. Therefore, we want to choose s so that this local complexity remains asymptotically smaller than $2^n/n$. As a result, the final lower bound (i.e., the largest s that we can choose) is determined by the local complexity λ. So the main question we study in our paper is: What is the smallest local complexity of a PRG against a given circuit class?

[2] Note that, as one of our reviewers pointed out, the notion of a local PRG can be also found in the context of cryptography [4], where a PRG $G : \{0,1\}^n \to \{0,1\}^m$ is called *k-local*, for some constant $k > 0$, if every output PRG bit $G(x)_j$, for any $x \in \{0,1\}^n$ and $j \in [m]$, depends only on k input bits $x_{i_1}, \ldots, x_{i_k}$, for $i_1, \ldots, i_k \in [n]$. In our work, however, locality refers to the circuit complexity of the PRG at hand and the output bits of our PRGs may depend on a superconstant number of input bits.

MCSP lower bound against de Morgan formulas. Our formula lower bound for MCSP is obtained by applying the framework described above to a local PRG against formulas. The state-of-the-art PRG against formulas is given by Impagliazzo, Meka, and Zuckerman [10], which we refer to as the IMZ PRG. Their PRG has a seed length of $s^{1/3+o(1)}$ for size s formulas (note that such a PRG is useful against sub-cubic formulas only). If we want to utilize the IMZ PRG to get an MCSP lower bound against formulas, we will need to argue that the IMZ PRG is local.

In fact, in order to get an almost-cubic lower bound, we will need such a PRG to be strongly local in the sense that any single output bit of the PRG (on any given fixed seed) can be computed by a circuit of size comparable to its seed length, which is $s^{1/3+o(1)}$. However, by inspecting the construction, the IMZ PRG does not seem to have such a property, and a straightforward implementation seems to require a circuit of size at least $s^{2/3}$ (see the full version for more details), which yields a weaker lower bound for MCSP.

To overcome this issue, we present an alternative PRG useful against sub-cubic formulas which is strongly local. The construction of this PRG can be viewed as a modification of the IMZ PRG. At a high level, it is based on the Ajtai-Wigderson construction [1], which is a framework for constructing PRGs against computations that can be simplified under (pseudo)random restrictions. This framework is then combined with the ideas of reducing (recycling) random bits using an extractor, by exploiting communication bottlenecks in computations [13]. Our modification, particularly the utilization of the Ajtai-Wigderson construction, allows us to compute any output bit of the PRG efficiently by reducing the number of calls to the extractor. Using some crucial observations on the circuit complexity of certain pseudorandom objects, we get a PRG that is locally computable by a $s^{1/3+o(1)}$-size circuit.[3]

MCSP lower bounds against formulas over an arbitrary basis or branching programs. The MCSP lower bounds against formulas over an arbitrary basis or branching programs are obtained similarly to those for de Morgan formulas. The idea is to construct strongly local PRGs against these models by modifying the PRGs in [10]. Then, by applying our "MCSP circuit lower bounds from local PRGs" framework, we get the desired lower bounds.

MCSP lower bounds against AC^0. We use a local PRG against AC^0 to get MCSP lower bounds. To get a lower bound matching the one in Theorem 3, we can use the state-of-the-art PRG against AC^0 by Trevisan and Xue [21], which has a seed length of $(\log s)^{d+O(1)}$ for size-s depth-d AC^0 circuits. By a careful analysis of the construction of this PRG, we can show that the Trevisan-Xue PRG is strongly local and can be used to get an MCSP lower bound that is close to the one stated in Theorem 3. However, in this paper, we will present a more direct proof of such a lower bound by using the pseudorandom switching lemma for constant-depth circuits, which is due to Trevisan and Xue [21], as well, and is a key ingredient in their PRG.

The idea is to show that for any small-depth circuit of size less than the claimed lower bound, there is some locally computable restriction that turns the circuit into a constant function, but leaves many variables unrestricted. However, MCSP cannot be constant under such a restriction, because depending on the partial assignment to the unrestricted variables, the resulting input function (which is composed of the restriction and the partial assignment) can be either easy or hard. Such an approach based on pseudorandom restrictions can also be applied to depth-2 circuits and yield almost optimal CNF (and DNF) MCSP lower bounds.

[3] It is also possible to use the original IMZ PRG to obtain an almost-cubic formula lower bound for MCSP. We can show that the IMZ PRG, although not fully strongly local, is "almost strongly local" in the sense that *most* of its outputs have very small circuit complexity; see the full version.

1.3 Remainder of the paper

We give the necessary background in Section 2. In Section 3, we describe our framework of using local PRGs to obtain lower bounds for MCSP. We prove the almost-cubic de Morgan formula lower bound for MCSP (Theorem 1) in Section 4, and the almost-quadratic lower bound against formulas over an arbitrary basis and branching programs (Theorem 2) in Section 5. The improved AC^0 lower bounds for MCSP (Theorem 3 and Theorem 4) are proved in Section 6. Finally, we give some open problems in Section 7. Due to space limitations we relegated some material to the full version, like some omitted proofs and the framework of proving MCSP lower bounds from average-case hardness.

2 Preliminaries

2.1 Notation

For any computational model, we use the term *size* to refer to its complexity measure. For example, if the model is circuits of some fixed depth, then the size is the number of gates in the circuit.

For a positive integer n, that is a power of two, we use the following notation: $[n]$ denotes the set $\{1, 2, \ldots, n\} \cong \{0,1\}^{\log n}$, $\mathbb{F}_n$ denotes the field with n elements, where the elements in $\mathbb{F}_n$ are represented by $(\log n)$-bit strings, U_n denotes the uniform distribution over $\{0,1\}^n$, and, for a function $f\colon \{0,1\}^n \to \{0,1\}$, $\mathtt{tt}(f) \in \{0,1\}^{N=2^n}$ denotes the truth table of f.

2.2 Pseudorandomness

▶ **Definition 5** (Pseudorandom generators). *Let $G\colon \{0,1\}^r \to \{0,1\}^n$ be a function, $\mathcal{F}$ be a class of Boolean functions, and $0 < \varepsilon < 1$. We say that G is a* pseudorandom generator of seed length r that ε-fools $\mathcal{F}$ *if, for every function $f \in \mathcal{F}$, it is the case that*

$$\left|\mathbb{E}_{z\sim\{0,1\}^r}[f(G(z))] - \mathbb{E}_{x\sim\{0,1\}^n}[f(x)]\right| \leq \varepsilon.$$

▶ **Definition 6** (k-wise independence). *A distribution X over $[m]^n$ is called* k-wise independent *if for any $1 \leq i_1 \leq i_2 \leq \cdots \leq i_k \leq n$ and every $b_1, b_2, \ldots, b_k \in [m]$, we have*

$$\mathbf{Pr}[X_{i_1} = b_1, X_{i_2} = b_2, \ldots, X_{i_k} = b_k] = m^{-k}.$$

The following simple fact (proved in the full version) will be convenient for us.

▶ **Lemma 7.** *Let X and Y be two random variables that take values in $\{0,1\}$ and $\mathcal{E}$ be some event. If $|\mathbb{E}[X \mid \mathcal{E}] - \mathbb{E}[Y \mid \mathcal{E}]| \leq \varepsilon_1$ and $\mathbf{Pr}[\neg\mathcal{E}] \leq \varepsilon_2$, then $|\mathbb{E}[X] - \mathbb{E}[Y]| \leq \varepsilon_1 + \varepsilon_2$.*

2.3 Random restrictions

A restriction for a n-variate Boolean function f, usually denoted as $\rho \in \{0,1,*\}^n$, specifies a way of fixing the values of some subset of variables for f. We denote by f_ρ the restricted function after the variables are restricted according to ρ, and denote by $\rho^{-1}(*)$ the set of unrestricted variables. A random restriction is then a distribution over restrictions, which can be specified by a pair $(\sigma, \beta) \in \{0,1\}^n \times \{0,1\}^n$, where σ (as a characteristic string) specifies the set of unrestricted variables, and β specifies the values for fixing the restricted variables. We say that a random restriction (or random selection) is p-regular if each variable is left unrestricted with probability p. One way to generate a p-regular random restriction is to leave each variable, independently, unrestricted with probability p, and otherwise assign to it a 0 or

a 1, uniformly at random. Such a random restriction is called a *(truly) p-random restriction*. Note that to sample such a restriction, we can first pick a string in $\{0,1\}^{n\cdot\log(1/p)} \cong [1/p]^n$ to specify the selection of the unrestricted variables, where a coordinate is unrestricted if and only if all of its corresponding $\log(1/p)$ bits are 0, and then a string in $\{0,1\}^n$ to specify the values assigned to each of the restricted variables. So sampling a restriction in this way requires $n \cdot \log(1/p) + n$ random bits.

We can also generate a restriction in a pseudorandom manner, which may use fewer random bits. For example, one way to do this is to use a limited-independence distribution over $[1/p]^n$, so that each variable is left unrestricted with probability p and any k variables are independent. Also, we can let each variable be assigned a 0 or a 1, uniformly at random, in a way such that any k of the variables are independent; this again can be done using a k-wise independent distribution on $\{0,1\}^n$.

2.4 Simple facts about Boolean circuits

▶ **Proposition 8.** *A Boolean circuit of size s can be specified using $O(s \log s)$ bits. Hence there are at most $2^{O(s \log s)} = s^{O(s)}$ distinct circuits of size at most s.*

▶ **Theorem 9** ([17])**.** *The fraction of functions on n variables that have a circuit of size less than $2^n/(3n)$ is $o(1)$.*

The following lemma is proved in the full version.

▶ **Lemma 10.** *For any integer $t > 0$, there exists a circuit C of size $\widetilde{O}(t)$ such that, given any string $x \in \{0,1\}^t$, the circuit does the following: If $x = 0^t$, then C outputs $(0, 0^{\log t})$ and if $x \neq 0^t$, then C outputs $(1, q)$, where $q \in \{0,1\}^{\log t}$ is the index of the first bit in x that is not 0.*

The following circuit upper bound for the addressing (storage access) function is well-known (see, e.g., [22]); we include a proof, in the full version, for completeness.

▶ **Lemma 11.** *For any integers $t, m > 0$, there exists a circuit of size $O(tm)$ such that, given any string $y = (y_1, y_2, \ldots, y_t)$, where $y_i \in \{0,1\}^m$ for each i, and an index $i \in \{0,1\}^{\log t}$, the circuit outputs y_i.*

3 The "MCSP circuit lower bounds from local PRGs" framework

We first describe how to use local PRGs to obtain MCSP lower bounds.

▶ **Definition 12** (Local PRGs)**.** *Let $\lambda\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be a size function. For any Boolean computational model and size $s > 0$, we say that a function $G\colon \{0,1\}^{r=r(N,s)} \to \{0,1\}^N$ is a $(N, s, \lambda(N,s))$-local PRG against the model if G 1/3-fools every device f on N variables of size s in the model; that is,*

$$\left|\mathbb{E}_{z\sim\{0,1\}^r}[f(G(z))] - \mathbb{E}_{x\sim\{0,1\}^N}[f(x)]\right| \leq 1/3,$$

and for any seed $z \in \{0,1\}^r$, the function $g\colon \{0,1\}^{\log N} \to \{0,1\}$, defined as $g_z(j) = G(z)_j$, can be computed by a general circuit of size at most $\lambda(N,s)$.

Note that $\lambda(N,s)$ is at least $r(N,s)$, by a counting argument (neglecting $\log \lambda(N,s)$ factors). This is to ensure that, for any function g, on n variables, which may be output by G, there is some $\lambda(N,s)$-size circuit that computes g.

▶ **Theorem 13.** *There exists a constant $c > 0$ such that the following holds. For any computational model, let s be such that* MCSP *on truth tables of length N can be computed by a device of size s in the model. If there exists some $(N, s, \lambda(N, s))$-local PRG against the model, then $\lambda(N, s) \geq \frac{N}{c \log N}$.*

Proof. Let C be a device in the computational model such that C computes MCSP on truth tables of length N. Suppose C has size s, and let G be a $(N, s, \lambda(N, s))$-local PRG against C with some seed length r.

For the sake of contradiction, suppose that $\lambda(N, s) < \frac{N}{c \log N}$. On the one hand, since most functions require circuits of size greater than $\frac{N}{c \log N}$ (Theorem 9) and C computes MCSP, we have $\mu = \mathbf{Pr}_{\mathtt{tt}(f) \sim \{0,1\}^N}[C(\mathtt{tt}(f), \lambda(N, s)) = 0] \geq 1/2$. Also, since G fools C, we have $\mathbf{Pr}_{z \sim \{0,1\}^r}[C(G(z), \lambda(N, s)) = 0] \geq \mu - 1/3 \geq 1/6$. On the other hand, because G is $(N, s, \lambda(N, s))$-local, we must have $C(G(z), \lambda(N, s)) = 1$, for every z. A contradiction. ◀

4 Almost-cubic de Morgan formula lower bounds for MCSP

In this section, we present our almost-cubic de Morgan formula lower bound for MCSP. By saying "formula" within this section, we refer to formulas over the de Morgan basis (AND, OR, and NOT). By *size* of a formula, we mean its usual leaf complexity, i.e., the number of leaves in the tree representation of the formula.

▶ **Theorem 14** (Theorem 1 restated)**.** *Any de Morgan formula computing* MCSP *on truth tables of length N must have size at least $N^3/2^{O\left(\log^{2/3} N\right)}$.*

We will construct a strongly local PRG useful against sub-cubic formulas. That is, given as input an index j, the j-th bit of the PRG can be computed by a circuit of size that is comparable to its seed length, which in our case is around $s^{1/3}$ for size s formulas.

▶ **Lemma 15.** *For any $s \geq N$, there exists a $\left(N, s, s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)}\right)$-local PRG against de Morgan formulas.*

Given the local PRG in Lemma 15, we can combine it with our Theorem 13 to obtain a formula lower bound for MCSP.

Proof of Theorem 14. Let s be such that MCSP on truth tables of length N can be computed by some formula of size s. We can assume that $s \leq N^3$ since, otherwise, the result trivially holds. By Theorem 13 and Lemma 15, we have $s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)} \geq N/(c \log N)$; then, $s \geq N^3 / \left(2^{O\left(\log^{2/3} N\right)} c^3 \log^3 N\right)$. ◀

The rest of this section is devoted to proving Lemma 15.

4.1 Almost-linear-size k-independent generators

The PRG in Lemma 15 will use k-wise independent distributions. Recall that a multidimensional distribution is called *k-wise independent* if any k coordinates of the distribution are uniformly distributed (see Definition 6). We say that a function G is a *k-independent generator* if, for random inputs, the distribution of the outputs of G is k-wise independent.

We will need a k-independent generator that is strongly local.

▶ **Lemma 16.** *For any integer $k > 0$, there exists a k-independent generator $G\colon \{0,1\}^r \to [m]^N$, with $r = k \cdot \max\{\log N, \log m\}$, such that the following holds. There exists a circuit of size $k \cdot \max\{\widetilde{O}(\log N), \widetilde{O}(\log m)\}$ such that, given $j \in \{0,1\}^{\log N}$ and a seed $z \in \{0,1\}^r$, the circuit computes the j-th coordinate of $G(z)$ (as an element of $\{0,1\}^{\log m}$).*

The above k-independent generator is constructed using finite fields (see the full version). Its efficiency crucially depends on the fact that finite field arithmetic can be done using *almost linear-size* Boolean circuits.

4.2 Almost-linear-size extractors

Our PRG will make use of randomness extractors. Here, we describe an extractor that is computable by a circuit of size that is almost linear in the length of its input. We start by reviewing the definitions of some basic notions regarding extractors.

▶ **Definition 17** (ε-closeness and statistical distance). *Let $0 \leq \varepsilon \leq 1$. We say two distributions X and Y (over some universe D) are* ε-close *if their* statistical distance, *defined as*

$$\max_{T:D\to\{0,1\}} \left|\mathbf{Pr}[T(X)=1] - \mathbf{Pr}[T(Y)=1]\right|,$$

is at most ε.

▶ **Definition 18** (Min-entropy). *Let X be a random variable. The* min-entropy *of X, denoted by $H_\infty(X)$, is the largest real number k such that $\mathbf{Pr}[X=x] \leq 2^{-k}$ for every x in the range of X. If X is a distribution over $\{0,1\}^\aleph$ with $H_\infty(X) \geq k$, then X is called a* $(\aleph, k)$-source.

▶ **Definition 19** (Extractors). *A function $E\colon \{0,1\}^\aleph \times \{0,1\}^d \to \{0,1\}^m$ is an* (k,ε)-extractor *if, for any $(\aleph,k)$-source X, the distribution $E(X,U_d)$ is ε-close to U_m.*

We now state the extractor, which for a high min-entropy source extracts a constant fraction of min-entropy, using seeds of polylogarithmic length. The construction and circuit complexity of this extractor are presented in the full version.

▶ **Lemma 20** (Almost-linear-size extractors, following [13]). *There exists some randomness extractor $E\colon \{0,1\}^\aleph \times \{0,1\}^d \to \{0,1\}^m$ that is an $(\aleph/2,\varepsilon)$-extractor with $m = \Omega(\aleph)$ and $d = \mathsf{polylog}(\aleph/\varepsilon)$. Moreover, E can be computed by a circuit of size $\aleph \cdot \mathsf{polylog}(\aleph/\varepsilon)$.*

4.3 Strongly local PRG useful against sub-cubic de Morgan formulas

For a formula F, let $L(F)$ denote the size (which is measured by the number of leaves) of F. We need the following pseudorandom shrinkage lemma for de Morgan formulas, which says that there exists a p-regular restriction, where the unrestricted variables are selected pseudorandomly and the restricted variables are fixed truly-randomly, such that *with high probability* the size of the restricted formula will "shrink" by a factor of p^2.

▶ **Lemma 21** (Pseudorandom shrinkage lemma, Lemma 4.8 of [10][4]). *There exists a constant $c_0 > 0$ such that the following holds. For any constant $c > c_0$, any $s \geq N$, $p \geq s^{-1/2}$, and any de Morgan formula F on N variables of size s, there exists a p-regular pseudorandom selection $\mathcal{D}$ over N variables, that is samplable using $r = 2^{O(\log^{2/3} s)}$ random bits, such that*

$$\mathbf{Pr}_{\sigma\sim\mathcal{D}, x\sim\{0,1\}^N}\left[L\big(F_{(\sigma,x)}\big) \geq 2^{3\cdot c\cdot \log^{2/3} s} \cdot p^2 \cdot s\right] \leq s^{-c}.$$

Moreover, there exists a circuit of size $2^{O(\log^{2/3} s)}$ such that, given $j \in \{0,1\}^{\log N}$ and a seed $z \in \{0,1\}^r$, the circuit computes the j-th bit of $\mathcal{D}(z)$.

[4] The pseudorandom shrinkage lemma in [10] is not stated in this form, but rather selects the unrestricted variables and fixes the restricted variables both pseudorandomly (based on limited independence). However, our version here follows from the proof of the original version in Section 4.2 of [10] by noting that limited-independence distributions can be computed locally.

We are now ready to show our PRG in Lemma 15.

Proof of Lemma 15. The construction is as follows: We first sample a p-regular pseudorandom selection from Lemma 21. Then, we fill the star coordinates, specified by the pseudorandom selection, in the output string with the output of some extractor which takes a min-entropy source sample and a short seed (in fact, it is the output of some limited-independence generator that takes the output of the extractor as a seed). We then sample another pseudorandom selection, and fill the star coordinates specified by this pseudorandom selection but this time only for those that have not been filled in previous steps, again with the output of the same extractor using the *same min-entropy source sample* but a *different short seed*. We continue this way until all the coordinates are filled.

More formally, our PRG uses the following parameters:[5]

- $p = 1/s^{1/3}$, the expected fraction of unrestricted variables in each of the pseudorandom selections;
- $\varepsilon = 1/\mathsf{poly}(N)$ and $\varepsilon_0 = \varepsilon/(10t)$, which specify the error of the PRG;
- $t = \ln(2N/\varepsilon)/p = s^{1/3} \cdot O(\log N)$, the number of steps needed so that all the coordinates will be filled with probability except $\varepsilon/2$;
- $s_0 = p^2 \cdot s \cdot 2^{O\left(\log^{2/3} s\right)} = s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)}$, the size of the formula after being simplified by a pseudorandom restriction;
- $k \geq s_0 = s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)}$, the amount of independence needed to fool the simplified formula, and $r_k = k \cdot \log N$ the seed length for the k-independent generator;
- $\aleph$, the length of the min-entropy source for the extractor, which is such that $\aleph \geq 2 \cdot (\log(1/\varepsilon_0) + c \cdot s_0 \cdot \log s_0)$, where $c > 0$ is some constant, and that $\aleph = \Omega(r_k)$. We can take $\aleph = s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)}$;
- $d = \mathsf{polylog}(\aleph/\varepsilon_0) = \mathsf{polylog}(N)$, the seed length of the extractor;
- $\ell = 2^{O(\log^{2/3} s)}$, the number of random bits for sampling a pseudorandom selection.

Construction. The PRG takes a seed $(X, Y_1, Y_2, \ldots, Y_t, \gamma_1, \gamma_2, \ldots, \gamma_t) \in \{0,1\}^r$, where

- $X \in \{0,1\}^{\aleph}$ is the min-entropy source sample of an extractor,
- $Y_i \in \{0,1\}^{\mathsf{polylog}(N)}$, for each $i \in [t]$, is the seed of an extractor, and
- $\gamma_i \in \{0,1\}^{\ell}$, for each $i \in [t]$, is the seed for sampling a pseudorandom selection.

The construction of the PRG proceeds in the following two stages.

1. Compute a sequence of t p-regular pseudorandom selections $\sigma_1, \ldots, \sigma_t$, using Lemma 21, with the seeds $\gamma_1, \ldots, \gamma_t$. Below, we denote the star coordinates in σ_i by $\sigma_i^{-1}(*)$. Let $S_1, \ldots, S_t \subseteq [N]$ be t disjoint sets defined by

$$S_i = \sigma_i^{-1}(*) \setminus (S_1 \cup S_2 \cup \cdots \cup S_{i-1}).$$

2. Define $Z_1, Z_2, \ldots, Z_t \in \{0,1\}^N$ by

$$Z_i = G_k(E(X, Y_i)),$$

where $E\colon \{0,1\}^{\aleph} \times \{0,1\}^d \to \{0,1\}^{\Omega(\aleph)}$ is an $(\aleph/2, \varepsilon_0)$-extractor and $G_k\colon \{0,1\}^{r_k} \to \{0,1\}^N$ is a k-independent generator. The final output of our PRG is the binary string that has the values $Z_i|_{S_i}$ in the positions indexed by S_i, for all $i \in [t]$, where $Z_i|_{S_i}$ denotes the bit values of Z_i projected to the set S_i. (We fix those positions that are not in any of the S_i's to be 0.) Stage 2 of the PRG construction is depicted in Figure 1.

[5] In fact, there are mainly two types of parameters here. Those that are close to $s^{1/3}$, which are $1/p, t, s_0, k, N$, and those that are close to $N^{o(1)}$, which are d and ℓ.

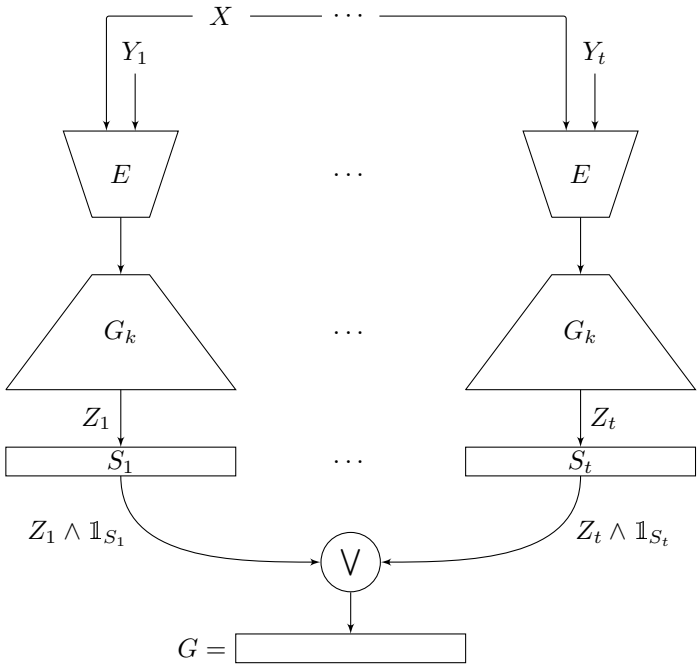


Figure 1 Construction of the PRG in Lemma 15, Stage 2. For each $i \in [t]$, $\mathbb{1}_{S_i} \in \{0,1\}^N$ denotes the characteristic Boolean vector of the set S_i, where $S_i \subseteq [N]$ is the set of star coordinates in the i-th pseudorandom selection that did not appear in the preceding sets $S_1, \ldots, S_{i-1}$. Also, $\wedge$ denotes a coordinate-wise AND operation (i.e., coordinate-wise *multiplication* of Boolean vectors) and $\bigvee$ is a coordinate-wise OR operation.

Correctness. Next, we show that the above PRG ε-fools N-variate formulas f of size s. First, note that, by our choice of t, with probability except $\varepsilon/2$, $S_1 \cup S_2 \cup \cdots \cup S_t$ covers all coordinates. For the rest of the argument, we will assume that this is the case. By Lemma 7, conditioning on this assumption contributes at most $\varepsilon/2$ to the error of the constructed PRG.

We continue the correctness analysis using a *hybrid argument*. Let G denote the distribution given by the PRG described above. Let U be the uniform distribution. Note that if in the above construction we replace Z_i, for all $i \in [t]$, with U, then we would get a uniform distribution. Now we can start from there and gradually replace U with the Z_i's step-by-step for a total of t steps. We will argue that after each replacement step, the expected value of the function does not change by much. Let A_i be the distribution so that we have replaced U with Z_i in the first i steps. That is,

$$A_i = \left(Z_1|_{S_1}, \ldots, Z_i|_{S_i}, U|_{S_{i+1}}, \ldots, U|_{S_t} \right) = \left(Z_1|_{S_1}, \ldots, Z_i|_{S_i}, U|_{S_{i+1} \cup \cdots \cup S_t} \right).$$

For the sake of contradiction, suppose there exists an N-variate size-s formula f such that

$$|\mathbb{E}[f(U)] - \mathbb{E}[f(G)]| = |\mathbb{E}[f(A_0)] - \mathbb{E}[f(A_t)]| > \varepsilon/2.$$

By the triangle inequality, there exists an $0 \leq i < t$ such that

$$|\mathbb{E}[f(A_i)] - \mathbb{E}[f(A_{i+1})]| > \varepsilon/(2t). \tag{1}$$

Let us say that the expectations in Equation (1) are over $\sigma_1, \ldots, \sigma_{i+1}, Y_1, \ldots, Y_{i+1}, X, U$, and we may remove the absolute value without loss of generality. Then, we have

$$\mathbb{E}_{\substack{\sigma_1,\ldots,\sigma_i,\\ Y_1,\ldots,Y_i,\\ X}} \left[\mathbb{E}_{\sigma_{i+1},Y_{i+1},U} \left[f(A_i) \right] - \mathbb{E}_{\sigma_{i+1},Y_{i+1},U} \left[f(A_{i+1}) \right] \right] > \varepsilon/(2t). \tag{2}$$

Let $W_i = (\sigma_1, \ldots, \sigma_i, Y_1, \ldots, Y_i, X)$, and let f' be the random function (where the randomness is over W_i) defined as $f' = f(Z_1|_{S_1}, \ldots, Z_i|_{S_i}, \quad \cdot \quad)$. That is, f' is the restricted function after the first i steps. Then, the left hand side of Equation (2) becomes

$$\mathbb{E}_{W_i}\left[\mathbb{E}_{\sigma_{i+1},U}\left[f'(U|_{S_{i+1}}, U|_{\ S_{i+2}\cup\cdots\cup S_t})\right] - \mathbb{E}_{\sigma_{i+1},Y_{i+1},U}\left[f'(Z_{i+1}|_{S_{i+1}}, U|_{\ S_{i+2}\cup\cdots\cup S_t})\right]\right]. \tag{3}$$

Note that, at this point, we can view $\rho_{i+1} = (\sigma_{i+1}, U)$ as a pseudorandom restriction (in the sense of Lemma 21) applied to f'. Next, let f'' be the random function defined as the restricted function of f' under ρ_{i+1} (note that the randomness is over W_i, and also the pseudorandom restriction ρ_{i+1}). Now Equation (3) becomes

$$\mathbb{E}_{W_i,\rho_{i+1}}\left[\mathbb{E}_U\left[f''(U)\right] - \mathbb{E}_{Y_{i+1}}\left[f''(Z_{i+1})\right]\right]. \tag{4}$$

Note that in the above, we abuse notation and use U and Z_{i+1} to denote $U|_{S_{i+1}}$ and $Z_{i+1}|_{S_{i+1}}$, respectively.

Next we want to show that the difference between the two expectations in Equation (4) is at most $3\varepsilon_0 = 3\varepsilon/(10t) \leq \varepsilon/(2t)$, which would give a contradiction, by Equation (2). The intuition is the following. On the one hand, f'' is obtained by a pseudorandom restriction ρ_{i+1}, and so, with high probability, it has size at most s_0. On the other hand, Z_{i+1} is obtained using an extractor that is supposed to extract enough random bits for an s_0-independent generator.

The issue, however, is that f'' depends on X, the source sample of the extractor. Therefore, f'' may contain information about X, so that X is not truly random anymore. Nonetheless, being a formula of size at most s_0, f'' cannot contain too much information, and so cannot take too much entropy away from X. We make this argument more formal next.

Let us define the set of good functions for f'', namely

$$\mathcal{F} = \left\{g \mid L(g) \leq s_0 \quad \text{and} \quad \mathbf{Pr}_{W_i,\rho_{i+1}}[f'' = g] \geq \varepsilon_0/s_0^{cs_0}\right\},$$

where c is some constant. Let $\mathcal{E}$ denote the event $f'' \in \mathcal{F}$. We first show the following.

▷ Claim 22. It is the case that $\mathbf{Pr}[\neg\mathcal{E}] \leq 2\varepsilon_0$.

Proof of Claim 22. We have

$$\begin{aligned}\mathbf{Pr}[\neg\mathcal{E}] &= \mathbf{Pr}[(f'' \notin \mathcal{F}) \wedge (L(f'') > s_0)] + \mathbf{Pr}[(f'' \notin \mathcal{F}) \wedge (L(f'') \leq s_0)] \\ &\leq \mathbf{Pr}[(L(f'') > s_0)] + \mathbf{Pr}[(f'' \notin \mathcal{F}) \wedge (L(f'') \leq s_0)].\end{aligned}$$

Note that, by the pseudorandom shrinkage lemma (Lemma 21), we have $\mathbf{Pr}[L(f'') > s_0] \leq \varepsilon_0$. Also note that under the condition that $L(f'') \leq s_0$, there can be at most $s_0^{O(s_0)}$ choices for f'', since a formula of size s_0 can be specified using $O(s_0 \log s_0)$ bits (Proposition 8). Therefore, $\mathbf{Pr}[(f'' \notin \mathcal{F}) \wedge (L(f'') \leq s_0)] \leq s_0^{O(s_0)} \cdot \varepsilon_0/s_0^{cs_0} \leq \varepsilon_0$. ◁

Let us now analyze Equation (4) while conditioning on the event $\mathcal{E}$. We show the following.

▷ Claim 23. It is the case that $\mathbb{E}\left[f''(U) \mid \mathcal{E}\right] - \mathbb{E}\left[f''(Z_{i+1}) \mid \mathcal{E}\right] \leq \varepsilon_0$.

Proof of Claim 23. First note that conditioning on $\mathcal{E}$, X still has large min-entropy. More precisely, for every $g \in \mathcal{F}$ it is the case that $H_\infty(X \mid f'' = g) \geq \aleph/2$. This is because, for every x in the range of X, we have

$$\mathbf{Pr}[X = x \mid f'' = g] \leq \frac{\mathbf{Pr}[X = x]}{\mathbf{Pr}[f'' = g]} \leq \frac{2^{-\aleph}}{\varepsilon_0/s_0^{c\cdot s_0}} = 2^{-(\aleph - \log(1/\varepsilon_0) - c\cdot s_0 \cdot \log s_0)} \leq 2^{-\aleph/2}.$$

Then, by the definition of the extractor, we have $\mathbb{E}\left[f''(G_k(U)) \mid \mathcal{E}\right] - \mathbb{E}\left[f''(Z_{i+1}) \mid \mathcal{E}\right] \leq \varepsilon_0$. Finally, note that $\mathbb{E}\left[f''(G_k(U)) \mid \mathcal{E}\right] = \mathbb{E}\left[f''(U) \mid \mathcal{E}\right]$, since s_0-wise independent distributions fool size-s_0 formulas. ◁

Combining Claim 22, Claim 23, and Lemma 7, we get that the quantity in Equation (4) is at most $3\varepsilon_0$, which leads to a contradiction. This completes the proof of correctness.

Locality. To see that the j-th bit of the PRG can be computed using a circuit of size $s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)}$, we observe the following equivalent construction:

1. Compute the j-th bits of the t pseudorandom selections $(\sigma_1)_j, (\sigma_2)_j, \ldots, (\sigma_t)_j$.
2. Retrieve Y_q, where q is the smallest integer such that $(\sigma_q)_j$ is a star.
3. Compute $(Z_q)_j = G_k(E(X, Y_q))_j$, as the j-th bit of the PRG.

Note that Step 1 can be done using a circuit of size $t \cdot 2^{O\left(\log^{2/3} s\right)} = s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)}$, by the pseudorandom shrinkage lemma (Lemma 21). Also, Step 2 can be done by first computing q from the sequence $((\sigma_i)_j)_{i\in[t]}$, using a circuit of size $\widetilde{O}(t)$ (Lemma 10), and then outputting Y_q from $(Y_i)_{i\in[t]}$, using a circuit of size $t \cdot \mathsf{polylog}(N)$ (Lemma 11). Finally, Step 3 can be done by a circuit of size $\widetilde{O}(\aleph)$, using the efficient extractor (Lemma 20) and the limited-independence generator (Lemma 16). ◀

5 Almost-quadratic lower bounds against arbitrary basis formulas and branching programs

The MCSP lower bounds against formulas, over an arbitrary basis, and branching programs are obtained similarly to those for de Morgan formulas in the previous section. The idea is to construct strongly local PRGs against these models by modifying the PRGs in [10].

▶ **Lemma 24.** *For any $s \geq n$, there exists a $\left(N, s, s^{1/2} \cdot 2^{O\left(\sqrt{\log s}\right)}\right)$-local PRG against size-$s$ formulas over an arbitrary basis (or branching programs).*

The MCSP lower bound in Theorem 2 follows from Lemma 24 and Theorem 13.

6 Improved AC^0 lower bounds for MCSP

In this section, we show improved lower bounds for MCSP against constant-depth circuits.

6.1 The case of depth $d > 2$

We first show the improved lower bound against circuits of depth $d > 2$ that almost matches the lower bound for PARITY.

▶ **Theorem 25** (Theorem 3 restated). *For every $d > 2$ and every constant $\gamma > 0$, any depth-d AC^0 circuit computing MCSP on truth tables of length N must have size $2^{\Omega\left(N^{1/(d+2+\gamma)}\right)}$.*

The above result is proved using the following structural property of small-depth circuits, which says that for any such circuit, there exists some locally computable restriction that simplifies the circuits to be a constant while leaving many variables unrestricted.

▶ **Lemma 26.** *For any size-s depth-d circuit C, there exists a restriction $\rho \in \{0,1,*\}^N$ such that C_ρ is a constant function, $\left|\rho^{-1}(*)\right| \geq \frac{N}{O(\log s)^{d-2}} - \log s$, and there exists a circuit of size $d \cdot \log(N) \cdot \widetilde{O}\left(\log^3 s\right)$ such that, given $j \in \{0,1\}^{\log N}$, the circuit computes the j-th coordinate of ρ.*

The proof of Lemma 26, presented in the full version, uses the pseudorandom switching lemma due to Trevisan and Xue [21], which says that a depth-2 circuit is likely to be simplified after hit by a pseudorandom restriction. We now prove Theorem 25.

Proof of Theorem 25. Let C be a depth-d AC^0 circuit on $\{0,1\}^N \times \{0,1\}^{\log N}$ such that C computes MCSP on truth tables of length N, and let s be the size of C.

For a size parameter $\lambda = d \cdot \log(N) \cdot \widetilde{O}\left(\log^3 s\right)$, let $C' = C(\cdot, \lambda)$. Let ρ be a restriction from Lemma 26 for C'. By Lemma 26, we have that C'_ρ is a constant function. First note that $C'_\rho(0^{|\rho^{-1}(*)|}) = 1$. To see this, note that $C'_\rho(0^{|\rho^{-1}(*)|}) = C(\mathtt{tt}(f), \lambda)$, where C computes MCSP and $f\colon \{0,1\}^{\log N} \to \{0,1\}$ is the following:

$$f(j) = \begin{cases} 0, & \text{if } \rho_j = 0 \text{ or } \rho_j = *, \\ 1, & \text{else if } \rho_j = 1. \end{cases}$$

By Item 3 of Lemma 26, such a function f can be computed by a λ-size circuit. On the other hand, there can be $2^{|\rho^{-1}(*)|}$ different functions corresponding to the different partial assignments to the unrestricted variables. Since there are at most $O(\lambda \log \lambda)$ different circuits of size at most λ, in order for C'_ρ to be the constant 1, we must have $2^{O(\lambda \log \lambda)} \geq 2^{|\rho^{-1}(*)|} = 2^{\frac{N}{O(\log s)^{d-2}} - \log s}$, which, by a simple calculation, implies $s = 2^{\Omega\left(N^{1/(d+2+\gamma)}\right)}$, for any $\gamma > 0$. ◀

6.2 The case of depth 2

Here we show that computing MCSP requires depth-2 circuits of almost maximum size.

▶ **Theorem 27** (Theorem 4 restated). *Any CNF or DNF computing* MCSP *on truth tables of length N must have size $2^{N/\widetilde{O}\left(\log^2 N\right)}$.*

The proof uses a variant of Lemma 26 which says that a depth-2 circuit can be made constant via a more efficient restriction. Given such a local restriction, it is straightforward to prove Theorem 27 following the argument in the proof of Theorem 25.

7 Open problems

Our de Morgan formula lower bound for MCSP is still slightly weaker than the state-of-the-art de Morgan formula lower bound due to Tal [19], which is $\Omega\left(N^3/\left(\log N \cdot (\log\log N)^2\right)\right)$. Can the MCSP lower bound be improved? Are there better constructions of local PRGs against formulas? Or, are there alternative proofs that do not rely on local PRGs?

A similar question can be asked for small-depth circuits. In particular, can we show that MCSP requires depth-2 circuits (i.e., CNFs or DNFs) of size $2^{\Omega(N)}$, as in the case of PARITY?

What are other restricted models of computation against which we can show MCSP lower bounds using local PRGs? The recent "random walk PRG" by Chattopadhyay, Hatami, Hosseini, and Lovett [3] is also local and can be used to get MCSP lower bounds. However, as a general PRG that can be used to fool a variety of restricted models, it has sub-optimal usefulness (which is determined by its seed length) compared to the best-known lower bounds for most of those models.

References

1 Miklós Ajtai and Avi Wigderson. Deterministic Simulation of Probabilistic Constant Depth Circuits. *Advances in Computing Research*, 5:199–222, 1989. doi:10.1109/SFCS.1985.19.

2 Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from Random Strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006. doi:10.1137/050628994.

3 Eshan Chattopadhyay, Pooya Hatami, Kaave Hosseini, and Shachar Lovett. Pseudorandom Generators from Polarizing Random Walks. In *CCC*, pages 1:1–1:21, 2018. URL: https://eccc.weizmann.ac.il/report/2018/015/.

4 Mary Cryan and Peter Bro Miltersen. On Pseudorandom Generators in NC^0. In *Mathematical Foundations of Computer Science 2001, 26th International Symposium, MFCS 2001 Marianske Lazne, Czech Republic, August 27-31, 2001, Proceedings*, pages 272–284, 2001. doi:10.1007/3-540-44683-4_24.

5 Irit Dinur and Or Meir. Toward the KRW Composition Conjecture: Cubic Formula Lower Bounds via Communication Complexity. *Computational Complexity*, 27(3):375–462, 2018. doi:10.1007/s00037-017-0159-x.

6 Johan Håstad. Almost Optimal Lower Bounds for Small Depth Circuits. In *STOC*, 1986. doi:10.1145/12130.12132. doi:10.1145/12130.12132.

7 Johan Håstad. The Shrinkage Exponent of de Morgan Formulas is 2. *SIAM J. Comput.*, 27(1):48–64, 1998. doi:10.1137/S0097539794261556.

8 Shuichi Hirahara. Non-Black-Box Worst-Case to Average-Case Reductions within NP. In *FOCS*, pages 247–258, 2018. URL: https://eccc.weizmann.ac.il/report/2018/138/.

9 Shuichi Hirahara and Rahul Santhanam. On the Average-Case Complexity of MCSP and Its Variants. In *CCC*, pages 7:1–7:20, 2017. doi:10.4230/LIPIcs.CCC.2017.7.

10 Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from Shrinkage. In *FOCS*, pages 111–119, 2012. URL: https://eccc.weizmann.ac.il/report/2012/057/.

11 Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *STOC*, pages 73–79, 2000. URL: https://eccc.weizmann.ac.il/report/1999/045/.

12 E.I. Nechiporuk. On a Boolean function. *Doklady Akademii Nauk SSSR*, 169(4):765–766, 1966. English translation in Soviet Mathematics Doklady.

13 Noam Nisan and David Zuckerman. Randomness is Linear in Space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996. doi:10.1006/jcss.1996.0004.

14 Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:158, 2018. URL: https://eccc.weizmann.ac.il/report/2018/158/.

15 Igor Carboni Oliveira and Rahul Santhanam. Conspiracies Between Learning Algorithms, Circuit Lower Bounds, and Pseudorandomness. In *CCC*, pages 18:1–18:49, 2017. URL: https://eccc.weizmann.ac.il/report/2016/197/.

16 Igor Carboni Oliveira and Rahul Santhanam. Hardness Magnification for Natural Problems. In *FOCS*, pages 65–76, 2018. URL: https://eccc.weizmann.ac.il/report/2018/139/.

17 Claude E. Shannon. The Synthesis of Two-terminal Switching Circuits. *Bell Systems Technical Journal*, 28:59–98, 1949. doi:10.1002/j.1538-7305.1949.tb03624.x.

18 Avishay Tal. Shrinkage of De Morgan Formulae by Spectral Techniques. In *FOCS*, pages 551–560, 2014. URL: https://eccc.weizmann.ac.il/report/2014/048/.

19 Avishay Tal. Formula lower bounds via the quantum method. In *STOC*, pages 1256–1268, 2017. doi:10.1145/3055399.3055472.

20 Boris A. Trakhtenbrot. A Survey of Russian Approaches to Perebor (Brute-Force Searches) Algorithms. *IEEE Annals of the History of Computing*, 6(4):384–400, 1984. doi:10.1109/MAHC.1984.10036.

21 Luca Trevisan and Tongke Xue. A Derandomized Switching Lemma and an Improved Derandomization of AC^0. In *CCC*, pages 242–247, 2013. URL: https://eccc.weizmann.ac.il/report/2012/116/.

22 Ingo Wegener. *The complexity of Boolean functions*. Wiley-Teubner, 1987. URL: http://ls2-www.cs.uni-dortmund.de/monographs/bluebook/.

The Norms of Graph Spanners

Eden Chlamtáč
Ben Gurion University of the Negev, Beersheva, Israel

Michael Dinitz
Johns Hopkins University, Baltimore, MD, USA

Thomas Robinson
Ben Gurion University of the Negev, Beersheva, Israel

Abstract

A t-spanner of a graph G is a subgraph H in which all distances are preserved up to a multiplicative t factor. A classical result of Althöfer et al. is that for every integer k and every graph G, there is a $(2k-1)$-spanner of G with at most $O(n^{1+1/k})$ edges. But for some settings the more interesting notion is not the number of edges, but the degrees of the nodes. This spurred interest in and study of spanners with small maximum degree. However, this is not necessarily a robust enough objective: we would like spanners that not only have small maximum degree, but also have "few" nodes of "large" degree. To interpolate between these two extremes, in this paper we initiate the study of graph spanners with respect to the ℓ_p-norm of their degree vector, thus simultaneously modeling the number of edges (the ℓ_1-norm) and the maximum degree (the ℓ_∞-norm). We give precise upper bounds for all ranges of p and stretch t: we prove that the greedy $(2k-1)$-spanner has ℓ_p norm of at most $\max(O(n), O(n^{\frac{k+p}{kp}}))$, and that this bound is tight (assuming the Erdős girth conjecture). We also study universal lower bounds, allowing us to give "generic" guarantees on the approximation ratio of the greedy algorithm which generalize and interpolate between the known approximations for the ℓ_1 and ℓ_∞ norm. Finally, we show that at least in some situations, the ℓ_p norm behaves fundamentally differently from ℓ_1 or ℓ_∞: there are regimes ($p = 2$ and stretch 3 in particular) where the greedy spanner has a provably superior approximation to the generic guarantee.

2012 ACM Subject Classification Theory of computation → Sparsification and spanners

Keywords and phrases spanners, approximations

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.40

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1903.07418`.

Funding *Eden Chlamtáč*: Supported in part by ISF grant 1002/14.
Michael Dinitz: Supported in part by NSF awards CCF-1464239 and CCF-1535887.
Thomas Robinson: Supported in part by ISF grant 1002/14.

1 Introduction

Graph spanners are subgraphs which approximately preserve distances. Slightly more formally, given a graph $G = (V, E)$ (possibly with lengths on the edges), a subgraph H of G is a *t-spanner* of G if $d_G(u, v) \leq d_H(u, v) \leq t \cdot d_G(u, v)$ for all $u, v \in V$, where d_G denotes shortest-path distances in G (and d_H in H). The value t is called the *stretch* of the spanner.

Graph spanners were originally introduced in the context of distributed computing [27, 26], but have since proved to be a fundamental building block that is useful in a variety of applications, from property testing [7] to network routing [28]. When building spanners there are many objectives which we could try to optimize, but probably the most popular is the number of edges (the *size* or the *sparsity*). Not only is sparsity important in many applications, it also admits a beautiful tradeoff with the stretch, proved by Althöfer et al. [2]:

▶ **Theorem 1** ([2]). *For every integer $k \geq 1$ and every weighted graph $G = (V, E)$ with $|V| = n$, there is a $(2k-1)$-spanner H of G with at most $O(n^{1+1/k})$ edges.*

While understanding the tradeoff between the size and the stretch was a seminal achievement, for many applications (particularly in distributed computing) we care not just about the size, but also about the *maximum degree*. Unfortunately, unlike the size, there is no possible tradeoff between the stretch and the maximum degree. This is trivial to see: if G is a star, then the only spanner of G with non-infinite stretch has maximum degree of $n-1$. In general, if G has maximum degree Δ, then all we can say is the trivial fact that G has a spanner with maximum degree at most Δ. Nevertheless, given the importance of the maximum degree objective, there has been significant work on building spanners that minimize the maximum degree from the perspective of *approximation algorithms* [23, 10, 9]. From this perspective, we are given a graph G and stretch value t and are asked to find the "best" t-spanner of G (where "best" means minimizing the maximum degree).

While this has been an interesting and productive line of research, clearly there are problems with the maximum degree objective as well. For example, if it is unavoidable for there to be some node of large degree d, the maximum degree objective allows us to make every other vertex also of degree d, with no change in the objective function. But clearly we would prefer to have fewer high-degree nodes if possible!

So we are left with a natural question: can we define a notion of "cost" of a spanner which discourages very high degree nodes, but if there are high degree nodes, still encourages the rest of the nodes to have small degree? There is of course an obvious candidate for such a cost function: the ℓ_p norm of the degree vector. That is, given a spanner H, we can define $\|H\|_p$ to be the ℓ_p-norm of the n-dimensional vector in which the coordinate corresponding to a node v contains the degree of v in H. Then $\|H\|_1$ is just (twice) the total number of edges, and $\|H\|_\infty$ is precisely the maximum degree. Thus the ℓ_p-norm is an interpolation between these two classical objectives. Moreover, for $1 < p < \infty$, this notion of cost has precisely the properties that we want: it encourages low-degree nodes rather than high-degree nodes, but if high-degree nodes are unavoidable it still encourages the rest of the nodes to be as low-degree as possible. These properties, of interpolating between the average and the maximum, are why the ℓ_p-norm has appeared as a popular objective for a variety of problems, ranging from clustering (the famous k-means problem [22, 24]), to scheduling [4, 3, 1], to covering [21].

1.1 Our Results and Techniques

In this paper we initiate the study of graph spanners under the ℓ_p-norm objective. We prove a variety of results, giving upper bounds, lower bounds, and approximation guarantees. Our main result is the analog of Theorem 1 for the ℓ_p-norm objective, but we also characterize universal lower bounds as part of an effort to understand the generic approximation ratio for the related optimization problem. We also show that in some ways the ℓ_p-norm can behave fundamentally differently than the traditional ℓ_1 or ℓ_∞ norms, by proving that the greedy algorithm can have an approximation ratio that is *strictly better* than the generic guarantee, unlike the ℓ_1 or ℓ_∞ settings.

1.1.1 Upper Bound

We begin by proving our main result: a universal upper bound (the analog of Theorem 1) for ℓ_p-norm spanners. Recall the classical greedy algorithm for constructing a t-spanner H of a graph $G = (V, E)$. Consider the edges in nondecreasing order of edge length, and when considering edge $\{u, v\}$, add it to H if currently $d_H(u, v) > t \cdot d_G(u, v)$. We call H the *greedy*

t-spanner of G. It is trivial to show that the greedy t-spanner has girth at least $t+2$. This is the algorithm that was used to prove Theorem 1, and it has since received extensive study (see, e.g., [20, 8]) and will form the basis of our upper bound:

▶ **Theorem 2.** *Let $k \geq 1$ be an integer, let $G = (V, E)$ be a graph (possibly with lengths on the edges), and let $H = (V, E_H)$ be the greedy $(2k-1)$-spanner of G. Then $\|H\|_p \leq \max\left(O(n), O\left(n^{\frac{k+p}{kp}}\right)\right)$ for all $p \geq 1$.*

In other words, if $p \geq \frac{k}{k-1}$ then our upper bound is $O(n)$, and otherwise it is $O\left(n^{\frac{k+p}{kp}}\right)$. Clearly this interpolates between $p = 1$ and $p = \infty$: when $p = 1$ this is the same bound as Theorem 1, while if $p = \infty$ this gives $O(n)$ which is the only possible bound in terms of n. To get some more intuition for this bound, note that $n^{\frac{k+p}{kp}}$ would be the ℓ_p-norm of H if H had the size guaranteed by Theorem 1 and was also regular. So one way to think of this bound is that while the greedy spanner can be non-regular, its ℓ_p-norm still acts as if it were regular.

It is also straightforward to prove that this bound is tight if we again assume the Erdős girth conjecture [19]; for completeness, we do this in the full version [12].

The proof of Theorem 1 from [2] is relatively simple: the greedy $(2k-1)$-spanner has girth at least $2k+1$, and any graph with more than $n^{1+1/k}$ edges must have a cycle of length at most $2k$. Generalizing this to the ℓ_p-norm is significantly more complicated, since it is not nearly as easy to show a relationship between the girth and the ℓ_p-norm. But this is precisely what we do.

It turns out to be easiest to prove Theorem 2 for stretch 3: it just takes one more step beyond [2] to split the vertices of the high-girth graph (the spanner) into "low" and "high" degrees, and show that each vertex set does not contribute too much to the ℓ_p norm. However for larger stretch values this approach does not work: the main lemma used for stretch 3 (Lemma 5) is simply false when generalized to larger stretch bounds. Instead, we need a much more involved decomposition into "low", "medium", and "high"-degree nodes. This decomposition is very subtle, since the categories are not purely about the degree, but rather about how the degree relates to expansion at some particular distances from the node. We also need to further decompose the "high"-degree nodes into sets determined by which distance level we consider the expansion of. We then separately bound the contribution to the p-norm of each class in the decomposition; for "low"-degree nodes this is quite straightforward, but for medium and high-degree nodes this requires some subtle arguments which strongly use the structure of large-girth graphs.

1.1.2 Universal Lower Bounds

To motivate our next set of results, consider the optimization problem of finding the "best" t-spanner of a given input graph. When "best" is the smallest ℓ_1-norm this is known as the Basic t-Spanner problem [16, 5, 15, 18], and when "best" is the smallest ℓ_∞-norm this is the Lowest-Degree t-Spanner problem [23, 10, 9]. It is natural to consider this problem for the ℓ_p-norm as well. It is also natural to consider how well the greedy algorithm (used to prove the upper bound of Theorem 2) performs as an approximation algorithm.

To see an obvious way of analyzing the greedy algorithm as an approximation algorithm, consider the ℓ_1-norm. Theorem 1 implies that the greedy algorithm always returns a spanner of size at most $O(n^{1+1/k})$, while clearly *every* spanner must have size at least $\Omega(n)$ (assuming that the input graph is connected). Thus we immediately get that the greedy algorithm is an $O(n^{1/k})$-approximation. By dividing a universal upper bound (an upper bound on the size of the greedy spanner that holds for every graph) by a universal lower bound (a lower bound on the size of *every* spanner in every graph), we can bound the approximation ratio in a way that is generic, i.e., that is essentially independent of the actual graph.

Now consider the ℓ_∞-norm. The generic approach seems to break down here: the universal upper bound is only $\Theta(n)$ (as shown by the star graph), while the universal lower bound is only $\Theta(1)$ (as shown by the path). So it seems like the generic guarantee is just the trivial $\Theta(n)$. But this is just because n is the wrong parameter in this setting: the correct parameterization is based on Δ, the maximum degree of G (i.e., $\Delta = \|G\|_\infty$). With respect to Δ, the greedy algorithm (or any algorithm) returns a spanner with maximum degree at most Δ, while *any* t-spanner of a graph with maximum degree Δ must have maximum degree at least $\Omega(\Delta^{1/t})$ (assuming the graph is unweighted). So there is still a "generic" guarantee which implies that the greedy algorithm is an $O(\Delta^{1-1/t}) \leq O(n^{1-1/t})$-approximation.

This suggests that for $1 < p < \infty$, we will need to parameterize by *both* the number of nodes n and the ℓ_p-norm Λ of G. We can define both universal upper bounds and universal lower bounds with respect to this dual parameterization:

$$\mathrm{UB}_t^p(n,\Lambda) = \max_{\substack{G=(V,E):|V|=n,\|G\|_p=\Lambda, \\ \text{and } G \text{ is connected}}} \min_{H:\ H \text{ is a } t\text{-spanner of } G} \|H\|_p$$

$$\mathrm{LB}_t^p(n,\Lambda) = \min_{\substack{G=(V,E):|V|=n,\|G\|_p=\Lambda, \\ \text{and } G \text{ is connected}}} \min_{H:\ H \text{ is a } t\text{-spanner of } G} \|H\|_p$$

With this notation, we can define the generic guarantee $g_t^p(n,\Lambda) = \mathrm{UB}_t^p(n,\Lambda)/\mathrm{LB}_t^p(n,\Lambda)$, and if we want a guarantee purely in terms of n we can define the generic guarantee $g_t^p(n) = \max_\Lambda g_t^p(n,\Lambda)$. Our upper bound of Theorem 2 can then be restated as the claim that $\mathrm{UB}_{2k-1}^p(n,\Lambda) \leq \min\left\{\Lambda, \max\left\{O(n), O(n^{\frac{k+p}{kp}})\right\}\right\}$ for all n,k,p,Λ. So in order to understand the generic guarantees $g_{2k-1}^p(n,\Lambda)$ or $g_{2k-1}^p(n)$, we need to understand the universal lower bound quantity $\mathrm{LB}_{2k-1}^p(n,\Lambda)$.

Surprisingly, unlike the ℓ_1 and ℓ_∞ cases, the universal lower bound for other values of p is extremely complex. Understanding its value, and understanding the structure of the extremal graphs which match the bound given by $\mathrm{LB}_t^p(n,\Lambda)$, are the most technically involved results in this paper. However, while the analysis and even the exact formulation of the lower bound is quite complex, it turns out to be easily computable from a simple linear program:

▶ **Theorem 3.** *There is an explicit linear program of size $O(t)$ which calculates $\mathrm{LB}_t^p(n,\Lambda)$ for any $t \in \mathbb{N}, p \geq 1$. The bound given by the program is tight up to a factor of $\log(n)^{O(t)}$.*

Our linear program and the proof of Theorem 3 appear in the full version [12]. In fact, our linear program not only calculates a lower bound on the ℓ_p-norm of any t-spanner, it also gives the parameters which define an extremal graph of ℓ_p-norm Λ with a t-spanner whose ℓ_p-norm matches this lower bound. While the structure of these extremal graphs is simple, the dependence of the parameters of these graphs on t and p is quite complex. Nevertheless, we give a complete explicit description of these graphs for every possible value of p, t.

Interestingly, despite the fact that $\mathrm{LB}_t^p(n,\Lambda)$ is fundamentally a question of extremal graph theory (although as discussed our motivation is the generic guarantee on approximation algorithms), our techniques are in some ways more related to approximation algorithms. We give a linear program which computes the LB function, and we reason about it by explicitly constructing dual solutions. This is, to the best of our knowledge, the first time that structural bounds on spanners (as opposed to approximation bounds) have been derived using linear programs. Moreover, the structure of the extremal graphs is fundamentally related to a quantity which we call the *p-log density* of the input graph. This is a generalization of the notion of "log-density", which was introduced as the fundamental parameter when designing approximation algorithms for the Densest k-Subgraph (DkS) problem [6], and has since proved useful in many approximation settings (see, e.g., [10, 11, 14, 13]).

1.1.3 Greedy Can Do Better Than The Generic Bound

As discussed, when $p = 1$ or $p = \infty$, the approximation ratio of the greedy algorithm can be bounded by the generic guarantee. But it turns out that the connection is actually even closer: when $p = 1$ and $p = \infty$, for every n and Λ the approximation ratio of the greedy algorithm is *equal* to the generic guarantee $g^p_{2k-1}(n, \Lambda)$. In other words, greedy is no better than generic in the traditional settings (we prove this for completeness, but it is essentially folklore). In fact, for the ℓ_1 objective, giving *any* approximation algorithm which is better than the generic guarantee $g^1_{2k-1}(n)$ is a long-standing open problem [18] which has only been accomplished for stretch 3 [5] and stretch 4 [18], while for the ℓ_∞ objective such an improvement was only shown recently [9] (and not with the greedy algorithm).

We show that, at least in some regimes of interest, ℓ_p-norm graph spanners exhibit fundamentally different behavior from ℓ_1 and ℓ_∞: the greedy algorithm has approximation ratio which is *better* than the generic guarantee, even though the universal upper bound is proved via the greedy algorithm! In particular, we consider the regime of stretch 3, $p = 2$, and $\Lambda = \Theta(n)$. This is a very natural regime, since $p = 2$ is the most obvious and widely-studied norm other than ℓ_1 and ℓ_∞, and stretch 3 is the smallest value for which nontrivial sparsification can occur.

Our theorems about UB and LB imply that $g^2_3(n) = g^2_3(n, n) = \Theta(\sqrt{n})$. But we show that in this setting (and in fact for any Λ as long as $p = 2$ and the stretch is 3) the greedy algorithm is an $O(n^{63/128})$-approximation. Thus we show that, unlike ℓ_1 and ℓ_∞, for $p = 2$ the greedy algorithm provides an approximation guarantee that is strictly better than the generic bound, both for specific values of Λ and when considering the worst case Λ.

1.2 Outline

We begin in Section 2 with some basic definitions and preliminaries. In order to illustrate the basic concepts in a simpler and more understandable setting, we then focus in Section 3 on the special case of stretch 3: we prove the stretch-3 version of Theorem 1 in Section 3.1, and then show that the greedy algorithm has approximation ratio better than the generic guarantee in Section 3.2. We then prove our upper and lower bounds in full generality: the upper bound (i.e., the proof of Theorem 2) in Section 4, and then our universal lower bound in Section 5. Due to space constraints, all missing proofs can be found in the appendices.

2 Definitions and Preliminaries

Let $G = (V, E)$ be a graph, possibly with lengths on the edges. For any vertex $u \in V$, we let $d(u)$ denote the degree of u and let $N(u)$ denote the neighbors of u. We will also generalize this notation slightly by letting $N_i(u)$ denote the set of vertices that are exactly i hops away from u (i.e., their distance from u if we ignore lengths is exactly i), and we let $d_i(u) = |N_i(u)|$. Note that by definition, $N_0(u) = \{u\}$ and $d_0(u) = 1$ for all $u \in V$. We will sometimes use $B(v, r) = \cup_{i=0}^{r} N_i(v)$ to denote the ball around v of radius r.

We let $d_G : V \times V \to \mathbb{R}_{\geq 0}$ denote the shortest-path distances in G. A subgraph $H = (V, E_H)$ of a graph $G = (V, E)$ is a *t-spanner* of G if $d_H(u, v) \leq t \cdot d_G(u, v)$ for all $u, v \in E$. Recall that $\|\vec{x}\|_p = (\sum_{i=1}^{n} x_i^p)^{1/p}$ for any $p \geq 1$ and $\vec{x} \in \mathbb{R}^n$. To measure the "cost" of a spanner, for any graph $G = (V, E)$, let $\vec{d_G}$ denote the vector of degrees in G and for any $p \geq 1$, let $\|G\|_p = \|\vec{d_G}\|_p$. For any subset $S \subseteq V$, we let $\|S\|_p$ denote the ℓ_p norm of the vector obtained from $\vec{d_G}$ by removing the coordinate of every node not in S (note that we do not remove the nodes from the graph, i.e., $\|S\|_p$ is the norm of the degrees in G of the nodes in S, not in the subgraph induced by S).

3 Warmup: Stretch 3

We begin by analyzing the special case of stretch 3, particularly for the ℓ_2-norm. More specifically, we will focus on bounding $\text{UB}_3^p(n, \Lambda)$. This is one of the simplest cases, but demonstrates (at a very high level) the outlines of our upper bound. Moreover, in this particular case we can prove that the greedy algorithm performs better than the generic guarantee, showing a fundamental difference between the ℓ_2 norm and the more traditional ℓ_1 and ℓ_∞ norms.

3.1 Upper Bound

Recall that *greedy spanner* is the spanner obtained from the obvious greedy algorithm: starting with an empty graph as the spanner, consider the edges one at a time in nondecreasing length order, and add an edge if the current spanner does not span it (within the given stretch requirement). It is obvious that when run with stretch parameter t this algorithm does indeed return a t-spanner, and moreover it will return a t-spanner that has girth at least $t+2$ (if there is a $(t+1)$-cycle then the algorithm would not have added the final edge).

Our main goal in this section will be to prove the following theorem.

▶ **Theorem 4.** *Let $G = (V, E)$ be a graph and let $H = (V, E_H)$ be the greedy 3-spanner of G. Then $\|H\|_p \leq \max(O(n), O(n^{(2+p)/(2p)}))$ for all $p \geq 1$.*

In other words, when $1 \leq p \leq 2$ the greedy 3-spanner H has $\|H\|_p \leq O(n^{(2+p)/(2p)})$, and when $p \geq 2$ we get that that $\|H\|_p \leq O(n)$.

To prove this theorem, we will first show that nodes with "large" degree cannot be incident with too many edges in any graph of girth at least 5 (like the greedy 3-spanner). This is the most important step, since for $p > 1$ the p-norm of a graph gives greater "weight" to nodes with larger degree.

▶ **Lemma 5.** *Let $G = (V, E)$ be a graph with girth at least 5. Then $\sum_{v \in V : d(v) \geq 2\sqrt{n}} d(v) \leq 2n$.*

Proof. Suppose for the sake of contradiction that these vertices have total degree greater than $2n$, and let $\{v_1, \ldots, v_{\ell+1}\}$ be a minimal set with this property. That is, all these vertices have degree at least $2\sqrt{n}$, and furthermore $\sum_{i=1}^{\ell} d(v_i) \leq 2n < \sum_{i=1}^{\ell+1} d(v)$.

Because G has girth at least 5, any two vertices v_i, v_j in this set have at most one common neighbor. That is, $|N(v_i) \cap N(v_j)| \leq 1$. Thus, for every $j \in [\ell+1]$, the number of "new" neighbors contributed by $N(v_j)$ is $\left|N(v_j) \setminus \left(\bigcup_{i=1}^{j-1} N(V_i)\right)\right| \geq |N(v_j)| - \sum_{i=1}^{j-1} |N(v_i) \cap N(v_j)| \geq d(v_j) - (j-1) \geq d(v_j) - \ell$.

On the other hand, we have $2n \geq \sum_{i=1}^{\ell} d(v_i) \geq \ell \cdot 2\sqrt{n}$, and so we have $\ell \leq \sqrt{n}$. Thus, every v_j contributes at least $d(v_j) - \ell \geq d(v_j) - \sqrt{n} \geq d(v_j)/2$ new neighbors, and so we get $n \geq \left|\bigcup_{j=1}^{\ell+1} N(v_j)\right| = \sum_{j=1}^{\ell+1} \left|N(v_j) \setminus \left(\bigcup_{i=1}^{j-1} N(v_i)\right)\right| \geq \sum_{j=1}^{\ell+1} d(v_j)/2$, which contradicts our assumption that $\sum_{j=1}^{\ell+1} d(v_j) > 2n$. ◀

We can now prove Theorem 4.

Proof of Theorem 4. Let $V_{low} = \{v \in V : d(v) \leq 2\sqrt{n}\}$, and let $V_{high} = \{v \in V : d(v) > 2\sqrt{n}\}$. Since H has girth at least 5, we can apply Lemma 5. So using this lemma and

standard algebraic inequalities, we get that

$$\|H\|_p = \left(\sum_{v\in V_{low}} d(v)^p + \sum_{v\in V_{high}} d(v)^p\right)^{\frac{1}{p}} \leq \left(\sum_{v\in V_{low}} d(v)^p\right)^{\frac{1}{p}} + \left(\sum_{v\in V_{high}} d(v)^p\right)^{\frac{1}{p}}$$
$$\leq \left(\sum_{v\in V_{low}} (2\sqrt{n})^p\right)^{\frac{1}{p}} + \sum_{v\in V_{high}} d(v) \leq \left(n\cdot 2^p n^{p/2}\right)^{\frac{1}{p}} + \sum_{v\in V_{high}} d(v) \leq 2n^{\frac{2+p}{2p}} + 2n,$$

which implies the theorem. ◀

It is easy to show that the above bound is tight: for every $p \geq 1$ there are graphs in which every 3-spanner has size at least $\max(\Omega(n), \Omega(n^{\frac{2+p}{2p}}))$. In fact, we can generalize slightly to also account for different values of Λ. Theorem 2 can be interpreted as claiming that $\mathrm{UB}_3^p(n,\Lambda) \leq O(\min(\max(n, n^{\frac{2+p}{2p}}), \Lambda))$. In the full version [12] we show that this is tight: $\mathrm{UB}_3^p(n,\Lambda) \geq \Omega(\min(\max(n, n^{\frac{2+p}{2p}}), \Lambda))$ for all $p \geq 1$ and $\Omega(n^{1/p}) \leq \Lambda \leq O(n^{\frac{1+p}{p}})$.

3.2 Greedy vs Generic

It is not hard to show that in the traditional settings in which spanners have been studied, the ℓ_1 and ℓ_∞ norms, the greedy algorithm does no better than the generic guarantee, for all relevant parameter regimes. In slightly more detail, for ℓ_∞ it is relatively easy to show that $\mathrm{UB}_t^\infty(n,\Lambda) = \Theta(\Lambda)$, while $\mathrm{LB}_t^\infty(n,\Lambda) = \Theta(\Lambda^{1/t})$. Thus the generic guarantee $g_t^\infty(n,\Lambda) = \Theta(\Lambda^{1-\frac{1}{t}})$, and moreover we can build graphs in which the approximation ratio of the greedy algorithm is also $\Theta(\Lambda^{1-\frac{1}{t}})$. Similarly, for the ℓ_1-norm, classical results on spanners imply that $\mathrm{UB}_{2k-1}^1(n,\Lambda) = \Theta(\min(n^{1+\frac{1}{k}}, \Lambda))$ and $\mathrm{LB}_{2k-1}^1(n,\Lambda) = \Theta(n)$, so the generic guarantee is $g_{2k-1}^1(n,\Lambda) = \Theta(\min(n^{1+\frac{1}{k}}, \Lambda)/n)$ and there are graphs for all parameter regimes where this is the approximation ratio achieved by greedy.

We show that the behavior of the greedy spanner in intermediate ℓ_p-norms is fundamentally different: in some parameter regimes of interest, greedy *outperforms* the generic guarantee!

To demonstrate this, consider the regime of stretch 3 with the ℓ_2 norm and with $\Lambda = n$ in unweighted graphs. In this regime, the results of Section 3.1 imply that $\mathrm{UB}_3^2(n,n) = \Theta(n)$. On the other hand, our results on the universal lower bound from Section 5 (Corollary 19 in particular) directly imply that $\mathrm{LB}_3^2(n,n) = \tilde{\Theta}(\sqrt{n})$. Thus the generic guarantee is $g_3^2(n,n) = \tilde{\Theta}(\sqrt{n})$, and this is the worst case over Λ and thus $g_3^2(n) = \tilde{\Theta}(\sqrt{n})$. However, we show that the greedy algorithm is a strictly better approximation, even without parameterizing by Λ.

▶ **Theorem 6.** *The greedy algorithm is an $O(n^{63/128})$-approximation for the problem of computing the 3-spanner with smallest ℓ_2-norm (where the input is an unweighted graph).*

To prove this, let $G = (V, E)$ be a graph with $|V| = n$, let H be the greedy 3-spanner of G, and let H^* be a 3-spanner of G with minimum ℓ_2-norm. Let $\alpha = \log_n \|H^*\|_2$, so $\|H^*\|_2 = n^\alpha$; note that $\alpha \geq 1/2$. We first prove a lemma which uses $\|H^*\|_2$ to bound neighborhoods.

▶ **Lemma 7.** *$|B_{H^*}(v,r)| \leq n^{2\alpha(1-2^{-r})}$ for all $v \in V$ and $r \in \mathbb{N}$.*

Proof. We use induction on r. For the base case $r = 1$, since $\|H^*\|_2 = n^\alpha$ we know that v has degree less than n^α, and thus $|B_{H^*}(v,1)| \leq n^\alpha = n^{2\alpha(1-2^{-r})}$.

Now suppose that the theorem is true for some integer $r-1$. Let $|B_{H^*}(v,r-1)| = n^\gamma \leq n^{2\alpha(1-2^{-(r-1)})}$ (by induction). Since $\|H^*\|_2 = n^\alpha$, the average degree (in H^*) of the nodes in $B_{H^*}(v,r-1)$ is at most $n^{\alpha-(\gamma/2)}$. Thus we get that $|B_{H^*}(v,r)| \leq n^\gamma \cdot n^{\alpha-(\gamma/2)} = n^{\alpha+(\gamma/2)} \leq n^{\alpha+\alpha(1-2^{-(r-1)})} = n^{2\alpha(1-2^{-r})}$, as claimed. ◀

Using this lemma, we can now prove Theorem 6.

Proof of Theorem 6. Lemma 7 implies that $|B_{H^*}(v,6)| \leq n^{(63/32)\alpha}$ for all $v \in V$. Since H^* is a 3-spanner of G, every vertex in $B_G(v,2)$ must be in $B_{H^*}(v,6)$, and thus $|B_G(v,2)| \leq n^{(63/32)\alpha}$. Now we can use this to bound the number of 2-paths in H. Let $P_2(H)$ denote the number of paths of length 2 in H. Since H is the greedy 3-spanner of G it must have girth at least 5. This means that every path of length 2 in H which starts from v must have a different other endpoint: there cannot be two different paths of the form $v-w-u$ and $v-x-u$ in H, or else H would have girth at most 4. Thus the number of 2-paths in H which start from v is bounded by $|B_H(v,2)| \leq |B_G(v,2)| \leq n^{(63/32)\alpha}$, and thus $P_2(H) \leq n^{1+(63/32)\alpha}$.

On the other hand, note that instead of counting 2-paths in H by their starting vertex, we could instead count them by their middle vertex. The number of 2-paths where v is the *middle* node is $\binom{d_H(v)}{2} \geq d_H(v)^2/4$, and thus $P_2(H) \geq \sum_{v \in V}(d_H(v)^2/4) = \|H\|_2^2/4$. Combining these two inequalities implies that $\|H\|_2 \leq 4n^{\frac{1}{2}+\frac{63}{64}\alpha}$, and hence the greedy spanner has approximation ratio of at most $\frac{\|H\|_2}{\|H^*\|_2} \leq \frac{4n^{\frac{1}{2}+\frac{63}{64}\alpha}}{n^\alpha} = 4n^{\frac{1}{2}-\frac{1}{64}\alpha} \leq 4n^{\frac{1}{2}-\frac{1}{128}} = 4n^{63/128}$. ◀

4 Upper Bound: General Stretch

We now want to generalize the bounds from Section 3 to hold for larger stretch ($2k-1$ in particular) in order to prove Theorem 2. A natural approach would be an extension of the stretch 3 analysis: if in Lemma 5 we replaced the bound of $2\sqrt{n}$ with $2n^{1/k}$, then the proof of Theorem 4 could easily be extended to prove Theorem 2. Unfortunately this is impossible: there are graphs of girth at least $2k+1$ where it is not true that the number of edges incident with nodes of degree at least $2n^{1/k}$ is at most $O(n)$. This can be seen from, e.g., [25] for $k=3$.

So we cannot just break the vertices into "high-degree" and "low-degree" as we did for stretch 3. Instead, our decomposition is more complicated. We will still have low-degree nodes, which can be analyzed trivially. But our definition of "high" will actually be parameterized by a distance j, and we will define a node to be "high-degree" at distance j if its degree is large relative to the expansion of its neighborhood at approximately distance j. We will also introduce a new type of "medium-degree" node. In Section 4.1 we define this decomposition and prove that it is a full decomposition of V, and then in Sections 4.2 and 4.3 we show that no part in this decomposition can contribute too much to the overall cost.

First, though, we make one simple observation that will allow us to simplify notation by only considering one particular value of p. While we could analyze general values of p as we did for stretch 3 in Section 3.1, it is actually sufficient to prove the bound for the special case of k and p where the two terms in the maximum are equal, i.e., when $\frac{k+p}{kp} = 1$.

▶ **Lemma 8.** *Let $k \geq 1$ be an integer, let $G=(V,E)$ be a graph, and let $H=(V,E_H)$ be the greedy $(2k-1)$-spanner of G. If $\|H\|_{p'} = O(n)$ for $p' = k/(k-1)$ then $\|H\|_p \leq \max\left(O(n), O\left(n^{\frac{k+p}{kp}}\right)\right)$ for all $p \geq 1$.*

Proof. First note that $p' = k/(k-1)$ if and only if $\frac{k+p'}{kp'} = 1$. So we break into two cases, one for $p > p'$ and one for $1 \leq p < p'$. For the first case, where $p > p'$, the result follows simply because of the monotonicity of p-norms: $\|H\|_p \leq \|H\|_{p'} = O(n) = \max(O(n), O(n^{\frac{k+p}{kp}}))$.

For the second case, where $1 \leq p < p'$, let q be the value such that $1 \leq p \leq p'$ and $\frac{1}{p'} + \frac{1}{q} = \frac{1}{p}$. Recall that $\vec{d_H}$ is the degree vector of H. Then Hölder's inequality implies that $\|\vec{d_H}\|_p \leq \|\vec{d_H}\|_{p'}\|1\|_q = n^{\frac{1}{p}-\frac{1}{p'}}\|\vec{d_H}\|_{p'}$. Since by assumption we have $\|\vec{d_H}\|_{p'} \leq O(n)$, this implies that $\|H\|_p \leq O(n^{1+\frac{1}{p}-\frac{1}{p'}}) = O(n^{\frac{1}{p}-\frac{k-1}{k}+1}) = O(n^{\frac{k+p}{kp}})$, as claimed. ◀

4.1 Graph Decomposition

Recall that $d_i(v)$ denotes the number of vertices at distance exactly i from v. This will let us define the following vertex sets.

▶ **Definition 9.** *Let $G = (V, E)$ be a graph of girth at least $2k + 1$, with $k \geq 3$. Then define*

$$V_{low} = \{v \in V : d_1(v) \leq n^{1/k}\}$$
$$V_{med} = \{v \in V : n^{(k-2)/(k-1)} d_1(v)^{1/(k-1)} \leq d_{k-1}(v)\}$$
$$V_{high,j} = \{v \in V : d_{k-2j-1}(v) \leq n^{1/(k-1)} d_{k-2j-3}(v) d_1(v)^{(k-2)/(k-1)}\},$$

where $0 \leq j \leq \lfloor (k-3)/2 \rfloor$.

It is not hard to see that this notion of high still corresponds to a deviation from regularity, as in the stretch 3 setting; the difference is that this deviation is relative to the size of the neighborhood at distance $k - 2j - 1$ vs the neighborhood at distance $k - 2j - 3$.

As we will see in Sections 4.2 and 4.3, analyzing the contribution of $V_{high,j}$ to the p-norm of the greedy spanner is in some sense the "main" technical step: analyzing V_{low} is straightforward, and analyzing V_{med}, while nontrivial, turns out to be easier than the case for $V_{high,j}$. Before we do this, though, we will show that we have a full decomposition of V:

▶ **Theorem 10.** *Let $G = (V, E)$ be a graph of girth at least $2k + 1$, with $k \geq 3$. Then $V = V_{low} \cup V_{med} \cup \left(\cup_{0 \leq j \leq \lfloor (k-3)/2 \rfloor} V_{high,j}\right)$.*

Proof. We prove the case when k is odd. The even case is similar. Assume that $v \notin \cup_{0 \leq j \leq \lfloor (k-3)/2 \rfloor} V_{high,j}$. Then by the definition of $V_{high,j}$, we know that $d_{k-2j-1}(v) > n^{1/(k-1)} d_{k-2j-3}(v) d_1(v)^{(k-2)/(k-1)}$ for all j. Then a straightforward induction on j implies that

$$d_{k-1}(v) > n^{1/2} d_1(v)^{(k-2)/2}. \tag{1}$$

If further we assume that $v \notin V_{low}$, then $d_1(v) > n^{1/k}$, and thus

$$(d_1(v))^{k(k-3)/(2(k-1))} > n^{(k-3)/(2(k-1))}. \tag{2}$$

Finally, assuming that $v \notin V_{med}$ implies that

$$n^{(k-2)/(k-1)} (d_1(v))^{1/(k-1)} > d_{k-1}(v). \tag{3}$$

If we then multiply inequalities (1), (2) and (3), after some elementary algebra we find that $1 > 1$, a contradiction. Thus $v \in V_{low} \cup V_{med} \cup \left(\cup_{0 \leq j \leq \lfloor (k-3)/2 \rfloor} V_{high,j}\right)$. ◀

4.2 Structural Lemmas for High-Girth Graphs

With Theorem 10 in hand, it remains to bound the contribution to the p-norm of the spanner of these different vertex sets. In order to do this, we start with a few useful lemmas (proofs can be found in the full version [12]). We first give a simple lemma: if the girth is large enough, then the neighborhoods around a node can be bounded by the neighborhoods around its neighbors.

▶ **Lemma 11.** *Let $G = (V, E)$ have girth at least $2k+1$ with $k \geq 2$. Then $\sum_{w \in N_1(v)} d_{k-1}(w) \leq d_k(v) + d_1(v) d_{k-2}(v)$ for all $v \in V$.*

With this lemma in hand, we will now prove a more complicated technical lemma which will likewise hold for all high-girth graphs. For a given v, w with $v \in N(w)$, we can consider the fraction of the k-neighborhood of w which is also contained in the $(k-1)$-neighborhood of v. Then if we sum this fraction over all neighbors v of w, we would of course get 1 since the girth constraint would imply that any two neighbors of v cannot both be first hops on paths to the same node in $N_k(w)$. But what if we consider the slightly different ratio of $d_{k-1}(v)/d_k(w)$? This is notably different since it includes in the numerator not just $N_{k-1}(v) \cap N_k(w)$, but also $N_{k-1}(v) \cap N_{k-2}(w)$. It will prove useful for us to reason about these values, so we show that "on average" they behave approximately the same: if we sum over the neighbors of any given node then these fractions can add up to something quite large (not 1), but overall they only add up to $O(n)$.

▶ **Lemma 12.** *Let $k \geq 1$ be an integer, and let $G = (V, E)$ have girth at least $2k+1$ and minimum degree at least 4. Then $\sum_{w \in V} \sum_{v \in N(w)} \frac{d_{k-1}(v)}{d_k(w)} \leq 2n$.*

The proof of this lemma is quite technical, but is done with an induction on k and careful use of the arithmetic-harmonic mean inequality. While Lemma 12 is the main structural result that we will use to bound the "high" degree nodes, the following corollary makes it slightly simpler to use.

▶ **Corollary 13.** *Let $k \geq 2$ be an integer, and let $G = (V, E)$ have girth at least $2k+1$ and minimum degree at least 4. Then $\sum_{v \in V} \frac{(d_1(v))^2 d_{k-2}(v)}{d_k(v) + d_1(v) d_{k-2}(v)} \leq 2n$.*

4.3 Proving Theorem 2

We can now finally prove Theorem 2 by analyzing the contribution of the different sets in the decomposition to any graph of girth at least $2k+1$ (in particular, the greedy $(2k-1)$-spanner). All missing proofs can be found in the full version [12].

The analysis of the low nodes is straightforward, while the analysis of the medium nodes is slightly more complex. But the main difficulty is in the high nodes.

▶ **Lemma 14.** *Let $k \geq 2$ be an integer, let $p = \frac{k}{k-1}$, and let $G = (V, E)$ have girth at least $2k+1$. Then $\|V_{low}\|_p \leq n$.*

We next bound the medium nodes.

▶ **Lemma 15.** *Let $k \geq 2$ be an integer, let $p = \frac{k}{k-1}$, and let $G = (V, E)$ have girth at least $2k+1$. Then $\|V_{med}\|_p \leq n$.*

We now bound the high nodes, with one degree assumption which we will later remove.

▶ **Lemma 16.** *Let $G = (V, E)$ be a graph of girth at least $2k+1$ with $k \geq 3$. Further assume that the graph has minimum degree at least 4. Then $\|V_{high,j}\|_{k/(k-1)} = O(n)$ for all $0 \leq j \leq \lfloor (k-3)/2 \rfloor$.*

Proof. We will break the high nodes into the following two sets:

$$V'_{high,j} = \{v \in V_{high,j} : d_{k-2j-1}(v) \geq d_{k-2j-3}(v) d_1(v)\}$$
$$V''_{high,j} = \{v \in V_{high,j} : d_{k-2j-1}(v) < d_{k-2j-3}(v) d_1(v)\}.$$

Obviously $V_{high,j} = V'_{high,j} \cup V''_{high,j}$, so we can bound each of the two sets separately. For the first set, we get that

$$\|V'_{high,j}\|_{\frac{k}{k-1}} = \left(\sum_{v \in V'_{high,j}} (d_1(v))^{\frac{k}{k-1}} \right)^{\frac{k-1}{k}} \leq \left(\sum_{v \in V'_{high,j}} \frac{n^{\frac{1}{k-1}} d_1(v)^2 d_{k-2j-3}(v)}{d_{k-2j-1}(v)} \right)^{\frac{k-1}{k}}$$
$$\leq \left(2 \sum_{v \in V'_{high,j}} \frac{n^{\frac{1}{k-1}} d_1(v)^2 d_{k-2j-3}(v)}{d_{k-2j-1}(v) + d_1(v) d_{k-2j-3}(v)} \right)^{\frac{k-1}{k}} \leq 4n.$$

The first inequality is from the definition of V_{high}, the second is from the definition of V'_{high}, and the final inequality is from Corollary 13.

To analyze V''_{high}, note that $d_{k-2j-1}(v) + d_1(v) d_{k-2j-3}(v) < 2 d_1(v) d_{k-2j-3}(v)$ by definition for all $v \in V''_{high,j}$. Combining this with Corollary 13 implies that

$$\|V''_{high,j}\|_{k/(k-1)} \leq \|V''_{high,j}\|_1 = \sum_{v \in V''_{high,j}} d_1(v) = \sum_{v \in V''_{high,j}} \frac{d_1(v)^2 d_{k-2j-3}(v)}{d_1(v) d_{k-2j-3}(v)}$$
$$\leq 2 \sum_{v \in V''_{high,j}} \frac{d_1(v)^2 d_{k-2j-3}(v)}{d_{k-2j-1}(v) + d_1(v) d_{k-2j-3}(v)} \leq 4n.$$

Thus $\|V_{high,j}\|_{k/(k-1)} \leq \|V'_{high,j}\|_{k/(k-1)} + \|V''_{high,j}\|_{k/(k-1)} \leq 8n$. ◀

Putting this all together gives the following theorem.

▶ **Theorem 17.** *Let $G = (V, E)$ have girth at least $2k+1$, $k \geq 2$ and minimum degree at least 4. Then $\|G\|_p \leq O(kn)$ for $p = \frac{k}{k-1}$.*

Proof. We know from Theorem 10 that $V = V_{low} \cup V_{med} \cup \left(\cup_{0 \leq j \leq \lfloor (k-3)/2 \rfloor} V_{high,j} \right)$ for $k \geq 3$. Thus $\|G\|_p \leq \|V_{low}\|_p + \|V_{med}\|_p + \sum_{j=0}^{\lfloor (k-3)/2 \rfloor} \|V_{high,j}\|_p \leq O(kn)$, where we used Lemmas 14, 15, 16, to bound the contribution of each set. If $k = 2$ then $V_{med} = V$ and the proof is similar (alternatively see Theorem 4). ◀

We can now remove the degree assumption and the restriction to $p = \frac{k}{k-1}$, to finally prove Theorem 2.

Proof of Theorem 2. The case of $k = 1$ is trivial since every graph H has $\|H\|_p \leq O(n^{\frac{p+1}{p}})$. For $k \geq 2$, by Lemma 8, we may assume that $p = \frac{k}{k-1}$. We will use induction on the number of vertices of degree less than 4. If H has no vertices with degree less than 4, then Theorem 17 implies Theorem 2. Otherwise, let $v \in V$ be a vertex of degree at most 3, and let $G' = G - v$ be the graph obtained by removing v. Then it is easy to see that $\|\vec{d_G} - \vec{d_{G'}}\|_1 \leq 6$, since one entry in the degree vector of value at most 3 gets removed and at most three other entries get decreased by 1. Thus we can use triangle inequality and monotonicity of norms to get that $\|G\|_p - \|G'\|_p \leq \|\vec{d_G} - \vec{d_{G'}}\|_p \leq \|\vec{d_G} - \vec{d_{G'}}\|_1 \leq 6$. Hence by the induction hypothesis we get that $\|G\|_p \leq O(kn)$ as required. ◀

5 Universal Lower Bound

As stated in Theorem 3, our lower bound can be calculated by a simple linear program of size $O(t)$ (where t is the stretch). We give this linear program formally in the full version [12]. The linear program assumes that the graph has a fairly regular structure. In particular, it

assumes that the extremal t-spanner H is a layered graph with $t+1$ layers $V_0, \ldots, V_t$, such that the subgraph induced on every two subsequent layers V_i, V_{i+1} is bipartite and biregular (in each side, all vertices have the same degree), and that the original extremal graph G (the graph whose spanner H achieves the lower bound) in addition has a biregular graph between V_0 and V_t which contributes most of the p-norm of G, and is spanned by the layered graph H. Such a spanner H can be briefly described by the cardinalities of the layers V_i and the degrees of the bipartite graphs connecting every two consecutive layers.

As we show, this assumption is without loss of generality, in the sense that pruning any graph to obtain this structure can change the p-norm of the graph or its spanner by at most a polylogarithmic factor. The linear program captures the constraints that the parameters of a spanner with such a regular structure must satisfy. These constraints are also sufficient in the sense that given any solution to the linear program, we can construct a graph G and spanner H of this form with the parameters given by this LP solution.

In fact, the extremal spanners which match our lower bound have a fairly specific structure with consistent properties:

- The layers in the extremal can be partitioned into three sections: an initial section in which we have layers of decreasing size $|V_0| \geq |V_1| \ldots \geq |V_L|$, a middle section consisting of equal size layers $|V_L| = \ldots = |V_{L+C}|$, and a final section with layers of increasing size $|V_{L+C}| \leq \ldots \leq |V_{L+C+R}|$. In some cases one of the first two sections may be missing.
- The bipartite graphs between every two consecutive layers in the spanner have the same contribution to the p-norm of the spanner.
- In addition to the edges in the spanner, the original graph also contains a biclique between the outer layers V_0 and V_t, so that $\|G\|_p = \Theta(|V_0|^{1/p}|V_t|)$.

The structure of these spanners has the property that given the lengths of the three sections, we can derive the exact structure of the spanner, and hence the exact value of the lower bound. In our analysis, we focus on this specific family of graphs, and show that it suffices to describe our lower bound.

While the lower bound for $p = 1$ or $p = \infty$ is simple, it turns out that the lower bound for intermediate values of p is quite complex, and depends on the stretch t, the norm parameter p, and the p-norm of the input graph Λ in a highly non-trivial way. To identify the extremal spanners and prove their optimality, we look at the dual of our linear program, and for every graph in our family of candidate extremal spanners, examine whether there exists a dual solution which satisfies complementary slackness w.r.t. the primal LP solution corresponding to our spanner. With this approach, for every p, t, Λ, we are able to identify the exact constraints that the parameters of an optimal spanner from our family must satisfy, and give an explicit solution, which gives our lower bound.

As an example, our analysis identifies the lower bound for relatively low values of p.

▶ **Theorem 18.** *If t is even, then for all $p \in [1, \varphi]$ (where $\varphi = \frac{1+\sqrt{5}}{2}$ is the golden ratio),*

$$\mathrm{LB}_t^p(n, \Lambda) = \tilde{\Theta}\left(\max\left\{n^{1/p}, \Lambda^{\alpha}\right\}\right) \text{ for } \alpha = 1/\left((p+1)\left(1 - ((p-1)/p)^{t/2}\right)\right).$$

If t is odd, then for all $p \in [1, 2]$,

$$\mathrm{LB}_t^p(n, \Lambda) = \tilde{\Theta}\left(\max\left\{n^{1/p}, \Lambda^{\beta}\right\}\right) \text{ for } \beta = 1/\left(1 + p\left(1 - ((p-1)/p)^{(t-1)/2}\right)\right).$$

▶ **Corollary 19.** *For all $p \in [1, \varphi]$, we have $\mathrm{LB}_2^p(n, \Lambda) = \tilde{\Theta}\left(\max\left\{n^{1/p}, \Lambda^{p/(p+1)}\right\}\right)$. For all $p \in [1, 2]$, we have $\mathrm{LB}_3^p(n, \Lambda) = \tilde{\Theta}\left(\max\left\{n^{1/p}, \sqrt{\Lambda}\right\}\right)$.*

Note that the dependence on n for this range of parameters is minimal. In fact, the only dependence on n is due to the fact that any connected n-vertex graph (such as the spanner of a connected n-vertex graph) must have p-norm at least $n^{1/p}$. If we remove the condition that the graph must be connected, the lower bounds in Theorem 18 become $\tilde{\Theta}(\Lambda^{\alpha})$ and $\tilde{\Theta}(\Lambda^{\beta})$.

For higher values of p, the lower bound becomes more complex. In particular, the parameters which determine the extremal spanner depend not only on p and t, but also on the *p-log density* of the graph, which we define to be $\log_n(\Lambda) = \log_n(\|G\|_p)$. This parameter generalizes the notion of log-density, which is at the heart of several recent breakthroughs in approximation algorithms [6, 10, 11, 14, 13], in which log-density was used to mean p-log density for $p = 1$ or $p = \infty$. As in that line of work, the structure and parameters of the graphs of interest here (the extremal spanners) is a function of the p-log density of our graph which does not depend on n. The complete technical details of our lower bound appear in the full version of the paper [12].

6 Future Work

In this paper we have initiated the study of graph spanners with cost defined by the ℓ_p-norm of the degree vector, since this provides an interesting interpolation between the ℓ_1-norm (only caring about the number of edges) and the ℓ_∞-norm (only caring about the maximum degree). But we have only scratched the surface: many of the hundreds of results on graph spanners can be extended or reexamined with respect to the ℓ_p-norm. There are also some very interesting direct extensions of this paper that would be interesting to study. In particular, we showed that the approximation ratio achieved by the greedy algorithm is strictly better than the generic guarantee for the ℓ_2-norm with stretch 3, unlike the ℓ_1 and ℓ_∞ norms. This suggests further study of the greedy algorithm in general, but also suggests extending the recent line of work on approximation algorithms for graph spanners (mostly using convex relaxations and rounding) to general ℓ_p-norms. The approaches taken for the ℓ_1-norm in the past [16, 17, 5, 18] have been quite different from the approaches used for the ℓ_∞-norm [23, 10, 9]; is there a way of interpolating between them to get even better approximations for intermediate ℓ_p-norms?

References

1 Noga Alon, Yossi Azar, Gerhard J. Woeginger, and Tal Yadid. Approximation Schemes for Scheduling. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '97, 1997.

2 Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete Comput. Geom.*, 9(1):81–100, 1993. doi:10.1007/BF02189308.

3 Yossi Azar, Leah Epstein, Yossi Richter, and Gerhard J. Woeginger. All-Norm Approximation Algorithms. In Martti Penttonen and Erik Meineche Schmidt, editors, *Algorithm Theory – SWAT 2002*, 2002.

4 Nikhil Bansal and Kirk Pruhs. Server Scheduling in the L_p Norm: A Rising Tide Lifts All Boat. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 242–250, 2003.

5 Piotr Berman, Arnab Bhattacharyya, Konstantin Makarychev, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Approximation algorithms for spanner problems and Directed Steiner Forest. *Inf. Comput.*, 222:93–107, 2013. doi:10.1016/j.ic.2012.10.007.

6 Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. Detecting high log-densities: an $O(n^{1/4})$-approximation for densest k-subgraph. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 201–210. ACM, 2010. doi:10.1145/1806689.1806719.

7 Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P. Woodruff. Transitive-closure Spanners. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 932–941, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=1496770.1496871.

8 Barun Chandra, Gautam Das, Giri Narasimhan, and José Soares. New Sparseness Results on Graph Spanners. In *Proceedings of the Eighth Annual Symposium on Computational Geometry*, SCG '92, pages 192–201, New York, NY, USA, 1992. ACM. doi:10.1145/142675.142717.

9 Eden Chlamtác and Michael Dinitz. Lowest Degree k-Spanner: Approximation and Hardness. In Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*, volume 28 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 80–95, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.APPROX-RANDOM.2014.80.

10 Eden Chlamtac, Michael Dinitz, and Robert Krauthgamer. Everywhere-Sparse Spanners via Dense Subgraphs. In *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, FOCS '12, pages 758–767, Washington, DC, USA, 2012. IEEE Computer Society. doi:10.1109/FOCS.2012.61.

11 Eden Chlamtác, Michael Dinitz, and Yury Makarychev. Minimizing the Union: Tight Approximations for Small Set Bipartite Vertex Expansion. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 881–899. SIAM, 2017. doi:10.1137/1.9781611974782.56.

12 Eden Chlamtác, Michael Dinitz, and Thomas Robinson. The Norms of Graph Spanners. *CoRR*, abs/1903.07418, 2019. arXiv:1903.07418.

13 Eden Chlamtác and Pasin Manurangsi. Sherali-Adams Integrality Gaps Matching the Log-Density Threshold. In Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20-22, 2018 - Princeton, NJ, USA*, volume 116 of *LIPIcs*, pages 10:1–10:19. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.10.

14 Eden Chlamtác, Pasin Manurangsi, Dana Moshkovitz, and Aravindan Vijayaraghavan. Approximation Algorithms for Label Cover and The Log-Density Threshold. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 900–919. SIAM, 2017. doi:10.1137/1.9781611974782.57.

15 Michael Dinitz, Guy Kortsarz, and Ran Raz. Label Cover Instances with Large Girth and the Hardness of Approximating Basic k-Spanner. *ACM Trans. Algorithms*, 12(2):25:1–25:16, December 2015. doi:10.1145/2818375.

16 Michael Dinitz and Robert Krauthgamer. Directed Spanners via Flow-based Linear Programs. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC '11, pages 323–332, New York, NY, USA, 2011. ACM. doi:10.1145/1993636.1993680.

17 Michael Dinitz and Robert Krauthgamer. Fault-tolerant Spanners: Better and Simpler. In *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, PODC '11, pages 169–178, New York, NY, USA, 2011. ACM. doi:10.1145/1993806.1993830.

18 Michael Dinitz and Zeyu Zhang. Approximating Low-stretch Spanners. In *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 821–840, Philadelphia, PA, USA, 2016. Society for Industrial and Applied Mathematics. URL: `http://dl.acm.org/citation.cfm?id=2884435.2884494`.

19 Paul Erdös. *Extremal problems in graph theory*, pages 29–36. Academia Praha, Czechoslovakia, 1964.

20 Arnold Filtser and Shay Solomon. The Greedy Spanner is Existentially Optimal. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, PODC '16, pages 9–17, 2016.

21 Daniel Golovin, Anupam Gupta, Amit Kumar, and Kanat Tangwongsan. All-Norms and All-L_p-Norms Approximation Algorithms. In *FSTTCS*, volume 2 of *LIPIcs*, pages 199–210. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2008.

22 Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. *Computational Geometry*, 28(2):89–112, 2004. Special Issue on the 18th Annual Symposium on Computational Geometry - SoCG2002. `doi:10.1016/j.comgeo.2004.03.003`.

23 Guy Kortsarz and David Peleg. Generating Low-Degree 2-Spanners. *SIAM J. Comput.*, 27(5):1438–1456, 1998. `doi:10.1137/S0097539794268753`.

24 S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982. `doi:10.1109/TIT.1982.1056489`.

25 Stefan Neuwirth. The size of bipartite graphs with girth eight. *arXiv Mathematics e-prints*, page math/0102210, February 2001. `arXiv:math/0102210`.

26 David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989. `doi:10.1002/jgt.3190130114`.

27 David Peleg and Jeffrey D. Ullman. An optimal synchronizer for the hypercube. *SIAM J. Comput.*, 18:740–747, August 1989. `doi:10.1137/0218050`.

28 Mikkel Thorup and Uri Zwick. Compact Routing Schemes. In *Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '01, pages 1–10, 2001.

On the Fixed-Parameter Tractability of Capacitated Clustering

Vincent Cohen-Addad
CNRS & Sorbonne Université, Paris, France

Jason Li
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

We study the complexity of the classic capacitated k-median and k-means problems parameterized by the number of centers, k. These problems are notoriously difficult since the best known approximation bound for high dimensional Euclidean space and general metric space is $\Theta(\log k)$ and it remains a major open problem whether a constant factor exists.

We show that there exists a $(3+\epsilon)$-approximation algorithm for the capacitated k-median and a $(9+\epsilon)$-approximation algorithm for the capacitated k-means problem in general metric spaces whose running times are $f(\epsilon,k)n^{O(1)}$. For Euclidean inputs of arbitrary dimension, we give a $(1+\epsilon)$-approximation algorithm for both problems with a similar running time. This is a significant improvement over the $(7+\epsilon)$-approximation of Adamczyk et al. for k-median in general metric spaces and the $(69+\epsilon)$-approximation of Xu et al. for Euclidean k-means.

2012 ACM Subject Classification Theory of computation → Facility location and clustering; Theory of computation → Fixed parameter tractability; Mathematics of computing → Probabilistic algorithms; Mathematics of computing → Dimensionality reduction

Keywords and phrases approximation algorithms, fixed-parameter tractability, capacitated, k-median, k-means, clustering, core-sets, Euclidean

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.41

Category Track A: Algorithms, Complexity and Games

Funding *Vincent Cohen-Addad*: Ce projet a bénéficié d'une aide de l'Etat gérée par l'Agence Nationale de la Recherche au titre du Programme Appel à projets générique JCJC 2018 portant la référence suivante: ANR-18-CE40-0004-01.
Jason Li: Supported in part by NSF awards CCF-1536002, CCF-1540541, and CCF-1617790.

1 Introduction

Clustering under capacity constraints is a fundamental problem whose complexity is still poorly understood. The capacitated k-median and k-means problems have attracted a lot of attention over the recent years (*e.g.*: [4, 22, 23, 24, 13, 3, 8, 6]), but the best known approximation algorithm for capacitated k-median remains a somewhat folklore $O(\log k)$-approximation using the classic technique of embeddings the metric space into trees that follows from the work of Charikar et al [5] on the uncapacitated version, see also [1] for a complete exposition.

Arguably, the hardness of the problem comes from having both a hard constraint on the number of clusters, k, and on the number of clients that can be assigned to each cluster. Indeed, constant factor approximation algorithms are known if the capacities [22, 23] or the number of clusters can be violated by a $(1+\epsilon)$ factor [4, 13], for constant ϵ. Moreover, the capacitated facility location problem admits constant factor approximation algorithms with no capacity violation. On the other hand and perhaps surprisingly, the best known lower bound for capacitated k-median is not higher than the $1+2/e$ lower bound for the uncapacitated version of the problem.


Thus, to improve the understanding of the problem a natural direction consists in obtaining better approximation algorithms in some specific metric spaces, or through the fixed-parameter complexity of the problem. For example, a quasi-polynomial time approximation scheme (QPTAS) for capacitated k-median in Euclidean space of fixed dimension with $(1+\epsilon)$ capacity violation was known since the late 90's [2]. This has been recently improved to a PTAS for $\mathbb{R}^2$ and a QPTAS for doubling metrics without capacity violation [9]. It remains an interesting open question to obtain constant factor approximation for other metrics such as planar graphs or Euclidean space of arbitrary dimension.

For many optimization problems are at least W[1]-hard and so obtaining exact fixed-parameter tractable (FPT) algorithms is unlikely. However, FPT algorithms have recently shown that they can help break long-standing barriers in the world of approximation algorithms. FPT approximation algorithms achieving better approximation guarantees than the best known polynomial-time approximation algorithms for some classic W[1]- and W[2]-hard problems have been designed. For example, for k-cut [15], for k-vertex separator [21] or k-treewidth-deletion [16].

For the fixed-parameter tractability of the k-median and k-means problems, a natural parameter is the number of clusters k. The FPT complexity of the classic uncapacitated k-median problem, parameterized by k, has received a lot of attention over the last 15 years. From a lower bound perspective, the problem is known to be W[2]-hard in general metric spaces and assuming the exponential time hypothesis (ETH), even for points in $\mathbb{R}^4$, there is no exact algorithm running in time $n^{o(k)}$ [10]. For $\mathbb{R}^2$ there exists an exact $n^{O(\sqrt{k})}$ which is the best one can hope for assuming ETH [10], see also [26].

From an upper bound perspective, *coreset* constructions and PTAS with running time $f(k,\epsilon)n^{O(1)}$ have been known since the early 00's [12, 19, 17, 18, 14]. In the language of fixed-parameter tractability, a coreset is essentially an "approximate kernel" for the problem: given a set P of n points in a metric space, a coreset is, loosely speaking, a mapping from the points in P to a set of points Q of size $(k \log n\epsilon^{-1})^{O(1)}$ such that any clustering of Q of cost γ can be converted into a clustering of P of cost at most $\gamma \pm \epsilon\mathsf{cost}(\mathrm{OPT})$, through the inverse of the mapping (where OPT is the optimal solution for P). See Definition 9 for a more complete definition.

In Euclidean space, several coreset constructions for uncapacitated k-median are independent of the input size and of the dimension and so are truly approximate kernels. Thus approximation schemes can simply be obtained by enumerating all possible partitions of the coreset points into k parts, evaluating the cost of each of them and outputing the one of minimum cost. However, obtaining similar results in general metric spaces seems much harder and is likely impossible. In fact, obtaining an FPT approximation algorithm with approximation guarantee less than $1 + 2/e$ is impossible assuming Gap-ETH, see [11].

For the capacitated k-median and k-means problems much less is known. First, the coreset constructions or the classic FPT-approximation schemes techniques of [20, 12] do not immediately apply. Thus, very little was known until the recent result of Adamczyk et al. [1] who proposed a $(7+\epsilon)$-approximation algorithm running in time $k^{O(k)}n^{O(1)}$. More recently, a $(69+\epsilon)$-approximation algorithm for the capacitated k-means problem with similar running time has been proposed by Xu et al. [28].

1.1 Our Results

We present a coreset construction for the capacitated k-median and k-means problems, with general capacities, and in general metric spaces (Theorem 11). For an n points set, the coreset has size $\mathrm{poly}(k\epsilon^{-1}\log n)$.

From this we derive a $(3+\epsilon)$-approximation for the k-median problem and a $(9+\epsilon)$-approximation for the k-means problem in general metric spaces.

▶ **Theorem 1.** *For any $\epsilon > 0$, there exists a $(3+\epsilon)$-approximation algorithm for the capacitated k-median problem and a $(9+\epsilon)$-approximation algorithm for the capacitated k-means problem running in time $(k\epsilon^{-1}\log n)^{O(k)}n^{O(1)}$. This running time can also be bounded by $(k/\epsilon)^{O(k)}n^{O(1)}$.*

This results in a significant improvement over the recent results of Adamczyk et al. [1] for k-median and Xu et al. [28] for (Euclidean) k-means, in the same asymptotic running time.

Moreover, combining with the techniques of Kumar et al. [20], we obtain a $(1+\epsilon)$-approximation algorithm for points in $\mathbb{R}^d$, where d is arbitrary. We believe that this is an interesting result: while it seems unlikely that one can obtain an FPT-approximation better than $1+2/e$ in general metrics, it is possible to obtain an FPT-$(1+\epsilon)$-approximation in Euclidean metrics of arbitrary dimension. This works for both the *discrete* and *continuous* settings: in the former, the set of centers must be chosen from a discrete set of candidate centers in $\mathbb{R}^d$ and the capacities may not be uniform, while in the latter the centers can be placed anywhere in $\mathbb{R}^d$ and the capacities are uniform.

▶ **Theorem 2.** *For any $\epsilon > 0$, there exists a $(1+\epsilon)$-approximation algorithm for the discrete, Euclidean, capacitated k-means and k-median problems which runs in time $(k\epsilon^{-1}\log n)^{k\epsilon^{-O(1)}}$ $n^{O(1)}$. This running time can also be bounded by $(k\epsilon^{-1})^{k\epsilon^{-O(1)}}$ $n^{O(1)}$.*

▶ **Theorem 3.** *For any $\epsilon > 0$, there exists a $(1+\epsilon)$-approximation algorithm for the continuous, Euclidean, capacitated k-means and k-median problems running in time $(k\epsilon^{-1}\log n)^{k\epsilon^{-O(1)}}$ $n^{O(1)}$. This running time can also be bounded by $(k\epsilon^{-1})^{k\epsilon^{-O(1)}}n^{O(1)}$.*

These two results are a major improvement over the 69-approximation algorithm of Xu et al. [28].

1.2 Preliminaries

We now provide a more formal definition of the problems.

▶ **Definition 4.** *Given a set of points V in a metric space with distance function d, together with a set of* clients *$C \subseteq V$, a set of* centers *$\mathbb{F} \subseteq V$ with a* capacity *$\eta_f \in \mathbb{Z}_+$ for each $f \in \mathbb{F}$, and an integer k, the* capacitated k-median problem *asks for a set $F \subseteq \mathbb{F}$ of k* centers *and an assignment $\mu : C \mapsto F$ such that $\forall f \in F$, $|\{c \mid \mu(c) = f\}| \leq \eta_f$ and that minimizes $\sum_{c\in C} d(c, \mu(c))$. We abbreviate the capacitated k-median instance as $((V,d), C, \mathbb{F}, k)$.*

▶ **Definition 5.** *The* capacitated k-means problem *is identical, except we seek to minimize $\sum_{c\in C} d(c,\mu(c))^2$.*

In the literature, centers are sometimes called *facilities*, but we will use *centers* throughout for consistency.

In the case of the capacitated Euclidean k-median and k-means, our approach works for the two main definitions. First, the definition of [28, 20]: $P = \mathbb{R}^d$ and capacities are uniform, namely $\eta_f = \eta_{f'}$, $\forall f, f' \in \mathbb{R}^d$. Second, P is some specific set of points in $\mathbb{R}^d$, and for each $f \in P$, the input specifies a specific capacity η_f

▶ **Definition 6.** *Given a capacitated k-median instance $((V,d), C, \mathbb{F}, k)$ and a set of chosen centers $F \subseteq \mathbb{F}$, define* $\mathsf{CapKMed}(C, F)$ *as the cost of the optimal assignment of the clients to the chosen centers. If it is impossible, i.e., the sum of the capacities of the centers is less than $|C|$, then* $\mathsf{CapKMed}(C, F) = \infty$.

In our analysis, we will also encounter formulations where the clients have positive *real* weights. In this case, we define a *fractional* variant of capacitated k-median, where the assignment μ is allowed to be fractional.

▶ **Definition 7.** *Suppose the clients also have weights, so we are given clients C and a weight function $w : C \to \mathbb{R}_+$. Let $W \subseteq C \times \mathbb{R}_+$ be the set of pairs $\{(c, w(c)) : c \in C\}$. Then,* $\mathsf{FracCapKMed}(W, F)$ *is the minimum value of $\sum_{c \in C, f \in F} \mu(c, f)\, d(c, f)$ over all "fractional assignments" $\mu : C \times F \to \mathbb{R}_+$ such that:*

1. *$\forall c \in C$, $\sum_{f \in F} \mu(c, f) = w(c)$, i.e., μ is a proper assignment of clients, and*
2. *$\forall f \in F$, $\sum_{c \in C} \mu(c, f) \leq \eta_f$, i.e., μ satisfies capacity constraints at all centers.*

▶ **Definition 8.** *We define* $\mathsf{CapKMeans}(C, F)$ *and* $\mathsf{FracCapKMeans}(W, F)$ *similarly, except our objective functions are $\sum_{c \in C} d(c, \mu(c))^2$ and $\sum_{c \in C, f \in F} \mu(c, f)\, d(c, f)^2$, respectively.*

It is well-known that, given a set $F \subseteq \mathbb{F}$ of centers, the problem of finding the optimum μ is an (integral) *minimum-cost flow* problem, which can be solved in polynomial time. Therefore, we assume that every time we have a set $F \subseteq \mathbb{F}$, we can evaluate $\mathsf{CapKMed}(C, F)$ and $\mathsf{CapKMeans}(C, F)$ in polynomial time. Similarly, FracCapKMed and FracCapKMeans can be solved through fractional min-cost flow, or even an LP, in polynomial time. Furthermore, if W is exactly the set C of clients with weight 1, i.e., $W = \{(c, 1) : c \in C\}$, then $\mathsf{CapKMed}(C, F) = \mathsf{FracCapKMed}(W, F)$, since the min-cost flow formulation of FracCapKMed has integral capacities and therefore integral flows as well.

We now formally state our definition of coresets, sometimes called *strong* coresets in the literature.

▶ **Definition 9.** *A (strong) coreset for a capacitated k-median instance $((V, d), C, \mathbb{F}, k)$ is a set of weighted clients $W \subseteq C \times \mathbb{R}_+$ such that for every set of centers $F \subseteq \mathbb{F}$ of size k,*

$$\mathsf{FracCapKMed}(W, F) \in (1 - \epsilon, 1 + \epsilon) \cdot \mathsf{CapKMed}(C, F).$$

The definition is identical for capacitated k-means, except CapKMed *and* FracCapKMed *are replaced by* CapKMeans *and* FracCapKMeans *above.*

▶ **Fact 10.** *Let W be a coreset for a capacitated k-median instance $((V, d), C, \mathbb{F}, k)$. We have*

$$\min_{\substack{F \subseteq \mathbb{F} \\ |F| = k}} \mathsf{FracCapKMed}(W, F) \in (1 - \epsilon, 1 + \epsilon) \cdot \min_{\substack{F \subseteq \mathbb{F} \\ |F| = k}} \mathsf{CapKMed}(C, F),$$

In particular, an α-approximation of $\min_{F \subseteq \mathbb{F}, |F| = k} \mathsf{FracCapKMed}(W, F)$ implies a $(1 + O(\epsilon))\alpha$-approximation to the capacitated k-median instance. The same holds in the capacitated k-means case, with FracCapKMed *and* CapKMed *replaced by* FracCapKMeans *and* CapKMeans*, respectively.*

For a capacitated k-median or k-means instance $((V, d), C, \mathbb{F}, k)$, the *aspect ratio* is the ratio of the maximum and minimum distances between any two points in $C \cup F$. It is well-known that we may assume, with a multiplicative error of $(1 + o(1))$ in the optimal solution, that the instance has $\mathrm{poly}(n)$ aspect ratio.[1] Therefore, we will make this assumption throughout the paper.

[1] For example, the following modification to the distances d does the trick. First, compute an $O(\log k)$-approximation [5] to the problem, and let that value be M. For any two points $u, v \in C \cup F$ with $d(u, v) > Mn^{10}$, truncate their distance to exactly Mn^{10}. Then, add Mn^{-10} distance to each pair of points $u, v \in C \cup F$. The aspect ratio is now bounded by $O(n^{20})$.

Lastly, we define $\mathbb{R}_+$ and $\mathbb{Z}_+$ as the set of positive reals and positive integers, respectively. As usual, we define *with high probability (w.h.p.)* as with probability $1-n^{-Z}$ for an arbitrarily large positive constant Z, fixed beforehand.

2 Coreset for k-median

In this section, we prove our main technical result for the k-median case: constructing a coreset for capacitated k-median of size $\text{poly}(k \log n\, \epsilon^{-1})$.

▶ **Theorem 11.** *For any small enough constant $\epsilon \geq 0$, there exists a Monte Carlo algorithm that, given an instance $((V,d), C, \mathbb{F}, k)$ of capacitated k-median, outputs a (strong) coreset $W \subseteq C$ with size $O(k^2 \log^2 n/\epsilon^3)$ in polynomial time, w.h.p.*

▶ **Theorem 12.** *For any small enough constant $\epsilon \geq 0$, there exists a Monte Carlo algorithm that, given an instance $((V,d), C, \mathbb{F}, k)$ of capacitated k-means, outputs a (strong) coreset $W \subseteq C$ with size $O(k^5 \log^5 n/\epsilon^3)$ in polynomial time, w.h.p.*

Our inspiration for the coreset construction is Chen's algorithm [7] based on random sampling. Our algorithm is essentially the same, with slightly worse bounds in the sampling step, although our analysis is a lot more involved. We describe the full algorithm in pseudocode below (see Algorithm 1).

At a high level, the algorithm first partitions the client set C into $\text{poly}(k, \log n)$ many subsets, called *rings*, with the help of a polynomial-time approximate solution (see line 1). The sets are called rings because they are of the form $C_i \cap (\mathsf{ball}(f_i', R) \setminus \mathsf{ball}(f_i', R/2))$ for some subset of clients $C_i \subseteq C$, some facility $f_i' \in \mathbb{F}$, and some positive number R (see line 7). Then, for each ring $C_{i,R}$, if $|C_{i,R}|$ is small enough, the algorithm adds the entire ring into the coreset (each with weight 1); otherwise, the algorithm takes a random sample of $r = \text{poly}(k, \log n)$ many clients in $C_{i,R}$, weights each sampled client by $|C_{i,R}|/r$, and adds the weighted sample to the coreset. The weighting ensures that the total weight of the sampled points is always equal to $|C_{i,R}|$. To prove that the algorithm produces a coreset w.h.p., Chen union bounds over all $\binom{|\mathbb{F}|}{k}$ choices of a set of k facilities, and shows that for each choice $F \subseteq \mathbb{F}$, with probability at least $1 - n^{-\Omega(k)}$, the total cost to assign the coreset points to F is approximately the total cost to assign the original clients C to F; this statement is proved through standard concentration bounds. More details and intuition for the algorithm can be found in Section 3 of Chen's paper [7].

2.1 Single ring case

We first restrict ourselves to sampling from a *single* ring $C_{i,R} \subseteq C$. That is, while we still consider the cost of serving the clients outside of $C_{i,R}$, we only perform the sampling (lines 12–13) on one ring $C_{i,R}$. The general case of $O(k \log n)$ many rings is more complicated than simply treating each ring separately. Due to space constraints, we only consider the single ring case in this extended abstract, and the rest is deferred to the full version.

Fix an arbitrary ring $C_{i,R}$ throughout this section, and define $C' := C_{i,R}$ for convenience. Let $N := |C'|$ be the number of clients, and let $f' := f_i'$ be the ring center of C' (line 4). Let W' be the (weighted) centers in $C_{i,R}$ sampled by the algorithm (lines 12–13), together with the (unweighted) centers in $C \setminus C'$, which have weight 1. Our goal is to show that $\mathsf{FracCapKMed}(W', F)$, the cost after sampling only from C', is close to the original cost $\mathsf{CapKMed}(C, F)$.

Algorithm 1 CoreSet(I).

1: $F' = \{f'_1, \ldots, f'_{O(k)}\} \leftarrow$ an $(O(1), O(1))$ bicriteria solution to instance I, namely a capacitated $O(k)$-median solution with total cost $ALG' \leq O(OPT)$ ▷ using, e.g., [23]
2: $W \leftarrow \emptyset$ ▷ $W \subseteq C \times \mathbb{R}_+$ is the final coreset at the end of the algorithm
3: Define $d_{\min}$ and $d_{\max}$ as the minimum and maximum distances, respectively, between any two points in $C \cup \mathbb{F}$ ▷ $d_{\max}/d_{\min}$ is the aspect ratio
4: **for** each center f'_i **do** ▷ $O(k)$ centers
5: $C_i \leftarrow$ the clients in C assigned to center f'_i
6: **for** each R, a power of 2 in the range $[d_{\min}, 2d_{\max}]$ **do** ▷ $O(\log n)$ iterations, assuming poly(n) aspect ratio
7: $C_{i,R} \leftarrow C_i \cap (\mathsf{ball}(f'_i, R) \setminus \mathsf{ball}(f'_i, R/2))$ ▷ We call the sets $C_{i,R}$ *rings*, with *ring center* f'_i. The rings $C_{i,R}$ over all i, R partition the client set C.
8: $r \leftarrow \gamma k \log n/\epsilon^3$ for sufficiently large (absolute) constant γ
9: **if** $|C_{i,R}| \leq r$ **then**
10: add $(c, 1)$ to W for each $c \in C_{i,R}$ ▷ $C_{i,R}$ small enough: add everything into coreset
11: **else**
12: sample r random centers in $C_{i,R}$ (without replacement)
13: add $(c, \frac{|C_{i,R}|}{r})$ to W for each sampled center c ▷ weighted so that total weight is still $|C_{i,R}|$

▶ **Lemma 13.** *W.h.p., for any set of k centers $F \subseteq \mathbb{F}$ satisfying* $\mathsf{CapKMed}(C, F) < \infty$,

$$|\mathsf{FracCapKMed}(W', F) - \mathsf{CapKMed}(C, F)| \leq \epsilon NR. \tag{1}$$

It is clear that the output W has size $O(k^2 \log^2 n/\epsilon^3)$. The rest of this section focuses on proving that W is indeed a coreset, w.h.p.

The intuition behind the ϵNR additive error is that we can "charge" this error to the cost of the bicriteria solution (line 1) that C' is responsible for. In particular, the total cost of assigning clients in C' to ring center f' in the bicriteria solution is at least $N \cdot R/2$, since all clients in C' are distance at least $R/2$ to f'. Therefore, we charge an additive error of ϵNR to a $NR/2$ portion of ALG', which is a "rate" of 2ϵ to 1. If we can do the same for all rings, then since the portions of ALG' sum to ALG', our total additive error is at most $2\epsilon \cdot ALG' = O(\epsilon) \cdot OPT$. Finally, replacing ϵ with a small enough $\Theta(\epsilon)$ gives the desired additive error of $\epsilon \cdot OPT$; note that this is where we use that the approximation ratio of ALG' is $O(1)$, and that the specific approximation ratio is not important (as long as it is constant). The formalization of this intuition is deferred to the full version; the argument is identical to Chen's [7], so we claim no novelty here.

We now prove Lemma 13. First of all, if $N = |C'| \leq r$ (line 9), then sampling changes nothing, and $\mathsf{FracCapKMed}(W', F) = \mathsf{CapKMed}(C, F)$. Therefore, for the rest of the proof, we assume that $N > r = \gamma k \log n/\epsilon^3$, with the γ taken to be a large enough constant.

Our high-level strategy is the same as Chen's: we union bound over all sets of centers $F \subseteq \mathbb{F}$ of size k, and prove that for a fixed set F, the probability of violating (1) is at most $n^{-(k+10)}$.[2] Union bounding over all $\leq \binom{n}{k}$ choices of F gives probability $\leq n^{-10}$ of

[2] For simplicity of presentation, we will focus on a success probability of $1 - n^{-10}$. The constants can be easily tweaked so that the algorithm succeeds w.h.p., i.e., with probaility $1 - n^{-Z}$ for any positive constant Z.

violating (1), proving the lemma. Therefore, from now on, we focus on a single, arbitrary set $F \subseteq \mathbb{F}$ of size k satisfying $\mathsf{CapKMed}(C, F) < \infty$, and aim to show that (1) fails with probability $\leq n^{-(k+10)}$.

For our analysis, we define a function $g : \mathbb{R}_+^{C'} \to \mathbb{R}_+$ as follows. For an input vector $\mathbf{d} \in \mathbb{R}_+^{C'}$ (indexed by clients in C'), consider a min-cost flow instance $\mathsf{FlowInstance}(\mathbf{d})$ on the graph metric with the following demands: set demand d_c at each client $c \in C'$, demand 1 at each client $c \in C \setminus C'$, and demand $N - \sum_{c \in C'} d_c$ (this demand can be negative) at ring center $f' = f'_i$ (so we are effectively treating f' as a special client with possibly negative demand, not a facility). Observe that $\mathsf{FlowInstance}(\mathbf{d})$ is a feasible min-cost flow instance, because the sum of demands is exactly

$$\sum_{c \in C'} d_c + |C \setminus C'| + \left(N - \sum_{c \in C'} d_c \right) = |C \setminus C'| + N = |C|,$$

which is the same as the sum of demands in the instance $\mathsf{CapKMed}(C, F)$, which is feasible by assumption.

Given this setup for an input vector $\mathbf{d} \in \mathbb{R}_+^{C'}$, we define the function $g(\mathbf{d})$ as the min-cost flow of $\mathsf{FlowInstance}(\mathbf{d})$. Observe that $g(\mathbb{1})$ is exactly $\mathsf{CapKMed}(C, F)$.

Now define a random vector $X \in \mathbb{R}_+^{C'}$ as follows. Each coordinate of X is independently N/r with probability r/N and 0 otherwise, so that $\mathbb{E}[X] = \mathbb{1}$. Note that X does not accurately represent our sampling of r clients, since this process is not guaranteed to sample exactly r clients. Nevertheless, it is intuitively clear that with probability $\Omega(1/n)$, X will indeed have exactly r nonzero entries, since r is the expected number; we prove this formally in the following simple claim (with $p = r/N$), whose routine proof is deferred to the full version. And if we *condition* on this event, then $g(X)$ and $\mathsf{CapKMed}(C, F)$ are now identically distributed.

▷ Claim 14. Let N be a positive integer, and let $p \in (0, 1)$ such that pN is an integer. The probability that $\mathsf{Binomial}(N, p) = pN$ is at least $\Omega(1/\sqrt{N})$.

In light of all this, our main argument has two steps. First, we show that $g(X)$ is concentrated around $\mathbb{E}[g(X)]$ using martingales. However, what we really need is concentration around $g(\mathbb{E}[X]) = g(\mathbb{1}) = \mathsf{CapKMed}(C, F)$, so our second step is to show that $\mathbb{E}[g(X)] \approx g(\mathbb{E}[X])$ (with probability 1). We formally state the lemmas below which, as discussed, together imply Lemma 13.

▶ **Lemma 15.** *Assume that $|C'| > \Theta(k \log n/\epsilon^3)$. With probability $\geq 1 - n^{-(k+20)}$, we have $|g(X) - \mathbb{E}[g(X)]| \leq \epsilon NR/2$.*

▶ **Lemma 16.** *Assume that $|C'| > \Theta(k \log n/\epsilon^3)$. Then, $|\mathbb{E}[g(X)] - g(\mathbb{E}[X])| \leq \epsilon NR/2$.*

2.1.1 Proof of Lemma 15: concentration around $\mathbb{E}[g(X)]$ via martingales.

To show that $g(X)$ is concentrated around its mean, we show that g is sufficiently Lipschitz (w.r.t. the ℓ_1 distance in $\mathbb{R}_+^{C'}$), and then apply standard martingale tools.

▷ Claim 17. The function g is R-Lipschitz w.r.t. the ℓ_1 distance in $\mathbb{R}_+^{C'}$.

Proof. Fix a client $c \in C'$, and consider two vectors $\mathbf{d}, \mathbf{d}' \in \mathbb{R}_+^{C'}$ with $\mathbf{d}' = \mathbf{d} + \delta \cdot \mathbb{1}_c$. By definition of $\mathsf{FlowInstance}$, the only difference between $\mathsf{FlowInstance}(\mathbf{d})$ and $\mathsf{FlowInstance}(\mathbf{d}')$ is that in $\mathsf{FlowInstance}(\mathbf{d}')$, client c has δ more demand and "special client" f' has δ less

demand. Therefore, if we begin with the min-cost flow of $\mathsf{FlowInstance}(\mathbf{d})$, and then add δ units of flow from c to f', then we now have a feasible flow for $\mathsf{FlowInstance}(\mathbf{d}')$.[3] This means that

$$g(\mathbf{d}') \leq g(\mathbf{d}) + \delta R.$$

Similarly, starting from a min-cost flow of $\mathsf{FlowInstance}(\mathbf{d}')$ and then adding δ units of flow from f' to c, we obtain a feasible flow for $\mathsf{FlowInstance}(\mathbf{d})$, so

$$g(\mathbf{d}) \leq g(\mathbf{d}') + \delta R.$$

Together, these two inequalities show that g is R-Lipschitz. ◁

We state the following Chernoff bound for Lipschitz functions, which can be proven by adapting the standard (multiplicative) Chernoff bound proof to a martingale.

▶ **Theorem 18.** *Let $x_1, \ldots, x_n$ be independent random variables taking value b with probability p and value 0 with probability $1-p$, and let $g : [0,1]^n \to \mathbb{R}$ be a L-Lipschitz function in ℓ_1 norm. Define $X := (x_1, \ldots, x_n)$ and $\mu := \mathbb{E}[g(X)]$. Then, for $0 \leq \epsilon \leq 1$:*

$$\Pr\left[\left|g(X) - \mathbb{E}[g(X)]\right| \geq \epsilon pnbL\right] \leq 2e^{-\epsilon^2 pn/3}$$

We apply Theorem 18 on the L-Lipschitz function g with the randomly sampled demands. Set $p := r/N$ as the sampling probability, so that $X \in \{0, 1/p\}^N$ is the random demand vector. Setting $n := N$, $b := 1/p$, and $L := R$, we obtain

$$\begin{aligned}
&\Pr\left[\left|g(X) - \mathbb{E}[g(X)]\right| \geq (\epsilon/2)NR\right] \\
&\quad = \Pr\left[\left|g(X) - \mathbb{E}[g(X)]\right| \geq (\epsilon/2)pnbL\right] \\
&\quad \leq 2\exp\left(\frac{-(\epsilon/2)^2 pn}{3}\right) \\
&\quad = 2\exp\left(\frac{-(\epsilon/2)^2 (r/N)N}{3}\right) = \exp\left(-\Theta(\epsilon^2 r)\right) = \exp\left(-\Omega(\epsilon^2 \cdot \frac{k\log n}{\epsilon^2})\right) \\
&\quad \leq n^{-(k+20)}
\end{aligned}$$

for sufficiently large γ in the definition of $r = \gamma k \log n/\epsilon^2$. This concludes Lemma 15.

2.1.2 Proof of Lemma 16: relating $\mathbb{E}[g(X)]$ with $g(\mathbb{E}[X])$.

We have obtained concentration about $\mathbb{E}[g(X)]$, but we really need concentration around $g(\mathbb{E}[X]) = \mathsf{CapKMed}(C', F)$. We establish this by proving Lemma 16.

We first show the easy direction, that $g(\mathbb{E}[X]) \leq \mathbb{E}[g(X)]$, which essentially follows from the convexity of min-cost flow: Suppose the outcomes of random variable X are $\mathbf{d}_1, \mathbf{d}_2, \ldots$ with respective probabilities $\mu_1, \mu_2, \ldots$, so that $\mathbb{E}[g(X)] = \sum_i \mu_i g(\mathbf{d}_i)$. Now consider the flow obtained by adding up, for each i, the min-cost flow of $\mathsf{FlowInstance}(\mathbf{d}_i)$ scaled by μ_i. This flow is a feasible flow to $\mathsf{FlowInstance}(\mathbb{E}[X])$ and has cost at most $\mathbb{E}[g(X)]$. Since the min-cost flow of $\mathsf{FlowInstance}(\mathbb{E}[X])$ can only be lower, we have $g(\mathbb{E}[X]) \leq \mathbb{E}[g(X)]$.

We now prove the other direction: $\mathbb{E}[g(X)] \leq g(\mathbb{E}[X]) + \epsilon NR/2$.

[3] We define demand so that if a vertex v has $d > 0$ demand, then d flow must exit v in a feasible flow, and if it has $d < 0$ demand, then $|d|$ flow must enter v.

▷ Claim 19. With probability 1, $g(X) \le g(\mathbb{E}[X]) + nNR$.

Proof. Since $X \in [0, N/r]^N$, and since g is R-Lipschitz, the entire range of $g(X)$ is contained in some interval of length $N \cdot N/r \cdot R \le N \cdot n \cdot R$. Since $\mathbb{E}[X] \in [0, N/r]^N$ as well, the value $g(\mathbb{E}[X])$ is also contained in that interval. The statement follows. ◁

▶ **Lemma 20.** *With probability* $\ge 1 - n^{-10}$, $g(X) \le g(\mathbb{E}[X]) + 0.49\epsilon NR$.

Due to space constraints, the proof of Lemma 20, which is long and technical, is deferred to the full version. Assuming Lemma 20, we now show how Claim 19 and Lemma 20 together imply Lemma 16: we have

$$\begin{aligned}\mathbb{E}[g(X)] &\le n^{-10} \cdot \big(g(\mathbb{E}[X]) + nNR\big) + (1 - n^{-10})\big(g(\mathbb{E}[X]) + 0.49\epsilon NR\big) \\ &= g(\mathbb{E}[X]) + \big(n^{-10} \cdot n + (1 - n^{-10}) \cdot 0.49\epsilon\big)NR \\ &\le g(\mathbb{E}[X]) + (\epsilon/2)NR,\end{aligned}$$

finishing the proof of Lemma 16.

2.2 $(3+\epsilon)$- and $(9+\epsilon)$-approximation – Proof of Theorem 1

In this section, we finish the algorithm for Theorem 1. We will focus mainly on the k-median case, since the k-means case is nearly identical.

Suppose we run the coreset for the capacitated k-median instance with parameter ϵ_0 (to be set later), obtaining a coreset $W \subseteq C \times \mathbb{R}^+$ of size $\text{poly}(k \log n\, \epsilon_0^{-1})$. We now want to compute some $F \subseteq \mathbb{F}$ of size k and an assignment μ of the clients in W to F minimizing $\sum_{(c,w)\in W} w \cdot d(c, \mu(c))$. By definition of coreset, if we compute an α-approximation to this problem, then we compute a $(1+\epsilon_0)\alpha$-approximation to the original capacitated k-median problem.

The strategy is similar to that in [11]: we guess a set of *leaders* and *distances* that match the optimal solution. More formally, let $F^* = \{f_1^*, \ldots, f_k^*\} \subseteq \mathbb{F}$ be the optimal solution with assignment μ^*. For each $f_i^* \in F^*$, let $(\mu^*)^{-1}(f_i^*)$ be the clients in the coreset assigned by μ^* to f_i^*, and let ℓ_i be the client in $(\mu^*)^{-1}(f_i^*)$ closest to f_i^*. We call ℓ_i the *leader* of the client set $(\mu^*)^{-1}(f_i^*)$. Also, let R_i be the distance $d(f_i^*, \ell_i)$, rounded down to the closest integer power of $(1+\epsilon_1)$ for some ϵ_1 we set later.

The algorithm begins with an enumeration phase. There are $|W|^k$ choices for the set $\{\ell_1, \ldots, \ell_k\}$, and $O(\epsilon_1^{-1} \log n)^k$ choices for the values $R_1, \ldots, R_k$, since we assumed that the instance has aspect ratio $\text{poly}(n)$. So by enumerating over $|W|^k O(\epsilon_1^{-1} \log n)^k = (k \log n\, \epsilon_0^{-1} \epsilon_1^{-1})^{O(k)}$ choices, we can assume that we have guessed the right values ℓ_i and R_i.

For each leader ℓ_i, define $\mathbb{F}_i$ as the centers $f \in \mathbb{F}$ satisfying $d(\ell_i, f) \in [1, 1+\epsilon_1) \cdot R_i$. Note that $f_i^* \in \mathbb{F}_i$ for each i. Next, the algorithm wants to pick the center in each $\mathbb{F}_i$ with the largest capacity. This way, even if it doesn't pick f_i^* for $\mathbb{F}_i$, it picks a center not much farther away that has at least as much capacity.

The most natural solution is to *greedily* choose the center with largest capacity in each $\mathbb{F}_i$. One immediate issue with this approach is that we might choose the same center twice, since the sets $\mathbb{F}_i$ are not necessarily disjoint. Note that this issue is not as pronounced in the uncapacitated k-median problem, since in that case, we can always imagine choosing the same center twice and then throwing out one copy, which changes nothing. In the capacitated case, choosing the same center twice effectively doubles the capacity at that center, so throwing out a copy affects the capacity at that center.

One simple fix to this issue is the simple idea of *color-coding*, common in the FPT literature: for each center $f \in \mathbb{F}$, independently assign a uniformly random label in $\{1, 2, 3, \ldots, k\}$. With probability $1/k^k$, each $f_i^* \in F^*$ is assigned label i. Moreover, repeating this routine $O(k^k \log n)$ times ensures that w.h.p., this will happen in some iteration. So with a $O(k^k \log n)$ multiplicative overhead in the running time, we may assume that each f_i^* is assigned label i.

The algorithm now chooses, from each $\mathbb{F}_i$, the center with the largest capacity among all centers with label i. Since f_i^* is an option for each $\mathbb{F}_i$, the center chosen can only have larger capacity. Let the center chosen from $\mathbb{F}_i$ be f_i. Let $F := \{f_1, \ldots, f_k\}$ be our chosen centers.

We now claim that F is a $(3+\epsilon_1)$-approximation. Recall μ^*, the optimal assignment to the centers F^*; we construct an assignment μ to F as follows: for each client c in the coreset, if μ^* assigns c to center f_i^*, then we set $\mu(c) = f_i$. Observe that if $\mu^*(c) = f_i^*$, then

$$d(c, f_i) \le d(c, f_i^*) + d(f_i^*, \ell_i) + d(\ell_i, f_i) \le d(c, f_i^*) + 2(1+\epsilon_1)R_i \le d(c, f_i^*) + 2(1+\epsilon_1) \cdot d(c, f_i^*),$$

where the first inequality follows from triangle inequality, the second follows since both f_i^* and f_i are approximately R_i away from ℓ_i, and the third follows from $d(c, f_i^*) \ge d(\ell_i, f_i^*) \ge R$ by our choice of ℓ_i. Therefore, we have $d(c, \mu(c)) = d(c, f_i) \le (3 + 2\epsilon_1) d(c, f_i^*) = (3 + 2\epsilon_1) d(c, \mu^*(c))$. Altogether, the total cost of the assignment μ is

$$\sum_{(c,w) \in W} w \cdot d(c, \mu(c)) \le \sum_{(c,w) \in W} w \cdot (3 + 2\epsilon_1) d(c, \mu^*(c)) = (3 + 2\epsilon_1)\, OPT.$$

The optimal assignment can only be better, hence the $(3+2\epsilon_1)$-approximation. This implies a $(1+\epsilon_0)(3+2\epsilon_1)$-approximation in time $\text{poly}(k \log n\, \epsilon_0^{-1} \epsilon_1^{-1})^{O(k)}$. Finally, setting $\epsilon_0, \epsilon_1 := \Theta(\epsilon)$, for $\Theta(\cdot)$ small enough, guarantees a $(3+\epsilon)$-approximation in time $(k \log n\, \epsilon^{-1})^{O(k)} n^{O(1)}$.

Lastly, we show that the $(\log n)^{O(k)}$ factor in the running time can be upper bounded by $k^{O(k)} n^{O(1)}$, proving the second running time in Theorem 1. If $k < \frac{\log n}{\log \log n}$, then $(\log n)^{O(k)} = (\log n)^{\frac{\log n}{\log \log n}} = n^{O(1)}$; otherwise, $k > \frac{\log n}{\log \log n} \ge \sqrt{\log n}$, so $(\log n)^{O(k)} \le (k^2)^{O(k)}$. Therefore, the running time in Theorem 1 is at most $O(k/\epsilon)^{O(k)} n^{O(1)}$.

For k-means, the algorithm and analysis are identical, except that the total cost is now

$$_{(c,w) \in W} w \cdot d(c, \mu(c))^2 \le \sum_{(c,w) \in W} w \cdot \big((3 + 2\epsilon_1) d(c, \mu^*(c))\big)^2 = (9 + O(\epsilon_1))\, OPT,$$

implying a $(9+\epsilon)$-approximation. This concludes the proof of Theorem 1.

3 A $(1+\epsilon)$-Approximation for Euclidean Inputs

3.1 The Continuous (Uniform-Capacity) Case – Proof of Theorem 3

In this section we consider the continuous case: namely the case where centers can be located at arbitary position in $\mathbb{R}^d$ and the capacities are uniform and $\eta \ge n/k$.

Let $\epsilon > 0$. Given a set of points P, denote by $\text{OPT}_1(P)$ the location of the optimal center of P (namely, the centroid of P in the case of the k-means problem or the median of P in the case of the k-median problem). We will make us of the following lemma of [20].

▶ **Lemma 21** (Lemma 5.3 in [20]). *Let P be a set of points in $\mathbb{R}^d$ and X be a random sample of size $O(\epsilon^{-3} \log(1/\epsilon))$ from P and a and b such that $a \le \mathsf{cost}(P, OPT_1(P)) \le b$. Then, we can construct a set Y of $O(2^{1/\epsilon^{O(1)}} \log(b/\epsilon a))$ points such that with constant probability there is at least one point $z \in X \cup Y$ satisfying $\mathsf{cost}(P, \{z\}) \le (1 + 2\epsilon)\mathsf{cost}(P, OPT_1(P))$. Further, the time taken to construct Y from X is $O(2^{1/\epsilon^{O(1)}} \log(b/\epsilon a) d)$.*

Our algorithm for obtaining a $(1+\epsilon)$-approximation is as follows:

1. Compute a coreset C for capacitated k-median as described by Lemma 21, and an estimate γ of the value of OPT using the classic $O(\log n)$-approximation.
   In the remaining, we assume that the minimum pairwise distance between pairs of points of C is at least $\epsilon\gamma/(n\log n)$ since otherwise one can simply take a net of the input and the additive error is at most ϵOPT (see e.g.: [11]). Moreover, we assume that there is no cluster containing only one point of the coreset since these clusters can be "guessed" and dealt with separately.
2. Start with $\mathcal{C}=\emptyset$, then for each subset S of C of size $O(\epsilon^{-3}\log(k/\epsilon))$, for each $s=(1+\epsilon)^i$ in the interval $[\epsilon\gamma/(n\log n),\gamma]$ apply the procedure of Lemma 21 with $a=s$ and $b=(1+\epsilon)a$ and add the output of the procedure to $\mathcal{C}$. We refer to $\mathcal{C}$ as a set of approximate candidate centers.
3. Consider all subsets of size k of $\mathcal{C}$. For each subset, compute the cost of using this set of centers for the capacitated k-median instance by using a min cost flow computation. Output the set of centers of minimum cost.

We first discuss the running time of the algorithm. The time for computing the coreset is polynomial by Theorem 11. Generating $\mathcal{C}$ takes $|C|^{O(\epsilon^{-3}\log(1/\epsilon))}\cdot 2^{1/\epsilon^{O(1)}}\log((1+\epsilon)/\epsilon)d$ time. For the last part, namely enumerating all subsets of $\mathcal{C}$ of size k, the running time is $|C|^{O(k\epsilon^{-3}\log(1/\epsilon))}\cdot 2^{k/\epsilon^{O(1)}}\log^k((1+\epsilon)/\epsilon)$. Theorem 11 implies that $|C|=\text{poly}(k\log n\,\epsilon^{-1})$ and so, the algorithm has running time $(k\log n\,\epsilon^{-1})^{k\epsilon^{-O(1)}}n^{O(1)}$. Again, the $(\log n)^{k\epsilon^{-O(1)}}$ factor can be upper bounded by $(k/\epsilon)^{k\epsilon^{-O(1)}}$ or $n^{O(1)}$ based on whether or not $k\epsilon^{-O(1)}<\frac{\log n}{\log\log n}$, hence the improved running time in Theorem 3.

We show that this algorithm provides a $(1+O(\epsilon))$-approximation. Theorem 11 immediately implies that the solution found for the coreset C can be lifted to a solution for the original input at a cost of an additive $O(\epsilon\text{OPT})$. For any (possibly weighted) set of client A and set of centers B, we define $\mathsf{cost}(A,B)$ to be the cost of the best assignment of the clients in A to the centers of B.

▶ **Lemma 22.** *The $\mathcal{C}$ computed by the algorithm contains a set of centers $\tilde{S}$ that is such that $\mathsf{cost}(C,\tilde{S})\leq(1+\epsilon)\mathsf{cost}(C,OPT)$.*

Proof. This follows almost immediately from Lemma 21. By Lemma 21, for each cluster C_i^* of OPT, there exists a set $S_i^*\subseteq C_i^*$ of size at most $O(\epsilon^{-3}\log(k/\epsilon))$ such that applying the procedure of Lemma 21 with the correct value of a to S_i^* yields a set of points containing a point z_i such that $\mathsf{cost}(C_i^*,z_i)\leq(1+2\epsilon)\mathsf{cost}(C_i^*,\text{OPT})$. Since the algorithm iterates over all subsets of size $O(\epsilon^{-3}\log(k/\epsilon))$, and that the pairwise distance is at least $\epsilon\text{OPT}/n$, it follows that S_i^* is one of the subset considered by the algorithm, and so z_i is part of $\mathcal{C}$. ◀

Finally, since the algorithm iterates over all subsets of $\mathcal{C}$ of size at most k, Lemma 22 implies that there exists a set $\{z_1,\ldots,z_k\}$ that is considered by the algorithm and on which solving a min cost flow instance yields a solution of cost at most $(1+O(\epsilon))\mathsf{cost}(\mathcal{P},\text{OPT})$.

3.2 The Non-Uniform Case – Proof of Theorem 2

We now consider the non-uniform case. In this setting, the input consists of a set of points in $\mathbb{R}^d$ together with a set of candidate centers in $\mathbb{R}^d$ and a capacity η_f for each such candidate center. We make use of the following lemma. As slightly worse bound for the lemma can also be found in [25].

▶ **Lemma 23** ([27]). *Let $\epsilon \in (0,1)$ and $X \subseteq \mathbb{R}^d$ be arbitrary with X having size $n > 1$. There exists $f : \mathbb{R}^d \mapsto \mathbb{R}^m$ with $m = O(\epsilon^{-2} \log n)$ such that $\forall x \in X$, $\forall y \in \mathbb{R}^d$, $||x-y||_2 \leq ||f(x)-f(y)||_2 \leq (1+\epsilon)||x-y||_2$.*

We describe a polynomial-time approximation scheme. Let $\epsilon > 0$. The algorithm is as follows. The first step of the algorithm is identical to the continous case.

1. Compute a coreset C for capacitated k-median as described by Theorem 21, and an estimate γ of the value of OPT using the classic $O(\log n)$-approximation.
   In the remaining, we assume that the minimum pairwise distance between pairs of points of C is at least $\epsilon\gamma/(n \log n)$ since otherwise one can simply take a net of the input and the additive error is at most ϵOPT (see e.g.: [11]). Moreover, we assume that there is no cluster containing only one point of the coreset since these clusters can be "guessed" and dealt with separately.
2. Apply Lemma 23 to the points of the coreset to obtain a set of points in a Euclidean space of dimension $\frac{\log k + \log\log n}{\epsilon^{O(1)}}$. Let C^* and A^* be respectively the image of the coreset points and of the candidate centers through the projection.
3. Start with $\mathcal{V} = \emptyset$ For each point p of the coreset do the following: For each $i \in \{1, 2, \ldots, n^2\}$, consider the ith-*ring* defined by $\mathsf{ball}(p, (1+\epsilon)^i \epsilon\gamma/(n \log n)) \setminus \mathsf{ball}(p, (1+\epsilon)^{i-1}\epsilon\gamma/(n \log n))$ and choose an $\epsilon \cdot (1+\epsilon)^i \epsilon\gamma/(n \log n)$-net. Consider the Voronoi diagram induced by the points of the net. Then, for each Voronoi cell, add to $\mathcal{V}$ the k candidate centers of A^* in the cell that are of maximum capacity.
4. Enumerate all possible subset of $\mathcal{V}$ of size k and output the one that leads to the solution of minimum cost.

3.2.1 Correctness

Theorem 11 implies that finding a near-optimal solution for the coreset points yields a near-optimal solution for the input point set.

Lemma 23 immediately implies that, given the coreset construction C, and the projection of the coreset points onto a $\frac{\log k + \log\log n}{\epsilon^{O(1)}}$-dimensional Euclidean space, finding a near-optimal set of centers in A^* yields a near-optimal set of centers in A through the inverse of the projection.

Therefore, it remains to show that the set $\mathcal{V}$ contains a set of candidate centers that yields a near-optimal solution. To see this, consider each center of the optimal solution in A^*. For each such optimal center f, consider the closest coreset point $c(f)$ together with the ring of $c(f)$ containing f. Let j be the index of this ring, namely $f \in \mathsf{ball}(p, (1+\epsilon)^j \epsilon\gamma/(n \log n)) \setminus \mathsf{ball}(p, (1+\epsilon)^{j-1}\epsilon\gamma/(n \log n))$.

By definition of the net, there exists a point p of the net at distance at most $\epsilon \cdot \mathsf{ball}(p, (1+\epsilon)^j \epsilon\gamma/(n \log n)) \leq 2\epsilon||c - c(f)||_2$ from $c(f)$. Therefore, consider the Voronoi cell of p and the top-k candidate centers in terms of capacity. If f is part of this top-k, then f is part of $\mathcal{V}$ and we are done. Otherwise, it is possible to associate to f a center f^* that has capacity at least the capacity of f, and so for all the optimal centers simultaneously since we consider the top-k. Therefore, consider replacing f by f^* in the optimal solution. The change in cost is at most, by the triangle inequality, $4\epsilon||c - c(f)||_2$ since both centers are in the Voronoi cell of p. Finally, since c is the closest client to $c(f)$, the cost increases by a factor at most $(1+4\epsilon)$ for each client and the correctness follows.

3.2.2 Running time

We now bound the running time. The first two steps are clearly polynomial time. An $\epsilon \cdot (1+\epsilon)^i \epsilon\gamma/(n \log n)$-net of a ball of radius $(1+\epsilon)^i \epsilon\gamma/(n \log n)$ has size $\epsilon^{-O(d)}$ and so in this context, after Step 2, a size $\epsilon^{-(\frac{\log k + \log\log n}{\epsilon^{O(1)}})}$. Since for each element of the net, k centers are chosen and since the number of rings is, by Step 1, at most $O(\epsilon^{-2} \log n)$, the total size of $\mathcal{V}$ is at most $|C| k \epsilon^{-2} \log n \epsilon^{-(\frac{\log k + \log\log n}{\epsilon^{O(1)}})}$ which is at most $|C| \epsilon^{-2} (k \log n)^{\epsilon^{-O(1)}} = (k\epsilon^{-1} \log n)^{\epsilon^{-O(1)}}$. Enumerating all subsets of size k takes time $(k\epsilon^{-1} \log n)^{k\epsilon^{-O(1)}}$ and the theorem follows.

References

1. M. Adamczyk, J. Byrka, J. Marcinkowski, S. M. Meesum, and M. Włodarczyk. Constant factor FPT approximation for capacitated k-median. *ArXiv e-prints*, September 2018. arXiv:1809.05791.
2. Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation Schemes for Euclidean k-Medians and Related Problems. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 106–113, 1998. doi:10.1145/276698.276718.
3. Jarosław Byrka, Krzysztof Fleszar, Bartosz Rybicki, and Joachim Spoerhase. Bi-factor approximation algorithms for hard capacitated k-median problems. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 722–736. SIAM, 2014.
4. Jarosław Byrka, Bartosz Rybicki, and Sumedha Uniyal. An Approximation Algorithm for Uniform Capacitated k-Median Problem with $1 + \epsilon$ Capacity Violation. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 262–274. Springer, 2016.
5. Moses Charikar, Chandra Chekuri, Ashish Goel, and Sudipto Guha. Rounding via Trees: Deterministic Approximation Algorithms for Group Steiner Trees and k-Median. In *STOC*, volume 98, pages 114–123. Citeseer, 1998.
6. Moses Charikar, Sudipto Guha, Éva Tardos, and David B Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65(1):129–149, 2002.
7. K. Chen. On Coresets for k-Median and k-Means Clustering in Metric and ELuclidean Spaces and Their Applications. *SIAM Journal on Computing*, 39(3):923–947, 2009.
8. Julia Chuzhoy and Yuval Rabani. Approximating K-median with Non-uniform Capacities. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, pages 952–958, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=1070432.1070569.
9. Vincent Cohen-Addad. Approximation Schemes for Capacitated Clustering in Doubling Metrics. *CoRR*, abs/1812.07721, 2018. arXiv:1812.07721.
10. Vincent Cohen-Addad, Arnaud de Mesmay, Eva Rotenberg, and Alan Roytman. The Bane of Low-Dimensionality Clustering. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 441–456, 2018. doi:10.1137/1.9781611975031.30.
11. Vincent Cohen-Addad, Anupam Gupta, Amit Kumar, Euiwoong Lee, and Jason Li. Tight FPT Approximations for k-Median and k-Means. In *ICALP 2019*, 2019.
12. Wenceslas Fernandez de la Vega, Marek Karpinski, Claire Kenyon, and Yuval Rabani. Approximation schemes for clustering problems. In Lawrence L. Larmore and Michel X. Goemans, editors, *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 50–58. ACM, 2003. doi:10.1145/780542.780550.
13. H. Gökalp Demirci and Shi Li. Constant Approximation for Capacitated k-Median with (1+epsilon)-Capacity Violation. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 73:1–73:14, 2016. doi:10.4230/LIPIcs.ICALP.2016.73.

14 G. Frahling and C. Sohler. Coresets in dynamic geometric data streams. In *STOC*, pages 209–217, 2005.

15 Anupam Gupta, Euiwoong Lee, and Jason Li. An FPT Algorithm Beating 2-approximation for K-cut. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 2821–2837, Philadelphia, PA, USA, 2018. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=3174304.3175483.

16 Anupam Gupta, Euiwoong Lee, Jason Li, Pasin Manurangsi, and Michal Wlodarczyk. Losing Treewidth by Separating Subsets. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1731–1749, 2019. doi:10.1137/1.9781611975482.104.

17 Sariel Har-Peled and Akash Kushal. Smaller Coresets for k-Median and k-Means Clustering. *Discrete & Computational Geometry*, 37(1):3–19, 2007. doi:10.1007/s00454-006-1271-x.

18 Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 291–300, 2004. doi:10.1145/1007352.1007400.

19 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. A Simple Linear Time (1+ ") - Approximation Algorithm for k-Means Clustering in Any Dimensions. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '04, pages 454–462, Washington, DC, USA, 2004. IEEE Computer Society. doi:10.1109/FOCS.2004.7.

20 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear-time approximation schemes for clustering problems in any dimensions. *J. ACM*, 57(2), 2010. doi:10.1145/1667053.1667054.

21 Euiwoong Lee. Partitioning a graph into small pieces with applications to path transversal. *Mathematical Programming*, March 2018. doi:10.1007/s10107-018-1255-7.

22 Shi Li. On Uniform Capacitated k-Median Beyond the Natural LP Relaxation. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 696–707, 2015. doi:10.1137/1.9781611973730.47.

23 Shi Li. Approximating capacitated k-median with $(1 + k$ open facilities. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 786–796, 2016. doi:10.1137/1.9781611974331.ch56.

24 Shi Li. On Uniform Capacitated k-Median Beyond the Natural LP Relaxation. *ACM Trans. Algorithms*, 13(2):22:1–22:18, 2017. doi:10.1145/2983633.

25 Sepideh Mahabadi, Konstantin Makarychev, Yury Makarychev, and Ilya Razenshteyn. Nonlinear Dimension Reduction via Outer Bi-Lipschitz Extensions. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 1088–1101, New York, NY, USA, 2018. ACM. doi:10.1145/3188745.3188828.

26 Dániel Marx and Michal Pilipczuk. Optimal Parameterized Algorithms for Planar Facility Location Problems Using Voronoi Diagrams. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 865–877, 2015. doi:10.1007/978-3-662-48350-3_72.

27 Shyam Narayanan and Jelani Nelson. Optimal terminal dimensionality reduction in Euclidean space. *CoRR – To appear in the proceedings of STOC'19*, abs/1810.09250, 2018. arXiv:1810.09250.

28 Yicheng Xu, Yong Zhang, and Yifei Zou. A constant parameterized approximation for hard-capacitated k-means. *CoRR*, abs/1901.04628, 2019. arXiv:1901.04628.

Tight FPT Approximations for k-Median and k-Means

Vincent Cohen-Addad
CNRS & Sorbonne Université,, Paris, France

Anupam Gupta
Carnegie Mellon University, Pittsburgh, PA, USA

Amit Kumar
IIT Delhi, India

Euiwoong Lee
New York University, NY, USA

Jason Li
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

We investigate the fine-grained complexity of approximating the classical k-Median/k-Means clustering problems in general metric spaces. We show how to improve the approximation factors to $(1 + 2/e + \varepsilon)$ and $(1 + 8/e + \varepsilon)$ respectively, using algorithms that run in fixed-parameter time. Moreover, we show that we cannot do better in FPT time, modulo recent complexity-theoretic conjectures.

2012 ACM Subject Classification Theory of computation → Facility location and clustering; Theory of computation → Fixed parameter tractability; Theory of computation → Submodular optimization and polymatroids

Keywords and phrases approximation algorithms, fixed-parameter tractability, k-median, k-means, clustering, core-sets

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.42

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1904.12334`.

Funding *Vincent Cohen-Addad*: Ce projet a bénéficié d'une aide de l'Etat gérée par l'Agence Nationale de la Recherche au titre du Programme Appel à projets générique JCJC 2018 portant la référence suivante: ANR-18-CE40-0004-01.
Anupam Gupta: Supported in part by NSF awards CCF-1536002, CCF-1540541, and CCF-1617790.
Euiwoong Lee: Supported in part by the Simons Collaboration on Algorithms and Geometry.
Jason Li: Supported in part by NSF awards CCF-1536002, CCF-1540541, and CCF-1617790.

Acknowledgements We thank Deeparnab Chakrabarty, Ola Svensson, and Pasin Manurangsi for useful discussions. This research was partially conducted when A. Kumar was visiting A. Gupta and Carnegie Mellon University as part of the Joint Indo-US Virtual Center for Algorithms under Uncertainty.

1 Introduction

How well can we approximate the k-Median and k-Means clustering problems? This question has been intensively studied over the past two decades, and many interesting algorithmic techniques have been developed and refined in an attempt to understand these problems. Let us elaborate for the k-Median problem; the story for k-Means is much the

same. Recall that in the k-Median problem, given a metric space (V, d) with n points and clients at some of the points, the goal is to open k *facilities* such that the sum of distances from the clients to their closest facilities is minimized.

The first constant-factor approximation algorithm for k-Median was given by Charikar et al. [6]. After many interesting developments (e.g., primal-dual schemes, sophisticated LP rounding schemes, and pseudo-approximations), today the best approximation guarantee is 2.611 [3]. The best lower bound, however, is still the $(1 + 2/e)$-hardness from 1998, due to Guha and Khuller [16]. In this paper, we ask: can we do better if we give ourselves more resources? The problem can be solved exactly by brute-force enumeration in time $n^{k+O(1)}$, but what can we do, say, in FPT time $f(k)n^{O(1)}$?

We cannot hope to solve the problem exactly in FPT time: the reduction of Guha and Khuller also shows a $W[2]$-hardness for finding the optimal solution for k-Median/k-Means exactly. Naturally, we then ask what we can achieve by combining the two approaches together, and whether good approximation algorithms can be given in FPT time.

Our Results. Our main algorithmic result is a positive result in this direction:

▶ **Theorem 1** (Algorithm for k-Median/k-Means)**.** *For every $\varepsilon > 0$, there is a $(1+2/e+\varepsilon)$-approximation algorithm for the k-Median problem, that runs in time FPT time, i.e., in $f(k, \varepsilon)n^{O(1)}$ time. For the k-Means problem, we can achieve a $(1 + 8/e + \varepsilon)$-approximation in the same runtime.*

The approximation guarantees in Theorem 1 match the NP-hardness results for the two problems implied by [16]. However, since we are allowing ourselves FPT time and not just $\text{poly}(n, k)$ time, can we do even better and go past this NP-hardness barrier? Our second main result shows that this is not possible, at least under recent complexity-theoretic conjectures. We prove that the results in Theorem 1 are essentially tight, assuming the Gap-Exponential Time Hypothesis [12, 24, 5]:

▶ **Theorem 2** (Hardness)**.** *There exists a function $g : \mathbb{R}^+ \to \mathbb{R}^+$ such that assuming the Gap-ETH, for any $\varepsilon > 0$, any $(1 + 2/e - \varepsilon)$-approximation algorithm for k-Median, and any $(1 + 8/e - \varepsilon)$-approximation for k-Means, must run in time at least $n^{k^{g(\varepsilon)}}$.*

The basic component of the above hardness result is an FPT-hardness of a factor of $(1 - 1/e)$ for the Max k-Coverage problem, again using the Gap-ETH (Theorem 15). Composing that hardness result with the reduction of Guha and Khuller [16] gives us Theorem 2 above.

Matroid Median. Finally, using our algorithmic techniques, we are able to also give an improved approximation for the matroid-median problem, which is a generalization of the k-Median problem.

▶ **Theorem 3** (Algorithm for Matroid Median)**.** *There is a $(2+\varepsilon)$-approximation algorithm for the* Matroid Median *problem, that runs in time FPT time, i.e., in $f(k, \varepsilon)n^{O(1)}$ time.*

Since the Matroid Median problem is a generalization of the k-Median problem, the $(1 + 2/e - \varepsilon)$-hardness from Theorem 2 translates immediately to Matroid Median. It remains an open problem to close the gap between this lower bound and the $(2 + \varepsilon)$-approximation in Theorem 3. We can also use our ideas to get an $(3 - \frac{2}{p+1} + \varepsilon)$ for the p-Matroid Median problem.

Facility Location. Facility Location is a problem closely related to k-Median, where each facility has an opening cost and the goal is to open facilities to minimize the sum of distances from clients to their closest facilities plus the sum of the total opening costs. For this problem, the best known hardness ratio is $\alpha_{\mathsf{FL}} \approx 1.463$ [16], which is defined to be $\max_{x \geq 0} \left(1 + \frac{x}{1+x} \ln \frac{2}{x}\right)$. On the other hand, the best algorithm achieves an 1.488-approximation [21]. When the parameter k denotes the number of facilities open in the optimal solution, we prove that our techniques also give an FPT algorithm for Facility Location whose approximation ratio matches the hardness ratio of [16].

▶ **Theorem 4** (Algorithm for Facility Location). *There is a $(\alpha_{\mathsf{FL}} + \varepsilon)$-approximation algorithm for the* Facility Location *problem, that runs in time FPT time, i.e., in $f(k, \varepsilon) n^{O(1)}$ time.*

Roadmap. In Section 2, we describe the approximation algorithms for these problems. We assume throughout that the aspect ratio is polynomially bounded. (In the full version of the paper, we show that this assumption is without loss of generality, in the case we consider where the clients have unit weights.) In Section 3, we then give the hardness results for FPT Max k-Coverage, k-Median, and k-Means.

1.1 Our Techniques

The algorithm is inspired by the hardness result from [16]: it relies on the result of Feige [13] that Max k-Coverage is hard to approximate better than $(1 - 1/e)$. Hence, if we build a "factor graph" with sets on one side and elements on another, with edges indicating inclusion, picking k sets covers $(1 - 1/e)$ elements at distance 1, and the remaining at distance at least 3 – hence $1 + 2/e$. Now what if we have a general instance, with different distances? We show how to do limited enumeration (in FPT) time to restrict our choices to picking one facility each from k disjoint sets. Moreover, via a surprisingly clean idea we can model the objective as submodular maximization (subject to a partition matroid constraint). And this problem can be approximated well: the factor again is $(1 - 1/e)$, hence giving the same factor upto additive ε terms!

The matching hardness result is via showing an FPT hardness for Max k-Coverage assuming the Gap-ETH. Firstly, we show that assuming the Gap-ETH, there is no FPT approximation algorithm for Label Cover problem parameterized by the number of vertices k on one side of the bipartition. (Trying all labelings on one side takes time $O(n^{k+O(1)})$, and doing much better is hard.) To do this, we construct a *variable-clause game* from a 3-SAT instance, merge *clause vertices* into ℓ super-vertices, and then use r rounds of parallel repetition. (The number of clause vertices becomes $k := \ell^r$.) Then we compose this with the classical reduction from Label Cover to Max k-Coverage [13]. Due to some technical details (e.g., our Label Cover instance is not guaranteed to be regular) and for the sake of completeness, we provide a formal proof in Lemma 19. While our techniques are similar to recent FPT hardnesses for the related k-Dominating Set problem [5, 10], some technical details (e.g., the *projection property* of Label Cover instances) prevent us from directly using prior results to get $(1 - 1/e + \varepsilon)$-hardness for Max k-Coverage.

1.2 Related Work

We briefly survey the state-of-the-art for k-Median and k-Means; please see references below for more historical context. For general metric spaces, the best approximation ratio for k-Median is 2.611 [3] by Byrka et al., building on work of Li and Svensson [22]. Kanungo et al. [17] gave a $(9 + \varepsilon)$-approximation algorithm for k-Means in general metric spaces, which

was later improved to 6.357 by Ahmadian et al. [1]. The first constant factor approximation algorithm for Matroid Median was given by Krishnaswamy et al. [18], which was improved by Swamy [26] to 8.

For Euclidean spaces, the problems are better approximable, at least when either k or the dimension d are fixed; we restrict this discussion to parameterizing by k. Specifically, PTASs for both k-Median and k-Means with running time $f(k,\varepsilon)\operatorname{poly}(n,d)$ were given by Kumar et al. [19]. The running times were improved by Chen [7] to $O(nkd + d^2 n^\sigma 2^{(k/\varepsilon)^{O(1)}})$ for any $\sigma > 0$ for k-Median, and by Feldman et al. [15] to $O(nkd + d\operatorname{poly}(k/\varepsilon) + 2^{\tilde{O}(k/\varepsilon)})$ for k-Means. Both these latter results were based on the notion of coresets. The k-Means problem is APX hard even in Euclidean space, if both k and d are allowed to be arbitrary [2, 20].

A result of direct interest to this work is that of Czumaj and Sohler [11], for the *min-sum clustering problem*. They give a $(4+\varepsilon)$-approximation on general metrics in FPT time. They construct a small (strong) core-set for the related Balanced k-Median problem, and enumerate over all choices of centers inside this core-set. In the full version, we show that their approach extends to give a 2-approximation for the *non-bipartite case* of k-Median– in this special case of k-Median a facility may be opened at any client location, and hence $C \subseteq \mathbb{F}$. Theorem 1 above shows how to get a better guarantee for a more general case. (As an aside, the hardness for this special non-bipartite case is only $(1+1/e)$; closing this gap is another interesting open question.)

Hardness-of-approximation results for parameterized problems have been actively studied recently. Lin [23] proved $W[1]$-hardness of approximation for k-Biclique. Chen and Lin [9] proved $W[1]$-hardness of approximation for k-Dominating Set in any constant factor, which was later improved to any function $f(k)$ in [5, 10]. Chalermsook et al. [5] also proved that there is no FPT $o(k)$-approximation algorithm for k-Clique assuming the Gap-ETH.

1.3 Preliminaries

An instance $\mathcal{I}$ of the k-Median problem is defined by a tuple $((V,d), C, \mathbb{F}, k)$, where (V,d) is a metric space over a set of points V with $d(i,j)$ denoting the distance between two points i, j in V. Further, C and $\mathbb{F}$ are subsets of V and are referred as "clients" and "facility locations", and k is a positive parameter. The goal is to find a subset F of k facilities in $\mathbb{F}$ to minimize

$$\mathsf{cost}(C,F) := \sum_{j \in C} d(j,F).$$

In the *weighted* version of k-Median, every client $j \in C$ has an associated weight w_j, and the goal is to find a subset F of $\mathbb{F}$ of size k such that $\mathsf{cost}(C,F) := \sum_{j\in C} w_j d(j,F)$ is minimized.

The k-Means problem is defined similarly except that the objective function gets modified to $\mathsf{cost}(C,F) := \sum_{j\in C} d(j,F)^2$ (and analogously for the weighted version). The names of the two problems come from the fact that if the metric space is the real line and $k=1$, the optimal solution is the median and the mean respectively. In the Matroid Median problem, we are given a matroid on the set $\mathbb{F}$, and the set of open facilities must be an independent set in the matroid. Again, the goal is to minimize the assignment cost of clients to the nearest open facility.

In the Facility Location problem, an instance is not given k, but additionally has $\mathsf{open} : \mathbb{F} \to \mathbb{R}^+$ that indicates the opening cost of each facility. The goal is to find a subset $F \subseteq \mathbb{F}$ (without any restriction on $|F|$) that minimizes $\mathsf{open}(F) + \mathsf{cost}(C,F)$ where $\mathsf{open}(F) := \sum_{f\in F} \mathsf{open}(f)$.

Finally, the *aspect ratio* of a metric space (V,d) is $\Delta := \frac{\max_{x,y\in V} d(x,y)}{\min_{x,y\in V} d(x,y)}$.

2 The Approximation Algorithm

We now give the $(1+2/e+\varepsilon)$-approximation algorithm for k-Median, where $\varepsilon > 0$ is a fixed parameter throughout this section. The running time of the algorithm is $f(k,\varepsilon) \cdot \text{poly}(n)$, where $f(k,\varepsilon) = O(\varepsilon^{-2} k \log k)^k$. We then indicate the alterations to get algorithms for k-Means and Matroid Median.

2.1 The Intuition

We focus on k-Median for now; the ideas for the other problems are analogous. The first idea is to reduce the size of the client set C to $O(\varepsilon^{-2} k \log n)$– this can be done by results on core-sets for k-Median, which consolidate the clients into a small number of distinct locations [8, 14]. The consolidated clients now have weights, but this extension to weighted k-Median does not pose a problem.

The next idea is to carefully enumerate over the structure of an optimal solution. Consider an optimal solution $F^\star = \{f_1^\star, \ldots, f_k^\star\}$. For a facility $f_i^\star \in F^\star$, let "cluster" $C_i^\star$ be the clients assigned to $f_i^\star$, i.e., the subset of clients C for which $f_i^\star$ is closest open facility. Let ℓ_i be the client in $C_i^\star$ closest to $f_i^\star$ – we call it the *leader* of cluster $C_i^\star$. Let R_i be the distance $d(f^\star, \ell_i)$, suitably discretized. Our algorithm guesses the leaders ℓ_i and the distances R_i for each $i \in [k]$. Since the size of C is $O(\varepsilon^{-2} k \log n)$, there are $(O(\varepsilon^{-2} k \log n))^k$ choices for leaders,[1] and a similar number of choices for the distances; moreover, this quantity can be shown to be $f(k,\varepsilon) \cdot n^{O(1)}$.

Assume now that we have correctly guessed the leaders ℓ_i and distances R_i. For each leader ℓ_i, let F_i be the facilities at distance about R_i from ℓ_i – this set F_i contains $f_i^\star$. By making copies, assume the sets F_i are disjoint. Now our task is to select one facility from each set F_i such that the total (weighted) assignment cost of the clients in C is minimized. As such, this seems like a decreasing *supermodular minimization* problem with a (partition) matroid constraint. (Observe that choosing an arbitrary center in each F_i gives us a 3-approximation in FPT time, but we want to do much better.)

The last idea is to convert this into a monotone submodular maximization problem, again with a partition matroid constraint. For each set F_i, we add a *fictitious facility* f_i' such that (i) the assignment cost of clients to the fictitious facilities is at most $3OPT$, and (ii) for a subset S of facilities, the "improvement" $\mathsf{cost}(C, F') - \mathsf{cost}(C, F' \cup S)$, where F' is the set of fictitious facilities, is a monotone submodular function. We finally show that a $(1-1/e)$-approximation for this submodular maximization problem gives the desired approximation guarantee. The next two sections describe the algorithm for k-Median in detail. The extension to k-Means, Matroid Median and Facility Location then appears in the full version of the paper.

2.2 Client Reduction via Coresets

Consider an instance $\mathcal{I} = ((V,d), C, \mathbb{F}, k)$ of the k-Median problem. Let $\varepsilon > 0$ be a fixed constant. We now define the notion of core-sets and use known results to reduce the size of C to (a weighted) a set of size $O(\varepsilon^{-2} k \log n)$.

[1] Our analysis will tighten this bound to $O(\varepsilon^{-2} \log n)^k$, but this improvement can be ignored for this intuition section.

▶ **Definition 5** (Core-set). *A* (strong) core-set *for $\mathcal{I}$ is a set of clients $C' \subseteq V$ along with weights w_j for all $j \in C'$, such that*

$$\sum_{j \in C'} w_j \, d(j, F) \in (1-\varepsilon, 1+\varepsilon) \cdot \sum_{j \in C} d(j, F),$$

for every $F \subseteq \mathbb{F}$ with $|F| = k$.

A similar definition holds for a strong core-set for the k-Means problem. Since we deal only with strong core-sets in this paper, we drop the modifier and refer to them only as core-sets. The first core-sets for metric k-Median were given by Chen [8]; the following result is the best current construction:

▶ **Theorem 6** ([14], Theorem 15.4). *For $0 \leq \varepsilon, \delta \leq 1/2$, there exists a Monte Carlo algorithm that for each instance I of k-Median on a general metric, outputs a core-set $C' \subseteq C$ with size*

$$|C'| = O\Big(\frac{k \log n + \log 1/\delta}{\varepsilon^2}\Big)$$

with probability $1-\delta$, where $n = |V|$. Moreover, the algorithm runs in time $O(k(n+k) + \log^2(1/\delta) \log^2 n)$. For k-Means, the core-set is of size $|C'| = O\big(\frac{k \log n + \log 1/\delta}{\varepsilon^4}\big)$, and the runtime remains the same.

The power of core-sets lies in the following fact.

▷ Fact 7. Consider a k-Median/k-Means instance $\mathcal{I} = ((V,d), C, \mathbb{F}, k)$, and let C' be a (strong) core-set with weights w. Consider the weighted instance $\mathcal{I}' = ((V,d), C', \mathbb{F}, k, w)$, which is the instance $\mathcal{I}$ with its clients replaced by the weighted clients in the core-set. Then, for any $\beta \geq 1$, a β-approximate solution $F \subseteq \mathbb{F}$ to $\mathcal{I}'$ is a $\beta(1+O(\varepsilon))$-approximate solution to $\mathcal{I}$.

Therefore, in order to find a $(1+2/e+O(\varepsilon))$-approximation to a k-Median $\mathcal{I}$, it suffices to find a $(1+2/e+O(\varepsilon))$-approximation to $\mathcal{I}'$, and analogously for k-Means. Henceforth, we restrict our attention to the core-set instance $\mathcal{I}'$. In other words, we assume that our instances have only a small number of clients, but now the clients have associated weights. In the following sections, we show how to approximate such weighted k-Median/k-Means instances in FPT time.

2.3 Reduction to Submodular Maximization

Given Fact 7, we only consider instances $\mathcal{I} = ((V,d), C, \mathbb{F}, k, w)$ of weighted k-Median, where clients in C have weights in the range $[1, n]$ and $|C|$ is bounded by $O(\varepsilon^{-2} k \log n)$. In this section we prove the following approximation guarantee for k-Median; this, combined with Fact 7, proves the k-Median statement in Theorem 1.

▶ **Theorem 8.** *Let ε be a fixed parameter. Given a k-Median instance $\mathcal{I} = ((V,d), C', \mathbb{F}, k, w)$ with $|C'| = O(\varepsilon^{-2} k \log n)$, there is a $(1+2/e+O(\varepsilon))$-approximation algorithm that runs in $f(k,\varepsilon) n^{O(1)}$ time.*

By scaling, assume the minimum distance between points in V is 1, so the aspect ratio Δ is the maximum distance between two points in V. For a positive integer a, define $[\![a]\!] := (1+\varepsilon)^{\lceil \log_{(1+\varepsilon)} a \rceil}$ as the smallest power of $(1+\varepsilon)$ larger than or equal to a. Here, ε is the same fixed parameter as the one used in the core-set.

The formal algorithm follows the intuition in §2.1 and is described in Algorithm 2.1; let us step through it now. We iterate over all possible values $\ell_1, \ldots, \ell_k$ for the leaders, and $R_1, \ldots, R_k$ for the corresponding distances. The same vertex could appear several times in the subset $\{\ell_1, \ldots, \ell_k\}$, and so the latter should be thought of as a multi-set. In Step 7, we add k new fictitious facilities: for each i, the new facility f_i' is at distance $2R_i$ from all the facilities in F_i. The distance to all other points is determined by triangle inequality in Step 8. Claim 9 shows that this forms a valid metric. In Step 9, we define the "improvement" function $\mathsf{improv}(S)$ as the reduction in cost due to adding in the facilities in S. Claim 10 shows this function is monotone submodular. This means we can use the $(1 - 1/e)$-approximation algorithm [4] for monotone submodular function maximization subject to a matroid constraint to find a set S which contains exactly one facility from each of the sets F_i, since this is a partition matroid constraint. Observe that the function $\mathsf{improv}(\cdot)$ can be computed efficiently. This completes the description of the algorithm.

Algorithm 2.1 `FindCenters`.

1: **for** every multi-set $\{\ell_1, \ell_2, \ldots, \ell_k\} \subseteq C$ **do**
2: **for** every $R_1, \ldots, R_k$ such that $R_i \in [1, \ldots, \lceil\Delta\rceil]$ and R_i is a power of $(1+\varepsilon)$ **do**
3: $F_i \leftarrow \{f \in F \mid \lceil d(f, \ell_i)\rceil = R_i\}$
4: make copies of facilities to ensure that $F_1, \ldots, F_k$ are disjoint
5: Initialize $F' \leftarrow \varnothing$; F' will be the set of fictitious facilities
6: **for** $i = 1, \ldots, k$ **do**
7: add a new fictitious facility f_i' to F' and set $d(f_i', f) := 2R_i$ for all $f \in F_i$
8: define $d(f_i', v) := \min_{f \in F_i}(d(f_i', f) + d(f, v))$ for all other points v
9: define $\mathsf{improv}(S) := \mathsf{cost}(C, F') - \mathsf{cost}(C, S \cup F')$ for every set $S \subseteq \mathbb{F}$
10: find $S \subseteq \mathbb{F}$ approximately maximizing $\mathsf{improv}(S)$, such that $|S \cap F_i| = 1$
11: among all sets S computed in line 10, output S for which $\mathsf{cost}(C, S)$ is minimized.

To prove correctness of the algorithm, we need to show two things: the distance function defined on $F' \cup V$ in Step 8 is a metric, and the function improv defined in Step 9 is monotone and submodular. We defer the simple proofs to the full version of the paper.

▷ Claim 9. Consider the set F' defined during an iteration of the algorithm. The distance function defined on $F' \cup V$ is a metric.

▷ Claim 10. The function $\mathsf{improv}(S)$ defined in Step 9 is monotone and submodular with $\mathsf{improv}(\varnothing) = 0$.

Now to bound the runtime. Since $|C| = O(\varepsilon^{-2} k \log n)$, there are at most $\binom{O(\varepsilon^{-2} k \log n)+k-1}{k} = (O(\varepsilon^{-2} \log n))^k$ different multi-sets of size k with elements in C. In addition, there are $\log_{1+\varepsilon} \Delta$ many choices for R_i for each $i \in [k]$. Therefore, the number of iterations in Step 1 of the algorithm can be bounded by

$$(O(\varepsilon^{-2} \log n))^k \cdot (\log_{1+\varepsilon} \Delta)^k \leq \left(O\left(\frac{(\log \Delta)(\log n)}{\varepsilon^2}\right)\right)^k. \tag{2.1}$$

Since we started with the unweighted k-Median problem, the aspect ratio Δ can be assumed to polynomially bounded in n, and so the number of iterations can be bounded by $(O(\log n/\varepsilon^2))^k$, which is at most $n \cdot (O(\varepsilon^{-2} k \log k))^k$. Indeed, in case $k < \frac{\log n}{\log\log n}$, $(O(\log n/\varepsilon^2))^k \leq (O(1/\varepsilon^2))^k \cdot (\log n)^{\frac{\log n}{\log\log n}} = (O(1/\varepsilon^2))^k \cdot n$. Else $\log n \leq O(k \log k)$, and hence $(O(\log n/\varepsilon^2))^k = (O(k \log k/\varepsilon^2))^k$.

The algorithm for submodular maximization subject to a matroid constraint takes polynomial time, given a value oracle for the function [4, Theorem 1.1]: in fact it can be sped up for the case of partition matroid constraints [4, §3.3]. The value oracle for $\mathsf{improv}(S)$ can itself be implemented in polynomial time. Hence each iteration of the algorithm can be run in time polynomial in n.

The submodular maximization algorithm is a randomized Monte-Carlo algorithm that succeeds with only probability $1 - 1/n^2$, but we can easily boost the success probability by repetition: by running it $\tau := \mathrm{poly}(\varepsilon^{-1} k \log n)$ times for each input S and returning the maximum value obtained, we can ensure that with high probability it succeeds in all the calls we make.

2.3.1 Approximation Ratio

We now argue about the approximation ratio of the algorithm. We fix an optimal solution to the instance. Let $F^\star = \{f_1^\star, \ldots, f_k^\star\}$ be the centers opened by this solution. Define $C_i^\star$ as the clients for which the closest open center is $f_i^\star$, i.e., $C_i^\star := \{j \in C : d(j, f_i^\star) = d(j, F^\star)\}$. We define the notion of leaders with respect to this solution.

▶ **Definition 11** (Leader). *For each $i \in [k]$, call a client $j \in C_i^\star$ that minimizes $d(j, f_i^\star)$ over all $j \in C_i^\star$ the* leader *$\ell_i^\star$ of center $f_i^\star$. If there are multiple clients $j \in C_i^\star$ achieving the minimum, declare an arbitrary one to be the leader. Note that a client can be the leader of multiple centers $f_i^\star$. The* leaders *w.r.t. the solution $F^\star$ is the multi-set $\{\ell_1^\star, \ldots, \ell_k^\star\}$. For each leader $\ell_i^\star$, the* radius *$R_i^\star$ is defined as $\lceil d(\ell_i^\star, f_i^\star) \rceil$.*

Consider the iteration of Algorithm 2.1 where $\ell_1, \ldots, \ell_k$ are equal to $\ell_1^\star, \ldots, \ell_k^\star$ respectively, and $R_1, \ldots, R_k$ are equal to $R_1^\star, \ldots, R_k^\star$ respectively. Let $S^\star$ be the set output in Step 10 of the algorithm. It suffices to show that $\mathsf{cost}(C, S^\star) \le (1 + 2/e + \varepsilon)\mathsf{cost}(C, F^\star)$. We proceed to show this in the rest of the section.

As in the algorithm, define

$$F_i := \{f \in F \mid \lceil d(f, \ell_i^\star) \rceil = R_i^\star\},$$

so that $f_i^\star \in F_i$ for each $i \in [k]$. (Recall that the sets F_i are disjoint by duplicating facilities.) Let $F' = \{f_1', \ldots, f_k'\}$ be the set of fictitious facilities defined in the algorithm.

We are interested in the solutions S that consist of one center from each F_i, since one such solution is the desired $F^\star$. More formally, define a solution S to be *valid* if the set S can be listed as $(f_1, \ldots, f_k)$ so that $f_i \in F_i$ for each $i \in [k]$.

▷ Claim 12. For every valid S, $\mathsf{cost}(C, F' \cup S) = \mathsf{cost}(C, S)$.

Proof. List the set S as $(f_1, \ldots, f_k)$, where $f_i \in F_i$ for each $i \in [k]$. Informally, this claim amounts to showing that the fictitious facilities F' do not improve the solution S. To formalize this idea, fix a client $j \in C$ and a fictitious facility f_i', and let $f \in F_i$ be a closest center to j in F_i. Below, we show that in fact, client j is closer to $f_i \in S$ than to $f_i' \in F'$:

$$d(j, f_i) \overset{(\triangle \text{ ineq.})}{\le} d(j, f) + d(f, \ell_i^\star) + d(\ell_i^\star, f_i) \le d(j, f) + R_i^\star + R_i^\star = d(j, f) + d(f, f_i') = d(j, f_i').$$

Therefore, we have $d(j, F') \ge d(j, S)$ for all clients j, so

$$\mathsf{cost}(C, F' \cup S) = \sum_{j \in C'} w_j \, d(j, F' \cup S) = \sum_{j \in C'} w_j \, d(j, S) = \mathsf{cost}(C, S),$$

as desired. ◁

We now bound the cost of the solution which opens facilities at F'.

▷ Claim 13. $\mathsf{cost}(C, F') \le (3+2\varepsilon)\,\mathsf{cost}(C, F^\star)$.

Proof. It suffices to show that $d(j, F') \le (3+2\varepsilon)\,d(j, F^\star)$ for each client $j \in C$. Fix a client $j \in C$, and let $f_i^\star \in F^\star$ be a center achieving $d(j, f_i^\star) = d(j, F^\star)$. Since $\ell_i^\star$ is the leader of center $f_i^\star$, we have

$$d(j, f_i^\star) \ge d(\ell_i, f_i^\star) \ge \frac{R_i^\star}{1+\varepsilon}. \tag{2.2}$$

Recall that $f_i^\star \in F_i$. Therefore,

$$\begin{aligned} d(j, F') \le d(j, f_i') &\overset{(\triangle)}{\le} d(j, f_i^\star) + d(f_i^\star, f_i') = d(j, f_i^\star) + 2R_i^\star \\ &\overset{(2.2)}{\le} d(j, f_i^\star) + 2(1+\varepsilon)\,d(j, f_i^\star) \le (3+2\varepsilon)\,d(j, F^\star), \end{aligned}$$

as desired. ◁

Let $S^\star$ be the set output in Step 11. Since the algorithm of [4] is $(1-1/e)$-approximation,

$$\mathsf{improv}(S^\star) \ge (1-1/e)\mathsf{improv}(F^\star) \tag{2.3}$$

▶ **Lemma 14.** *The solution $S^\star$ in (2.3) satisfies* $\mathsf{cost}(C, S^\star) \le (1+2/e+O(\varepsilon))\,\mathsf{cost}(C, F^\star)$.

Proof. We bound the cost associated with this solution as follows.

$$\begin{aligned} \mathsf{cost}(C, S^\star) &\overset{(\text{Lem } 12)}{=} \mathsf{cost}(F' \cup S^\star) = \mathsf{cost}(C, F') - \mathsf{improv}(S^\star) \\ &\overset{(2.3)}{\le} \mathsf{cost}(C, F') - (1-1/e)\,\mathsf{improv}(F^\star) \\ &= \mathsf{cost}(C, F') - (1-1/e)\,(\mathsf{cost}(C, F') - \mathsf{cost}(C, F^\star)) \\ &= (1/e)\,\mathsf{cost}(C, F') + (1-1/e)\,\mathsf{cost}(C, F^\star) \\ &\overset{(\text{Lem } 13)}{\le} (3+2\varepsilon)(1/e)\,\mathsf{cost}(C, F^\star) + (1-1/e)\,\mathsf{cost}(C, F^\star) \\ &= (1+2/e+O(\varepsilon))\,\mathsf{cost}(C, F^\star). \end{aligned} \tag{2.4}$$

Hence the proof. ◀

2.3.2 Putting it all together

Our algorithm is a Monte Carlo randomized algorithm: both our subroutines use randomness. The first is the core-set construction in §2.2, and the second is the submodular maximization procedure in Step 10 of the algorithm. For each, we can make the error probability $1/\operatorname{poly}(n)$. Since each iteration of the algorithm can be implemented in $\operatorname{poly}(n)$ time, the runtime is dominated by the number of iterations, which is $(O(\varepsilon^{-2}k\log k)^k \operatorname{poly}(n))$. Moreover, combining the two steps of finding the core-set and the submodular maximization, the approximation ratio is $(1+\varepsilon)(1+2/e+O(\varepsilon)) = 1+2/e+O(\varepsilon)$. This proves Theorem 1 for the k-Median problem.

3 Gap-ETH Hardness of Max k-Coverage

In this section, we show that assuming the Gap Exponential Time Hypothesis (Gap-ETH) [12, 24], for any $\varepsilon > 0$, there is no FPT-approximation algorithm that approximates Max k-Coverage better than a factor $(1-1/e+\varepsilon)$.

▶ **Theorem 15** (Hardness for Max-Coverage). *There exists a function $g : \mathbb{R}^+ \to \mathbb{R}^+$ such that assuming the Gap-ETH, for any $\varepsilon > 0$, any $(1 - 1/e + \varepsilon)$-approximation algorithm for* Max k-Coverage *with n elements and m sets must run in time at least $(n+m)^{k^{g(\varepsilon)}}$.*

Using the reduction of Guha and Khuller [16], this immediately implies Theorem 2. The rest of the section is devoted to the proof of Theorem 15. The proof has two main components: the first part shows under the Gap-ETH, it takes at least $n^{h(k)}$ time to approximate the Label Cover problem even when one side of the bipartition has only k vertices; here $h(\cdot)$ is some increasing function depending on the quality of approximation. This reduction is inspired by the recent progress on the hardness of parameterized problems [5, 10] and was communicated to us by Pasin Manurangsi. The second part is the classical reduction from Label Cover to Max k-Coverage given by Feige [13].

3.1 Hardness of Label Cover from Gap-ETH

We begin with the standard definition of Label Cover.

▶ **Definition 16** (Label Cover). *An instance of* Label Cover $\mathcal{L}$ *consists of a bipartite graph $G = (U \cup V, E)$ with possibly parallel edges, two label sets Σ_U, Σ_V, and a projection $\pi_e : \Sigma_U \to \Sigma_V$ for each $e \in E$. Given a labeling $\sigma : (U \cup V) \to (\Sigma_U \cup \Sigma_V)$, an edge $e = (u, v) \in E$ is satisfied when $\pi_e(\sigma(u)) = \sigma(v)$. The goal of* Label Cover *is to find a labeling σ that maximizes the number of satisfied edges. Let $\mathsf{OPT}(\mathcal{L})$ be the maximum fraction of edges simultaneously satisfied by any labeling.*

Note that we include the *projection property* in the definition; all Label Cover instances in the paper will have this property. For a vertex $u \in U \cup V$, let d_u be the degree of u, and let d_U (resp. d_V) be the maximum degree of U (resp. V). We also call an instance *U-regular* (resp. *V-regular*) if all vertices in U (resp. V) have the same degree. All subsequent Label Cover instances will be U-regular, though the lack of V-regularity will require us to do a little more work in §3.2.

Given a 3-SAT formula ϕ, let $\mathsf{OPT}(\phi)$ be the maximum fraction of clauses that can be satisfied by any assignment. The Gap-ETH [12, 24] states that there exist some constants $\delta > 0, s < 1$ for which no algorithm, given a 3-SAT formula ϕ on n variables and $m = O(n)$ clauses, can distinguish whether $\mathsf{OPT}(\phi) = 1$ or $\mathsf{OPT}(\phi) < s$ in time $O(2^{\delta n})$. The main result of this subsection is the following lemma.

▶ **Lemma 17.** *For every $\ell, r \in \mathbb{N}$, there is a reduction that, given 3-SAT formula ϕ with n variables and m clauses, outputs a U-regular* Label Cover *instance $\mathcal{L}$ such that*

- *(Completeness)* $\mathsf{OPT}(\phi) = 1 \implies \mathsf{OPT}(\mathcal{L}) = 1$, *and*
- *(Soundness)* $\mathsf{OPT}(\phi) < s \implies \mathsf{OPT}(\mathcal{L}) < s^{\Omega(r)}$,

where $|U| = \ell^r, |V| = n^r, |\Sigma_U| = 2^{O(mr/\ell)}, |\Sigma_V| = 2^{O(r)}, d_V \le m^r$. The running time of this reduction is $m^{O(r)} \cdot |\Sigma_U|$.

In particular, assuming Gap-ETH, for any $\eta > 0$, if we let $r = \Theta(\log(1/\eta))$ so that

$$|\Sigma_U|^{|U|^{O(1/\log(1/\eta))}} = |\Sigma_U|^{|U|^{1/2r}} = |\Sigma_U|^{\ell^{1/2}} = 2^{O(mr/\sqrt{\ell})},$$

no algorithm can take a Label Cover *instance $\mathcal{L}$ and can decide whether $\mathsf{OPT}(\mathcal{L}) = 1$ or $\mathsf{OPT}(\mathcal{L}) < \eta$ in time $|\Sigma_U|^{|U|^{O(1/\log(1/\eta))}}$.*

Note that a brute-force algorithm that tries every assignment to U and chooses the best assignment for V for it runs in $O(|\Sigma_U|^{|U|})$ times a polynomial. Lemma 17 shows that assuming the Gap-ETH, even approximately solving Label Cover requires significant time.

Lemma 17 is proved by a series of well-known transformations between Label Cover instances. We start with the following basic hardness result for Label Cover assuming the Gap-ETH, which follows from essentially restating Gap-ETH as a *clause-variable* game:

▶ **Theorem 18** (Theorem 4.1 of [5]). *There is a reduction that, given 3-SAT formula ϕ with n variables and m clauses, outputs a U-regular* Label Cover *instance $\mathcal{L}$ such that*

- *(Completeness)* $\mathsf{OPT}(\phi) = 1 \implies \mathsf{OPT}(\mathcal{L}) = 1$, *and*
- *(Soundness)* $\mathsf{OPT}(\phi) < s' \implies \mathsf{OPT}(\mathcal{L}) < s = 1 - (1 - s')/3$,

where $|U| = m, |V| = n, |\Sigma_U| = 7, |\Sigma_V| = 2, d_V \leq m$, and G is U-regular with $d_U = 3$. In particular, assuming the Gap-ETH, there exist constants $\delta > 0$, $s < 1$ such that no algorithm can take a Label Cover *instance $\mathcal{L}$ and can decide whether $\mathsf{OPT}(\mathcal{L}) = 1$ or $\mathsf{OPT}(\mathcal{L}) < s$ in $O(2^{\delta|U|})$ time.*

Let ℓ be a parameter that will be related to k in Max k-Coverage later. We can ensure ℓ divides $|U|$ by taking an arbitrary vertex in U and making $\ell\lceil|U|/\ell\rceil - |U|$ copies of it. This does not change any of the properties in Theorem 18 except to increase the soundness s by $o_n(1)$; however, the soundness still remains bounded away from 1.

Since we want few vertices on the left, we construct a new Label Cover instance $\mathcal{L}_1$ by partitioning U into ℓ groups and creating super-vertices for each one. Formally, index the vertices of u as $U = \{u_{i,j}\}_{i\in[\ell],j\in[m/\ell]}$, and let the i^{th} part be $S_i := \{u_{i,j}\}_{j\in[m/\ell]}$. The new instance $\mathcal{L}_1 = ((U_1 \cup V_1, E_1), \Sigma_{U_1}, \Sigma_{V_1}, \{\pi^1_e\}_{e\in E_1})$ is constructed as follows.

- $V_1 = V$ and $\Sigma_{V_1} = \Sigma_V$ (the RHS remains unchanged),
- $U_1 = \{S_1, \ldots, S_\ell\}$. $\Sigma_{U_1} = (\Sigma_U)^{m/\ell}$ (the LHS has one super-vertex for each group), and
- for each $e = (u, v) \in E$ such that $u = u_{i,j}$, add an edge $e' = (S_i, v)$ to E_1 with the projection $\pi^1_{e'}(\sigma_1, \ldots, \sigma_{m/\ell}) := \pi_e(\sigma_j)$ where the latter π_e denotes the projection in $\mathcal{L}$. (Recall we allow parallel edges with different projections.)

Since the set of possible labelings and the set of edges remain the same except for syntactic changes, the completeness c and the soundness s do not change. The parameters become $|U_1| = \ell, |V_1| = |V|, |\Sigma_{U_1}| = 2^{O(m/\ell)}, |\Sigma_{V_1}| = O(1)$. It still maintains U-regularity and $d_{V_1} = d_V \leq m$.

The final transformation is the powerful *parallel repetition* step, which shows that the soundness decreases exponentially as we take the natural graph power. Fix $r \in \mathbb{N}$. The instance $\mathcal{L}_2 = ((U_2 \cup V_2, E_2), \Sigma_{U_2}, \Sigma_{V_2}, \{\pi^2_e\}_{e\in E_2})$ is constructed as follows.

- $U_2 = (U_1)^r$ and $\Sigma_{U_2} = (\Sigma_{U_1})^r$.
- $V_2 = (V_1)^r$ and $\Sigma_{V_2} = (\Sigma_{V_1})^r$.
- $E_2 = (E_1)^r$. For each $e = (e_i)_{i\in[r]} \in E_2$ with $e_i = (u_i, v_i) \in E_1$ and $(\sigma_1, \ldots, \sigma_r) \in \Sigma^r_{U_1}$, $\pi^2_e(\sigma_1, \ldots, \sigma_r) = (\pi^1_{e_1}(\sigma_1), \ldots, \pi^1_{e_r}(\sigma_r))$.

The parameters become $|U_2| = \ell^r, |V_2| = |V|^r = n^r, |\Sigma_{U_2}| = 2^{O(mr/\ell)}, |\Sigma_{V_2}| = 2^{O(r)}, d_{V_2} \leq d^r_{V_1} \leq m^r$, and $\mathcal{L}_2$ maintains U-regularity. The completeness c still remains 1, and by the parallel repetition theorem [25], the soundness drops $s = 2^{-\Theta(r)}$, where the constant hiding in the Θ depends on the original soundness. This proves Lemma 17.

3.2 Hardness of Max k-Coverage from Label Cover

Given the "nice" Label Cover instance from Lemma 17 we now show how to reduce this to Max k-Coverage. This reduction is standard and closely follows the classical one given by Feige [13], modulo some minor issues arising from it not being V-regular.

Recall that an instance of $\mathcal{I}$ of Max k-Coverage consists of an underlying universe $\mathcal{U}$, a family $\mathcal{S}$ of subsets, and an integer k. The goal is to find a subfamily $\mathcal{S}' \subseteq \mathcal{S}$ with $|\mathcal{S}'| = k$ that covers the largest number of elements. For notational simplicity, we prove the hardness

of the *weighted version* of Max k-Coverage where each element $e \in \mathcal{U}$ has weight $w(e)$ and we want to maximize the total weight of the covered elements. Note that weighted instances can be easily converted to unweighted instances by duplicating elements according to their weights. In our reduction, the ratio between the maximum and the minimum weight will be bounded by the number of elements. The proof appears in the full version of the paper.

▶ **Lemma 19** (Reduction #2). *There exist functions $a : \mathbb{R}^+ \to \mathbb{N}$ and $f : \mathbb{R}^+ \to \mathbb{R}^+$ such that for any $\varepsilon > 0$, there exists a polynomial-time reduction that takes a* Label Cover *instance $\mathcal{L} = ((U \cup V, E), \Sigma_U, \Sigma_V, \{\pi_e\}_{e \in E})$ that is U-regular and has the maximum V-degree d_V, and produces a* Max k-Coverage *instance $\mathcal{I} = (\mathcal{U}, \mathcal{S}, k)$ such that*

- *(Completeness)* $\mathsf{OPT}(\mathcal{L}) = 1 \implies \mathsf{OPT}(\mathcal{I}) = w(\mathcal{U})$.
- *(Soundness)* $\mathsf{OPT}(\mathcal{L}) < f(\varepsilon) \implies \mathsf{OPT}(\mathcal{I}) \le (1 - 1/e + \varepsilon) \cdot w(\mathcal{U})$.

The reduction satisfies $|\mathcal{U}| \le |V| \cdot |d_V|^{a(\varepsilon)} \cdot a(\varepsilon)^{\Sigma_V}$, $|\mathcal{S}| = a(\varepsilon) \cdot |U| \cdot |\Sigma_U|$, and $k = a|U|$.

We can now finish the proof of Theorem 15 based on Lemma 17 and Lemma 19.

Proof of Theorem 15. Fix $\varepsilon > 0$ that determines $a(\varepsilon)$ and $f(\varepsilon)$ in Lemma 19. Let $r \in \mathbb{N}$ in Lemma 17 so that the soundness $2^{-\Omega(r)} \le f(\varepsilon)$.

With ℓ still being a free parameter, Lemma 17 shows a reduction from an initial 3-SAT instance ϕ with n variables and $m = O(n)$ clauses to a Label Cover instance with $|U| = \ell^r, |V| = n^r, |\Sigma_U| = 2^{O(mr/\ell)}, |\Sigma_V| = 2^{O(r)}$, and $d_V \le m^r$. Lemma 19 with this Label Cover instance produces a Max k-Coverage instance with

$$\begin{aligned} |\mathcal{U}| &\le |V| \cdot |d_V|^a \cdot a^{\Sigma_V} = n^{O(ar)} \cdot a^{2^{O(r)}} \\ |\mathcal{S}| &= a \cdot |U| \cdot |\Sigma_U| = a\ell^r \cdot 2^{O(nr/\ell)} \\ k &= a|U| = a\ell^r. \end{aligned}$$

An $(1 - 1/e + \varepsilon)$-approximation algorithm for Max k-Coverage that runs in time $|\mathcal{S}|^{k^{1/2r}}$ will distinguish whether $\mathsf{OPT}(\phi) = 1$ or $\mathsf{OPT}(\phi) < s'$ for some s' in time

$$2^{O((nr/\ell) \cdot k^{1/2r})} = 2^{O((nr/\ell) \cdot \sqrt{\ell})} = 2^{O(nr/\sqrt{\ell})},$$

which will contradict the Gap-ETH for large enough ℓ. Observe that $|\mathcal{S}| \gg |\mathcal{U}|$; if we set $g(\varepsilon) := 1/2r$, we get the same implication from an algorithm that runs in time $|\mathcal{U}|^{k^{g(\varepsilon)}}$, which proves the theorem. ◀

References

1. Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better Guarantees for k-Means and Euclidean k-Median by Primal-Dual Algorithms. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 61–72, 2017.
2. Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The Hardness of Approximation of Euclidean k-Means. In *31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands*, pages 754–767, 2015.
3. Jarosław Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median, and positive correlation in budgeted optimization. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 737–756. SIAM, 2014.

4 Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.*, 40(6):1740–1766, 2011. doi:10.1137/080733991.

5 Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. From gap-ETH to FPT-inapproximability: Clique, dominating set, and more. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 743–754. IEEE, 2017.

6 Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A Constant-Factor Approximation Algorithm for the k-Median Problem. *J. Comput. Syst. Sci.*, 65(1):129–149, 2002. doi:10.1006/jcss.2002.1882.

7 Ke Chen. On k-Median clustering in high dimensions. In *SODA*, 2006.

8 Ke Chen. On coresets for k-median and k-means clustering in metric and Euclidean spaces and their applications. *SIAM Journal on Computing*, 39(3):923–947, 2009.

9 Yijia Chen and Bingkai Lin. The constant inapproximability of the parameterized dominating set problem. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 505–514. IEEE, 2016.

10 Karthik C.S., Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1283–1296. ACM, 2018.

11 Artur Czumaj and Christian Sohler. Small Space Representations for Metric Min-sum k-Clustering and Their Applications. *Theory Comput. Syst.*, 46(3):416–442, 2010. doi:10.1007/s00224-009-9235-1.

12 Irit Dinur. Mildly exponential reduction from gap 3SAT to polynomial-gap label cover. *Electronic Colloquium on Computational Complexity (ECCC)*, pages TR 16–128, 2016.

13 Uriel Feige. A Threshold of Ln N for Approximating Set Cover. *J. ACM*, 45(4), July 1998.

14 Dan Feldman and Michael Langberg. A Unified Framework for Approximating and Clustering Data. *CoRR*, abs/1106.1379, 2011. (An extended abstract appeared at STOC 11). arXiv:1106.1379.

15 Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, PCA and projective clustering. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1434–1453. Society for Industrial and Applied Mathematics, 2013.

16 Sudipto Guha and Samir Khuller. Greedy strikes back: improved facility location algorithms. *J. Algorithms*, 31(1):228–248, 1999. doi:10.1006/jagm.1998.0993.

17 Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. *Comput. Geom.*, 28(2-3):89–112, 2004. doi:10.1016/j.comgeo.2004.03.003.

18 Ravishankar Krishnaswamy, Amit Kumar, Viswanath Nagarajan, Yogish Sabharwal, and Barna Saha. Facility Location with Matroid or Knapsack Constraints. *Math. Oper. Res.*, 40(2):446–459, 2015. doi:10.1287/moor.2014.0678.

19 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear-time approximation schemes for clustering problems in any dimensions. *J. ACM*, 57(2):5:1–5:32, 2010. doi:10.1145/1667053.1667054.

20 Euiwoong Lee, Melanie Schmidt, and John Wright. Improved and simplified inapproximability for k-means. *Inf. Process. Lett.*, 120:40–43, 2017. doi:10.1016/j.ipl.2016.11.009.

21 Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45–58, 2013.

22 Shi Li and Ola Svensson. Approximating k-Median via Pseudo-Approximation. *SIAM J. Comput.*, 45(2):530–547, 2016. doi:10.1137/130938645.

23 Bingkai Lin. The parameterized complexity of k-biclique. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 605–615. SIAM, 2015.

24 Pasin Manurangsi and Prasad Raghavendra. A Birthday Repetition Theorem and Complexity of Approximating Dense CSPs. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 78:1–78:15, 2017.

25 Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998.

26 Chaitanya Swamy. Improved Approximation Algorithms for Matroid and Knapsack Median Problems and Applications. *ACM Trans. Algorithms*, 12(4):49:1–49:22, 2016. doi:10.1145/2963170.

Information-Theoretic and Algorithmic Thresholds for Group Testing

Amin Coja-Oghlan
Goethe University, Frankfurt, Germany
acoghlan@math.uni-frankfurt.de

Oliver Gebhard
Goethe University, Frankfurt, Germany
gebhard@math.uni-frankfurt.de

Max Hahn-Klimroth
Goethe University, Frankfurt, Germany
hahnklim@math.uni-frankfurt.de

Philipp Loick
Goethe University, Frankfurt, Germany
loick@math.uni-frankfurt.de

Abstract

In the group testing problem we aim to identify a small number of infected individuals within a large population. We avail ourselves to a procedure that can test a group of multiple individuals, with the test result coming out positive iff at least one individual in the group is infected. With all tests conducted in parallel, what is the least number of tests required to identify the status of all individuals? In a recent test design [Aldridge et al. 2016] the individuals are assigned to test groups randomly, with every individual joining an equal number of groups. We pinpoint the sharp threshold for the number of tests required in this randomised design so that it is information-theoretically possible to infer the infection status of every individual. Moreover, we analyse two efficient inference algorithms. These results settle conjectures from [Aldridge et al. 2014, Johnson et al. 2019].

2012 ACM Subject Classification Theory of computation → Theory and algorithms for application domains; Theory of computation → Bayesian analysis; Theory of computation → Machine learning theory

Keywords and phrases Group testing problem, phase transitions, information theory, efficient algorithms, sharp threshold, Bayesian inference

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.43

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/pdf/1902.02202.pdf`.

Funding *Amin Coja-Oghlan*: Supported of DFG 646/3.
Max Hahn-Klimroth: Supported by Stiftung Polytechnische Gesellschaft.
Philipp Loick: Supported of DFG 646/3.

Acknowledgements We thank Arya Mazumdar for bringing the group testing problem to our attention.

1 Introduction

1.1 Background and motivation

The group testing problem goes back to the work of Dorfman from the 1940s [19]. Among a large population a few individuals are infected with a rare disease. The objective is to identify the infected individuals effectively. At our disposal we have a testing procedure capable of

not merely testing one individual, but several. The test result will be positive if any one individual in the test group is infected, and negative otherwise; all tests are conducted in parallel. We are at liberty to assign a single individual to several test groups. The aim is to devise a test design that identifies the status of every single individual correctly while requiring as small a number of tests as possible.

A recently proposed test design allocates the individuals to tests randomly [7, 10, 11, 28, 32]. To be precise, given integers $n, m, \Delta > 0$ we create a random bipartite multi-graph by choosing independently for each of the n vertices $x_1, \ldots, x_n$ "on the left" Δ neighbours among the m vertices $a_1, \ldots, a_m$ "on the right" uniformly at random with replacement. The vertices $x_1, \ldots, x_n$ represent the individuals, the $a_1, \ldots, a_m$ represent the test groups and an individual joins a test group iff the corresponding vertices are adjacent. The wisdom behind this construction is that the expansion properties of the random bipartite graph precipitate virtuous correlations, facilitating inference.

Given n and (an estimate of) the number k of infected individuals, what is the least m for which, with a suitable choice of Δ, the status of every individual can be inferred correctly from the test results with high probability? Like in many other inference problems the answer comes in two instalments. First, we might ask for what m it is *information-theoretically* possible to detect the infected individuals. In other words, regardless of computational resources, do the test results contain enough information in principle to identify the infection status of every individual? Second, for what m does this problem admit *efficient algorithms*?

The first main result of this paper resolves the information-theoretic question completely. Specifically, Aldridge, Johnson and Scarlett [11] obtained a function $m_{\inf} = m_{\inf}(n, k)$ such that for any fixed $\varepsilon > 0$ the inference problem is information-theoretically infeasible if $m < (1-\varepsilon)m_{\inf}$. They conjectured that this bound is tight, i.e., that for $m > (1+\varepsilon)m_{\inf}(n, k)$ there is an (exponential) algorithm that correctly identifies the infected individuals with high probability. We prove this conjecture.

Furthermore, concerning the algorithmic question, Johnson, Aldridge and Scarlett [28] obtained a function $m_{\mathrm{alg}} = m_{\mathrm{alg}}(n, k)$ that exceeds $m_{\inf}$ by a modest constant factor such that for $m > (1 + \varepsilon)m_{\mathrm{alg}}$ certain efficient algorithms successfully identify the infected individuals with high probability. They conjectured that `SCOMP`, their most sophisticated algorithm, actually succeeds for smaller values of m. We refute this conjecture and show that `SCOMP` fails to outperform a much simpler algorithm called `DD`.

A technical novelty of the present work is that we investigate the group testing problem from a new perspective. While most prior contributions rely either on elementary calculations and/or information-theoretic arguments [10, 11, 28, 38, 39], here we bring to bear techniques from the theory of random constraint satisfaction problems [5, 31]. Indeed, group testing can be viewed naturally as a constraint satisfaction problem: the tests provide the constraints and the task is to find all possible ways of assigning a status ("infected" or "not infected") to the n individuals in a way consistent with the given test results. Since the allocation of individuals to tests is random, this question is similar in nature to, e.g., the random k-SAT problem that asks for a Boolean assignment that satisfies a random collection of clauses [4, 6, 16, 18]. Apart from obtaining the aforementioned new results, this novel perspective allows for short proofs of results that were established more laboriously in prior work. It also puts the group testing problem in the same framework as the considerable body of recent work on other inference problems on random graphs such as the stochastic block model (e.g., [1, 15, 17, 34, 35, 41]).

We proceed to state the main results of the paper precisely, followed by a detailed discussion of the prior literature on group testing. An outline of the proof strategy follows in Section 2.

1.2 The information-theoretic threshold

Throughout the paper we labour under the assumptions commonly made in the context of group testing; we will revisit their merit in Section 1.4. Specifically, we assume that the number k of infected individuals satisfies $k \sim n^\theta$ for a fixed $0 < \theta < 1$. Moreover, let $\boldsymbol{\sigma} \in \{0,1\}^{\{x_1,\ldots,x_n\}}$ be a vector of Hamming weight k chosen uniformly at random. The (one-)entries of $\boldsymbol{\sigma}$ indicate which of the n individuals are infected. Moreover, let $\boldsymbol{G} = \boldsymbol{G}(n, m, \Delta)$ signify the aforementioned random bipartite graph. Then $\boldsymbol{\sigma}$ induces a vector $\hat{\boldsymbol{\sigma}} \in \{0,1\}^{\{a_1,\ldots,a_m\}}$ that indicates which of the m tests come out positive. To be precise, $\hat{\boldsymbol{\sigma}}_i = 1$ iff test a_i is adjacent to an individual x_j with $\boldsymbol{\sigma}_{x_j} = 1$. For what m is it possible to recover $\boldsymbol{\sigma}$ from $\boldsymbol{G}, \hat{\boldsymbol{\sigma}}$? Here, we settle an important open question [28] on the sharpness of the information-theoretic threshold. (Throughout the paper all logarithms are base e.)

▶ **Theorem 1.** *Suppose that $0 < \theta < 1$, and $\varepsilon > 0$ and let*

$$m_{\inf} = m_{\inf}(n,\theta) = \frac{n^\theta(1-\theta)\log(n)}{\min\left\{1, \frac{1-\theta}{\theta}\log 2\right\}\log 2}.$$

- **(i)** *If $m < (1-\varepsilon)m_{\inf}(n,\theta)$, then there does not exist any algorithm that given $\boldsymbol{G}, \hat{\boldsymbol{\sigma}}, k$ outputs $\boldsymbol{\sigma}$ with a non-vanishing probability.*
- **(ii)** *If $m > (1+\varepsilon)m_{\inf}(n,\theta)$, then there exists an algorithm that given $\boldsymbol{G}, \hat{\boldsymbol{\sigma}}$ outputs $\boldsymbol{\sigma}$ with high probability.*

Since for $\theta \le \log(2)/(1+\log(2))$ the first part of Theorem 1 readily follows from a folklore argument [20], the interesting regime is $\theta > \log(2)/(1+\log(2)) \approx 0.41$. In this regime Theorem 1 strengthens a result from [11], who showed that for $m < (1-\varepsilon)m_{\inf}$ any inference algorithm has a strictly positive error probability. By comparison, Theorem 1 shows that any algorithm fails with *high* probability.

But the main contribution of Theorem 1 is the second, positive statement. While the case $\theta > 1/2$ is easy because a plain greedy algorithms succeeds [28], the case $\theta < 1/2$ proved more challenging and was so far only heuristically justified using techniques from statistical physics [32] and for $\theta < 1/3$ for a different test design [38, 39]. Indeed, Aldridge et al. [10] conjectured that in this case inferring $\boldsymbol{\sigma}$ from $\boldsymbol{G}, \hat{\boldsymbol{\sigma}}$ is equivalent to solving a hypergraph minimum vertex cover problem. The proof of Theorem 1 vindicates this conjecture. Specifically, the vertex set of the hypergraph comprises all "potentially infected" individuals, i.e., those that do not appear in any negative test. The hyperedges are the neighbourhoods ∂a_i of the positive tests a_i in $\boldsymbol{G}$. Exhaustive search solves this vertex cover problem in time $\exp(O(n^\theta \log n))$. But how about efficient algorithms for general θ?

1.3 Efficient algorithms for group testing

Several polynomial time group testing algorithms have been proposed. A very simple greedy strategy called `DD` (for "definitive defectives") first labels all individuals that are members of negative test groups as healthy. Subsequently it checks for positive tests in which all individuals but one have been identified as healthy in the first step. Clearly, the single as yet unlabelled individual in such a test group must be infected. Up to this point all decisions made by `DD` are correct. But in the final step `DD` marks all as yet unclassified individuals as healthy, possibly causing false negatives. In fact, the output of `DD` may be inconsistent with the test results as possibly some positive tests may fail to spot an individual classified as "infected".

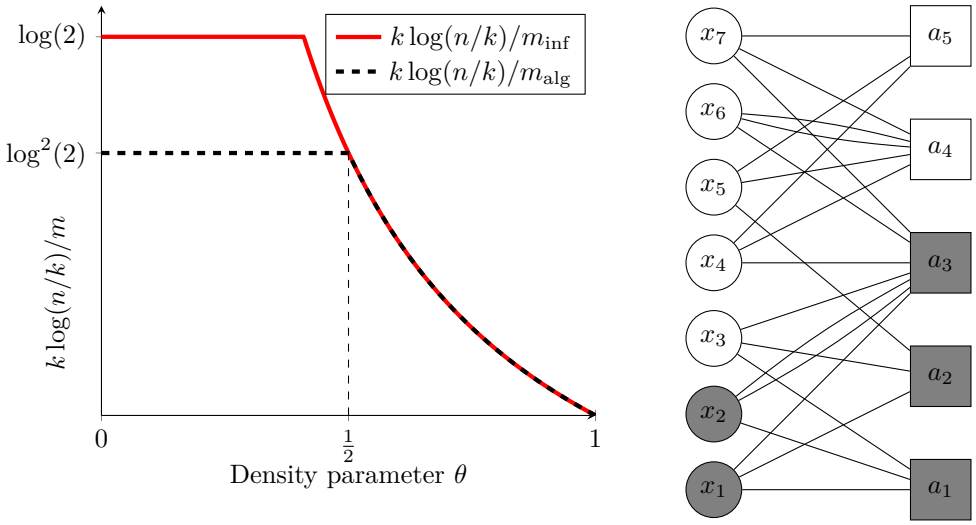


Figure 1 The left diagram displays $m_{\mathrm{inf}}, m_{\mathrm{alg}}$. The red line shows the information theoretic threshold m_{inf}, the dashed black line signifies the bound m_{alg} which is achieved by the both the SCOMP and the DD algorithm. The graph on the right illustrates a small example of a group testing instance, with the individuals $x_1, \dots, x_7$ on the left and the tests $a_1, \dots, a_5$ on the right. Infected individuals and positive tests are coloured in grey.

The more sophisticated SCOMP algorithm is roughly equivalent to the well-known greedy algorithm for the hypergraph vertex cover problem applied to the hypergraph from the previous paragraph. Specifically, in its first step SCOMP proceeds just like DD, classifying all individuals that occur in negative tests as healthy. Then SCOMP identifies as infected all unmarked individuals that appear in at least one test whose other participants are already known to be healthy. Subsequently the algorithm keeps picking an individual that appears in the largest number of as yet "unexplained" (viz. uncovered) positive tests and marks that individual as infected, with ties broken randomly, until every positive test contains an individual classified as infected. Clearly, SCOMP may produce false positives as well as false negatives. But at least the output is consistent with the test results.

Analysing SCOMP has been prominently posed as an open problem in the group testing literature [8, 10, 28]. Indeed, Aldridge et al. [10] opined that "the complicated sequential nature of SCOMP makes it difficult to analyse mathematically". On the positive side, [10] proved that SCOMP succeeds in recovering $\boldsymbol{\sigma}$ correctly given $(\boldsymbol{G}, \hat{\boldsymbol{\sigma}})$ if $m > (1+\varepsilon)m_{\mathrm{alg}}(n,\theta)$ w.h.p., where

$$m_{\mathrm{alg}} = m_{\mathrm{alg}}(n,\theta) = \frac{n^{\theta}(1-\theta)\log(n)}{\min\left\{1, \frac{1-\theta}{\theta}\right\}\log^2 2}. \tag{1.1}$$

However, the algorithm succeeds for a trivial reason; namely, for $m > (1+\varepsilon)m_{\mathrm{alg}}$ even DD suffices to recover $\boldsymbol{\sigma}$ w.h.p. Yet based on experimental evidence [10, 28] conjectured that SCOMP strictly outperforms DD. The following theorem refutes this conjecture.

▶ **Theorem 2.** *Suppose that $0 < \theta < 1$ and $\varepsilon > 0$. If $m < (1-\varepsilon)m_{\mathrm{alg}}(n,\theta)$, then given $\boldsymbol{G}, \hat{\boldsymbol{\sigma}}$ w.h.p. both SCOMP and DD fail to output $\boldsymbol{\sigma}$.*

For $\theta < 1/2$ the information-theoretic bound provided by Theorem 1 and the algorithmic bound m_{alg} supplied by Theorem 2 remain a modest constant factor apart; see Figure 1. In some other inference problems on random graphs such as the stochastic block model similar

gaps appear between the information-theoretic and the algorithmic bounds [1, 17, 34, 41]. There have been attempts at investigating to what extent these gaps are due to genuine computational barriers, i.e., [23, 24, 25, 26]. Whether there actually exists a computationally hard regime for group testing, or whether the gap can be closed by smarter algorithms, remains an exciting question for future research.

1.4 Discussion and related work

Dorfman's original group testing scheme, intended to test the American army for syphilis, was *adaptive*. In a first round of tests each soldier would be allocated to precisely one test group. If the test result came out negative, none of the soldiers in the group were infected. In a second round the soldiers whose group was tested positively would then be tested individually. Of course, Dorfman's scheme was not information-theoretically optimal. An optimal adaptive scheme that involves several test stages, with the tests conducted in the present stage governed by the results from the previous stages, is known [20, 12]. In the adaptive scenario the information-theoretic threshold works out to be

$$m_{\mathrm{inf}}^{\mathrm{adapt}}(n,k) = \frac{n^\theta(1-\theta)\log(n)}{\log 2}.$$

The lower bound, i.e., that no adaptive design gets by with $(1-\varepsilon)m_{\mathrm{inf}}^{\mathrm{adapt}}(n,k)$ tests, follows from a very simple information-theoretic consideration. Namely, with a total of m tests at our disposal there are merely 2^m possible test outcomes, and we need this number to exceed the count $\binom{n}{k}$ of possible vectors $\boldsymbol{\sigma}$.

More recently there has been a great deal of interest in non-adaptive group testing, where the infection status of each individual is to be determined after just one round of tests [7, 9, 10, 11, 14, 22, 28, 32, 38, 39]. This is the version of the problem that we deal with in the present paper. An important advantage of the non-adaptive scenario is that tests, which may be time-consuming, can be conducted in parallel. Indeed, some of today's most popular applications of group testing are non-adaptive such as DNA screening [14, 30, 37] or protein interaction experiments [36, 40] in computational molecular biology. The randomised test design that we deal with here is the best currently known non-adaptive design (in terms of the number of tests required).

The most interesting regime for the group testing problem is when the number k of infected individuals scales as a power n^θ of the entire population. Mathematically this is because in the linear regime $k = \Omega(n)$ the optimal strategy is to perform n individual tests [9]. Thus, for k linear in n there is nothing interesting to do. But the sublinear case is also of practical relevance, as witnessed by Heap's law in epidemiology [13] or biological applications [22].

Apart from the randomised test design $\boldsymbol{G}$ where each individual chooses precisely Δ tests (with replacement), the so-called Bernoulli design assigns each individual to every test with a certain probability independently. A considerable amount of attention has been devoted to this model, and its information-theoretic threshold as well as the thresholds for various algorithms have been determined [8, 7, 10, 38, 39]. However, the Bernoulli test design, while easier to analyse, is provably inferior to the test design $\boldsymbol{G}$ that we study here. This is because in the Bernoulli design there are likely quite a few individuals that participate in far fewer tests than expected due to random degree fluctuations.

1.5 Notation

Throughout the paper $\boldsymbol{G} = \boldsymbol{G}(n, m, \Delta)$ denotes the random bipartite graph that describes which individuals take part in which test groups, the vector $\boldsymbol{\sigma} \in \{0,1\}^{\{x_1,\dots,x_n\}}$ encodes which individuals are infected, and $\hat{\boldsymbol{\sigma}} \in \{0,1\}^{\{a_1,\dots,a_m\}}$ indicates the test results. Moreover, $k \sim n^\theta$ signifies the number of infected individuals. Additionally, we write $V = V_n = \{x_1, \dots, x_n\}$ for the set of all individuals and $V_0 = \{x_i \in V : \boldsymbol{\sigma}_{x_i} = 0\}$, $V_1 = V \setminus V_0$ for the set of healthy and infected individuals, respectively. For an individual $x \in V$ we write ∂x for the set of tests a_i adjacent to x. Analogously, for a test a_i we denote by ∂a_i the set of individuals that take part in the test. Finally, all asymptotic notation refers to the limit $n \to \infty$. Thus, $o(1)$ denotes a term that vanishes in the limit of large n, while $\omega(1)$ stands for function that diverges to ∞ as $n \to \infty$.

2 Outline

We give an overview of the main arguments upon which the proofs of Theorems 1 and 2 rest.

2.1 The Nishimori identity

The very first item on the agenda is to get a handle on the posterior distribution of $\boldsymbol{\sigma}$ given $\boldsymbol{G}$ and $\hat{\boldsymbol{\sigma}}$. To this end, let $S_k(\boldsymbol{G}, \hat{\boldsymbol{\sigma}})$ be the set of all vectors $\sigma \in \{0,1\}^V$ of Hamming weight k such that

$$\hat{\boldsymbol{\sigma}}_{a_i} = \mathbf{1}\{\exists x \in \partial a_i : \sigma_x = 1\} \qquad \text{for all } i \in [m].$$

In words, $S_k(\boldsymbol{G}, \hat{\boldsymbol{\sigma}})$ contains the set of all vectors σ with k ones that label the individuals infected/healthy in a way consistent with the test results. Let $Z_k(\boldsymbol{G}, \hat{\boldsymbol{\sigma}}) = |S_k(\boldsymbol{G}, \hat{\boldsymbol{\sigma}})|$. The following proposition shows that the posterior of $\boldsymbol{\sigma}$ given $\boldsymbol{G}, \hat{\boldsymbol{\sigma}}$ is uniform on $S_k(\boldsymbol{G}, \hat{\boldsymbol{\sigma}})$.

▶ **Proposition 3** ([7]).
For all $\tau \in \{0,1\}^{\{x_1,\dots,x_n\}}$ *we have* $\mathbb{P}[\boldsymbol{\sigma} = \tau \mid \boldsymbol{G}, \hat{\boldsymbol{\sigma}}] = \dfrac{\mathbf{1}\{\tau \in S_k(\boldsymbol{G}, \hat{\boldsymbol{\sigma}})\}}{Z_k(\boldsymbol{G}, \hat{\boldsymbol{\sigma}})}$.

Adopting the jargon of the recent literature on inference problems on random graphs, we refer to Proposition 3 as the *Nishimori identity* [15, 41]. The proposition shows that apart from the actual test results, there is no further "hidden information" about $\boldsymbol{\sigma}$ encoded in $\boldsymbol{G}, \hat{\boldsymbol{\sigma}}$. In particular, the information-theoretically optimal inference algorithm just outputs a uniform sample from $S_k(\boldsymbol{G}, \hat{\boldsymbol{\sigma}})$. In effect, we obtain the following.

▶ **Corollary 4.**
1. *If* $Z_k(\boldsymbol{G}, \hat{\boldsymbol{\sigma}}) = \omega(1)$ *w.h.p., then for any algorithm* $\mathcal{A}$ *we have*

$$\mathbb{P}[\mathcal{A}(\boldsymbol{G}, \hat{\boldsymbol{\sigma}}, k) = \boldsymbol{\sigma}] = o(1).$$

2. *If* $Z_k(\boldsymbol{G}, \hat{\boldsymbol{\sigma}}) = 1$ *w.h.p., then there is an algorithm* $\mathcal{A}$ *such that*

$$\mathbb{P}[\mathcal{A}(\boldsymbol{G}, \hat{\boldsymbol{\sigma}}, k) = \boldsymbol{\sigma}] = 1 - o(1).$$

Both the positive and the negative part of Corollary 4 assume that the precise number k of infected individuals is known to the algorithm. This assumption makes the negative part stronger, but weakens the positive part. Yet we will see in due course how in the positive scenario the assumption that k be known can be removed. The upshot is that we need to get a handle on $Z_k(\boldsymbol{G}, \hat{\boldsymbol{\sigma}})$.

2.2 The information-theoretic threshold

We proceed to discuss the proof of Theorem 1. The proofs of the first, negative statement and of the second, positive statement hinge on two separate arguments. We begin with the negative statement that w.h.p. $\boldsymbol{\sigma}$ cannot be inferred if $m < (1-\varepsilon)m_{\inf}$.

2.2.1 The information-theoretic lower bound

In light of Corollary 4 in order to prove the first part of Theorem 1 we need to show that the number $Z_k(\boldsymbol{G}, \hat{\boldsymbol{\sigma}})$ of assignments consistent with the test results $\hat{\boldsymbol{\sigma}}$ is unbounded w.h.p. The proof of this fact is based on a very simple idea: we just identify a biggish number of individuals whose infection status could be flipped without affecting the test results. To be precise, let $V_0^+ = V_0^+(\boldsymbol{G}, \hat{\boldsymbol{\sigma}})$ be the set of all healthy individuals x_i such that every test in which x_i occurs is positive; in symbols,

$$V_0^+ = \{x_i \in V_0 : \forall a \in \partial x_i \exists y \in \partial a : \boldsymbol{\sigma}_y = 1\}. \tag{2.1}$$

Similarly, let V_1^+ be the set of all infected individuals x_i such that every test in which x_i occurs features another infected individual; in symbols,

$$V_1^+ = \{x_i \in V_1 : \forall a \in \partial x_i \exists y \in \partial a \setminus \{x_i\} : \boldsymbol{\sigma}_y = 1\}.$$

We think of the individuals in V_0^+ as the "potential false positives". Indeed, if for any $x_i \in V_0^+$ we obtain $\boldsymbol{\sigma}'$ from $\boldsymbol{\sigma}$ by setting x_i to one, then $\boldsymbol{\sigma}'$ will render the same test results as $\boldsymbol{\sigma}$. Similarly, the individuals in V_1^+ are potential false negatives.

The following lemma yields a bound on m below which potential false positives and negatives abound. A simple (omitted) calculation also yields the value of Δ that is optimal to facilitate inference, namely $\Delta = \lceil \frac{m}{k} \log 2 \rceil$.

▶ **Lemma 5.** *Let $\varepsilon > 0$ and $0 < \theta < 1$ and assume that*

$$m < \frac{(1-\varepsilon)\theta}{(1-\theta)\log^2 2} n^\theta (1-\theta) \log n$$

Then even with the optimal choice $\Delta = \lceil \frac{m}{k} \log 2 \rceil$ we have $|V_0^+|, |V_1^+| = n^{\Omega(1)}$ w.h.p.

The proof of Lemma 5 relies on a basic random graphs argument. As an immediate application we obtain the following information-theoretic lower bound.

▶ **Corollary 6.** *Let $\varepsilon > 0$ and $0 < \theta < 1$ and assume that*

$$m < \frac{(1-\varepsilon)\theta}{(1-\theta)\log^2 2} n^\theta (1-\theta) \log n \tag{2.2}$$

Then $Z_k(\boldsymbol{G}, \hat{\boldsymbol{\sigma}}) = \omega(1)$ w.h.p.

Proof. We need to exhibit alternative vectors $\boldsymbol{\sigma}' \in \{0,1\}^V$ with Hamming weight k that render the same test results as $\boldsymbol{\sigma}$. Thus, pick any $x_i \in V_0^+$ and any $x_j \in V_1^+$ and obtain $\boldsymbol{\sigma}'$ from $\boldsymbol{\sigma}$ by setting $\boldsymbol{\sigma}'_{x_i} = 1$ and $\boldsymbol{\sigma}'_{x_j} = 0$. By construction, $\boldsymbol{\sigma}'$ has Hamming weight k and renders the same test results. Hence, Lemma 5 shows that $Z_k(\boldsymbol{G}, \hat{\boldsymbol{\sigma}}) \geq |V_0^+ \times V_1^+| = \Omega(n^{2\theta}) \gg 1$ w.h.p. ◀

The bound (2.2) matches $m_{\inf}$ for $\theta \gtrapprox 0.41$. A simpler, purely information-theoretic argument covers the remaining θ.

▶ **Lemma 7.** *Let $\varepsilon > 0$, $0 < \theta < 1$. If $m < \frac{1-\varepsilon}{\log 2} n^\theta (1-\theta) \log n$, then $Z_k(\boldsymbol{G}, \hat{\boldsymbol{\sigma}}) = \omega(1)$ w.h.p.*

We thus conclude that for all $0 < \theta < 1$, w.h.p. $Z_k(\boldsymbol{G}, \hat{\boldsymbol{\sigma}}) = \omega(1)$ if $m < (1-\varepsilon)m_{\inf}$. Therefore, the desired information-theoretic lower bound follows from Corollary 4.

2.2.2 The information-theoretic upper bound

The proof of the information-theoretic upper bound is the principal achievement of the present work. The proof rests upon techniques that have come to play an important role in the theory of random constraint satisfaction problems. Specifically, we need to show that $Z_k(\boldsymbol{G}, \hat{\boldsymbol{\sigma}}) = 1$ w.h.p., i.e., that $\boldsymbol{\sigma}$ is the only assignment compatible with the test results w.h.p. We establish this result by combining two separate arguments. First, we use a moment calculation to show that w.h.p. there are no other solutions that have a small "overlap" with $\boldsymbol{\sigma}$. Then we use an expansion argument to show that w.h.p. there are no alternative solutions with a big overlap. Both these arguments are variants of the arguments that have been used to study the solution space geometry of random constraint satisfaction problems such as random k-SAT or random k-XORSAT [3, 4, 21], as well as the freezing thresholds of random constraint satisfaction problems [2, 33]. Yet to our knowledge these methods have thus far not been applied to the group testing problem.

Formally, we define

$$Z_{k,\ell}(\boldsymbol{G}, \hat{\boldsymbol{\sigma}}) = |\{\sigma \in S_k(\boldsymbol{G}, \hat{\boldsymbol{\sigma}}) : \langle \boldsymbol{\sigma}, \sigma \rangle = \ell\}|$$

as the number of assignments $\sigma \in S_k(\boldsymbol{G}, \hat{\boldsymbol{\sigma}})$ whose *overlap* $\langle \boldsymbol{\sigma}, \sigma \rangle = \sum_{i=1}^{n} \mathbf{1}\{\sigma_{x_i} = \boldsymbol{\sigma}_{x_i} = 1\}$ with $\boldsymbol{\sigma}$ is equal to ℓ. The following two propositions rule out assignments with a small and a big overlap, respectively. In either case we choose $\Delta = \lceil \frac{m}{k} \log 2 \rceil$ to take its optimal value.

▶ **Proposition 8.** *Let $\varepsilon > 0$ and $0 < \theta < 1$ and assume that $m > (1+\varepsilon) m_{\inf}(k, \theta)$. W.h.p. we have $Z_{k,\ell}(\boldsymbol{G}, \hat{\boldsymbol{\sigma}}) = 0$ for all $\ell < (1 - 1/\log n)k$.*

▶ **Proposition 9.** *Let $\varepsilon > 0$ and $0 < \theta < 1$ and assume that $m > (1+\varepsilon) m_{\inf}(k, \theta)$. W.h.p. we have $Z_{k,\ell}(\boldsymbol{G}, \hat{\boldsymbol{\sigma}}) = 0$ for all $(1 - 1/\log n)k \leq \ell < k$.*

We defer the proofs of Propositions 8 and 9 to Sections 3 and 4, respectively.

Propositions 8 and 9 readily imply that $Z_k(\boldsymbol{G}, \hat{\boldsymbol{\sigma}}) = 1$ w.h.p. if $m > (1+\varepsilon) m_{\inf}(k, \theta)$. Hence, Corollary 4 shows that there exists an inference algorithm that given $\boldsymbol{G}, \hat{\boldsymbol{\sigma}}$ *and* k outputs $\boldsymbol{\sigma}$ w.h.p. However, up to now this algorithm relies on exactly knowing the number of infected individuals k, which in practice could be rather difficult to learn.

Fortunately this assumption can be removed. Namely, the following proposition shows that w.h.p. there is no assignment σ that is compatible with the test results and that has Hamming weight less than k.

▶ **Proposition 10.** *Let $\varepsilon > 0$ and $0 < \theta < 1$ and assume that $m > (1+\varepsilon) m_{\inf}(k, \theta)$. W.h.p. we have $\sum_{k' < k} Z_{k'}(\boldsymbol{G}, \hat{\boldsymbol{\sigma}}) = 0$.*

As an immediate consequence of Proposition 10 we conclude that for $m > (1+\varepsilon) m_{\inf}(k, \theta)$ the problem of inferring $\boldsymbol{\sigma}$ boils down to a minimum vertex cover problem, as previously conjectured by Aldridge, Baldassini and Johnson [10]. Namely, let $\mathcal{P}$ be the set of all positive tests, i.e., all tests a_i, $i \in [m]$, with $\hat{\boldsymbol{\sigma}}_{a_i} = 1$. Moreover, let V^+ be the set of all variables $x_i \in V$ such that $\partial x_i \subseteq \mathcal{P}$; in words, x_i takes part in positive tests only. We set up a hypergraph $\boldsymbol{H}$ with vertex set V^+ and hyperedges $\partial a_i \cap V^+$, $a_i \in \mathcal{P}$. Clearly, the set of all individuals x_i with $\boldsymbol{\sigma}_{x_i} = 1$ provides a valid vertex cover of $\boldsymbol{H}$ (as any positive test must feature an infected individual). Conversely, Propositions 8 and 9 show that w.h.p. this is the unique vertex cover of size k, and Proposition 10 shows that there is no strictly smaller vertex cover w.h.p. Therefore, w.h.p. we can infer $\boldsymbol{\sigma}$ even without prior knowledge of k by way of solving this minimum vertex cover instance.

2.3 The SCOMP algorithm

For $\theta \geq 1/2$ we have $m_{\mathrm{alg}} = m_{\mathrm{inf}}$ and thus Theorem 1 implies that SCOMP fails to infer $\boldsymbol{\sigma}$ w.h.p. for $m < (1-\varepsilon)m_{\mathrm{alg}}$. Therefore, we are left to establish Theorem 2 for $\theta < 1/2$, in which case

$$m_{\mathrm{alg}} = \frac{n^{\theta}(1-\theta)\log(n)}{\log^2 2}. \tag{2.3}$$

The proof of Theorem 2 for $\theta < 1/2$ hinges on two lemmas. First we show that below m_{alg}, the set V_1^{--} of infected individuals that the second step of SCOMP identifies correctly is empty. Formally, with V_0^+ from (2.1),

$$V_1^{--} = \left\{x \in V_1 : \exists a \in \partial x : \partial a \setminus \{x\} \subseteq V_0 \setminus V_0^+\right\}.$$

▶ **Lemma 11.** *Suppose that $0 < \theta < 1/2$ and $\varepsilon > 0$. If $m < (1-\varepsilon)m_{\mathrm{alg}}$, then for all $\Delta > 0$ we have $V_1^{--}(\boldsymbol{G}, \hat{\boldsymbol{\sigma}}^*) = \emptyset$ w.h.p.*

With the second step of SCOMP failing to 'explain' (viz. cover) any positive tests, the greedy vertex cover algorithm takes over. This algorithm is applied to the hypergraph whose vertices are the as yet unclassified individuals and whose edges are the neighbourhoods of the positive tests. Our second lemma shows that the set $V^{+,\Delta}$ of potententially false positive individuals $x \in V_0^+$ that participate in the maximum number Δ of different test is far greater than the actual number k of infected individuals. Formally, let

$$V_0^{+,\Delta} = \left\{x \in V_0^+ : |\partial x| = \Delta\right\}.$$

▶ **Lemma 12.** *Suppose that $0 < \theta < 1/2$ and $\varepsilon > 0$. If $m < (1-\varepsilon)m_{\mathrm{alg}}$, then for all $\Delta > 0$ we have $V_0^{+,\Delta} \geq k \log n$ w.h.p.*

The proofs of Lemmas 11 and 12 are based on moment calculations that turn out to be mildly subtle due to the potentially very large degrees of the underlying graph $\boldsymbol{G}$. We complete the proof of Theorem 2 as follows.

Proof of Theorem 2. The first step of SCOMP (correctly) marks all individuals that appear in negative tests as healthy. Moreover, Lemma 11 implies that the second step of SCOMP is void w.h.p., because there is no single infected individual that appears in a test whose other individuals have already been identified as healthy by the first step. Consequently, SCOMP simply applies the greedy vertex cover algorithm. Now, thanks to Lemma 12 it suffices to prove that SCOMP will fail w.h.p. if $V_0^+ = \omega(k)$. Because they belong to positive tests only, all the individuals of V_0^+ are present in the vertex cover instance that SCOMP attempts to solve. Moreover, in the hypergraph no vertex has degree greater than Δ, because the degrees of $x_1, \ldots, x_n$ in $\boldsymbol{G}$ are equal to Δ. (Some of the hypergraph degrees may be strictly smaller than Δ because $\boldsymbol{G}$ is a multi-graph.) Therefore, since $|V_0^+| \geq k \log n$ while the actual set of infected individuals only has size k, w.h.p. the individual classified as infected by the very first step of the greedy set cover algorithm belongs to V_0^+. Hence, this individual is not actually infected, i.e., SCOMP errs w.h.p. ◀

Since the success probability of the SCOMP algorithm is at least as high as of the DD algorithm, we can prove the conjecture of [28] regarding the upper bound of the DD algorithm.

▶ **Corollary 13.** *If $m < (1-\varepsilon)m_{\mathrm{alg}}$, the DD algorithm will fail to retrieve the correct set of infected individuals w.h.p..*

3 Proof of Proposition 8

For $i \in [m]$ let Γ_i be the degree of a_i in $\boldsymbol{G}$, i.e., the number of edges incident with a_i; this number may exceed the number of different individuals that participate in test a_i as $\boldsymbol{G}$ may feature multi-edges. Let $\mathcal{G}$ be the σ-algebra generated by the random variables $(\Gamma_i)_{i\in[m]}$.

Given $\mathcal{G}$ we can generate $\boldsymbol{G}$ from the well-known *pairing model* [27]. Specifically, we create a set $\{x_i\} \times [\Delta]$ of Δ clones of each individual as well as sets $\{a_i\} \times [\Gamma_i]$ of clones of the tests. Then we draw a perfect matching of the complete bipartite graph on the vertex sets $\bigcup_{i=1}^{n} \{x_i\} \times [\Delta]$, $\bigcup_{i=1}^{m} \{a_i\} \times [\Gamma_i]$ uniformly at random. For each matching edge linking a clone of x_i with a clone of a_j we insert an i-j-edge. The resulting bipartite random multi-graph has the same distribution as $\boldsymbol{G}$ given $\mathcal{G}$. As an immediate application of this observation we obtain the following estimate.

▶ **Lemma 14.** *For every integer $0 \le \ell < k$ we have*

$$\mathbb{E}[Z_{k,\ell}(\boldsymbol{G}, \hat{\boldsymbol{\sigma}}) \mid \mathcal{G}] \le O(1) \cdot \binom{k}{\ell}\binom{n-k}{k-\ell} \prod_{i=1}^{m} 1 - 2(1-k/n)^{\Gamma_i} + 2\left(1 - 2k/n + \ell/n\right)^{\Gamma_i} \tag{3.1}$$

Proof. We use the linearity of expectation. The product of the two binomial coefficients simply accounts for the number of assignments σ that have overlap ℓ with $\boldsymbol{\sigma}$. Hence, with $\mathcal{S}$ the event that one specific $\sigma \in \{0,1\}^V$ that has overlap ℓ with $\boldsymbol{\sigma}$ belongs to $S_{k,\ell}(\boldsymbol{G}, \hat{\boldsymbol{\sigma}})$, we need to show that

$$\mathbb{P}\left[\mathcal{S} \mid \mathcal{G}\right] \le \prod_{i=1}^{m} 1 - 2(1-k/n)^{\Gamma_i} + 2\left(1 - 2k/n + \ell/n\right)^{\Gamma_i}. \tag{3.2}$$

By symmetry we may assume that $\boldsymbol{\sigma}_{x_i} = \mathbf{1}\{i \le k\}$ and that $\sigma_{x_i} = \mathbf{1}\{i \le \ell\} + \mathbf{1}\{k < i \le 2k - \ell\}$.

To establish (3.2) we harness the pairing model. Namely, given $\mathcal{G}$ we can think of each test a_i as a bin of capacity Γ_i. Moreover, we think of each clone (x_i, h), $h \in [\Delta]$, of an individual as a ball. The ball is labelled $(\boldsymbol{\sigma}_{x_i}, \sigma_{x_i}) \in \{0,1\}^2$. The random matching that creates $\boldsymbol{G}$ effectively tosses the Δn balls randomly into the bins. Hence, for $i \in [m]$ and for $j \in [\Gamma_i]$ let us write $\boldsymbol{A}_{i,j} = (\boldsymbol{A}_{i,j,1}, \boldsymbol{A}_{i,j,2}) \in \{0,1\}^2$ for the label of the jth ball that ends up in bin number i. Then we are left to calculate

$$\mathbb{P}\left[\mathcal{S} \mid \mathcal{G}\right] = \mathbb{P}\left[\forall i \in [m] : \max_{j\in[\Gamma_i]} \boldsymbol{A}_{i,j,1} = \max_{j\in[\Gamma_i]} \boldsymbol{A}_{i,j,2} \,\middle|\, \mathcal{G}\right], \tag{3.3}$$

i.e., the probability that a test a_i is positive with respect to first assignment $(\boldsymbol{A}_{i,j,1})_{j\in[\Gamma_i]}$ iff it is positive with respect to the second assignment $(\boldsymbol{A}_{i,j,2})_{j\in[\Gamma_i]}$.

To calculate this probability we borrow a trick from the analysis of the random k-SAT model [16]. Namely, we consider a new set $\{0,1\}^2$-valued random variables $\boldsymbol{A}'_{i,j} = (\boldsymbol{A}'_{i,j,1}, \boldsymbol{A}''_{i,j,2})$ such that $(\boldsymbol{A}'_{i,j})_{i\in[m],j\in[\Gamma_i]}$ are mutually independent and such that

$$\mathbb{P}\left[\boldsymbol{A}'_{i,j} = (1,1)\right] = \ell/n, \qquad \mathbb{P}\left[\boldsymbol{A}'_{i,j} = (0,1)\right] = \mathbb{P}\left[\boldsymbol{A}'_{i,j} = (1,0)\right] = (k-\ell)/n,$$
$$\mathbb{P}\left[\boldsymbol{A}'_{i,j} = (0,0)\right] = (n - 2k + \ell)/n$$

for all i, j. Now, let $\mathcal{R}$ be the event that

$$\sum_{i=1}^{m}\sum_{j=1}^{\Gamma_i}\mathbf{1}\left\{\boldsymbol{A}'_{i,j}=(1,1)\right\}=\ell\Delta, \qquad \sum_{i=1}^{m}\sum_{j=1}^{\Gamma_i}\mathbf{1}\left\{\boldsymbol{A}'_{i,j}=(0,0)\right\}=(n-2k+\ell)\Delta,$$
$$\sum_{i=1}^{m}\sum_{j=1}^{\Gamma_i}\mathbf{1}\left\{\boldsymbol{A}'_{i,j}=(1,0)\right\}=\sum_{i=1}^{m}\sum_{j=1}^{\Gamma_i}\mathbf{1}\left\{\boldsymbol{A}'_{i,j}=(0,1)\right\}=(k-\ell)\Delta,$$

i..e, that all of the sums on the l.h.s. are *precisely* equal to their expected values. Then $\boldsymbol{A}'=(\boldsymbol{A}'_{i,j})_{i,j}$ given $\mathcal{R}$ is distributed precisely as $\boldsymbol{A}=(\boldsymbol{A}_{i,j})_{i,j}$. Hence, (3.3) yields

$$\mathbb{P}\left[\mathcal{S}\mid\mathcal{G}\right]=\mathbb{P}\left[\forall i\in[m]:\max_{j\in[\Gamma_i]}\boldsymbol{A}'_{i,j,1}=\max_{j\in[\Gamma_i]}\boldsymbol{A}'_{i,j,2}\mid\mathcal{G},\mathcal{R}\right]. \tag{3.4}$$

Thus, let $\mathcal{A}=\left\{\forall i\in[m]:\max_{j\in[\Gamma_i]}\boldsymbol{A}'_{i,j,1}=\max_{j\in[\Gamma_i]}\boldsymbol{A}'_{i,j,2}\right\}$. Because the $(\boldsymbol{A}'_{i,j})_{i,j}$ are mutually independent, we can easily compute the unconditional probability $\mathcal{A}$: by inclusion/exclusion,

$$\mathbb{P}\left[\mathcal{A}\mid\mathcal{G}\right]=\prod_{i=1}^{m}1-2(1-k/n)^{\Gamma_i}+2(1-2k/n+\ell/n)^{\Gamma_i} \tag{3.5}$$

(the probability that $\max\boldsymbol{A}'_{i,j,1}=\max\boldsymbol{A}'_{i,j,2}=1$, i.e., both tests positive, equals one minus the probability that $\max\boldsymbol{A}'_{i,j,1}=0$ minus the probability that $\max\boldsymbol{A}'_{i,j,2}=0$ plus the probability that $\max\boldsymbol{A}'_{i,j,1}=\max\boldsymbol{A}'_{i,j,2}=0$; then add the probability that $\max\boldsymbol{A}'_{i,j,1}=\max\boldsymbol{A}'_{i,j,2}=0$, i.e., both tests negative).

Finally, to deal with the conditioning we use Bayes' rule:

$$\mathbb{P}\left[\mathcal{A}\mid\mathcal{R},\mathcal{G}\right]=\frac{\mathbb{P}\left[\mathcal{A}\mid\mathcal{G}\right]\mathbb{P}\left[\mathcal{R}\mid\mathcal{A},\mathcal{G}\right]}{\mathbb{P}\left[\mathcal{R}\mid\mathcal{G}\right]}. \tag{3.6}$$

Since the $(\boldsymbol{A}'_{i,j})_{i,j}$ are independent, the Local Limit Theorem for sums of independent variables [29] yields $\mathbb{P}\left[\mathcal{R}\mid\mathcal{G}\right]=\Theta(\Delta n)^{-3/2}$, $\mathbb{P}\left[\mathcal{R}\mid\mathcal{A},\mathcal{G}\right]=\Theta(\Delta n)^{-3/2}$. Hence, (3.2) follows from (3.4)–(3.6). ◀

Proof of Proposition 8. The Chernoff bound implies that $\Gamma_i\geq\Gamma_{\min}=\Delta n/m-\sqrt{\Delta n/m}\log n$ for all $i\in[m]$ w.h.p. Further, assuming that the Γ_i satisfy this bound, we perform an elementary calculation to check that

$$\sum_{0\leq\ell\leq(1-1/\log n)k}\binom{k}{\ell}\binom{n-k}{k-\ell}\prod_{i=1}^{m}1-2(1-k/n)^{\Gamma_i}+2\left(1-2k/n+\ell/n\right)^{\Gamma_i}=o(1). \tag{3.7}$$

Therefore, the proposition follows from Lemma 14 and Markov's inequality. ◀

4 Proof of Proposition 9

The argument from Section 3 does not extend large overlaps (close to k) because the expression on the r.h.s. of (3.1) gets too large. In other words, merely just computing the expected number of solutions with a given overlap does not do the trick. This "lottery phenomenon" is ubiquitous in random constraint satisfaction problems: for big overlap values rare solution-rich instances drive up the expected number of solutions [4, 5]. In order to cope with this issue we take another leaf out of the random CSP literature [2, 33]. Namely, we show that the solution $\boldsymbol{\sigma}$ is locally rigid. That is, the expansion properties of the random bipartite graph $\boldsymbol{G}$ preclude the existence of other solutions that have a big overlap with $\boldsymbol{\sigma}$. The following lemma holds the key to this effect.

▶ **Lemma 15.** *For any $\varepsilon > 0$ there exists $\delta = \delta(\varepsilon) > 0$ such that for all $m > (1+\varepsilon)m_{\inf}$ the following is true. Let $\mathcal{R}$ be the event that for every x_i with $\boldsymbol{\sigma}_{x_i} = 1$ there are at least $\delta\Delta$ tests $a \in \partial x_i$ such that $\partial a \setminus \{x_i\} \subseteq V_0$. Then $\mathbb{P}[\mathcal{R}] = 1 - o(1)$.*

Hence, w.h.p. any infected individual appears in plenty of tests where all the other individuals are healthy. This property causes $\boldsymbol{\sigma}$ to be locally rigid. To see why, consider the repercussions of just changing the status of a single individual x_i from infected to healthy. Because given $\mathcal{R}$ the individual x_i appears as the only infected individual in at least $\delta\Delta$ tests, in order to maintain the same tests results we will also need to flip at least one individual in each of these tests from healthy to infected. Since tests typically have relatively few individuals in common, the necessary number of flips from 0 to 1 will be $\Omega(\Delta) = \Omega(\log n)$. But then in order to keep the total number of infected individuals constant k, we will need to perform another $\Omega(\Delta)$ flips from 1 to 0. Yet given $\mathcal{R}$ each of these "second generation" individuals that we flip from infected to healthy is itself the only infected individual in many tests. Thus, the single flip that we started from triggers a veritable avalanche of flips, which will stop only after the overlap has dropped significantly. The next lemma formalises this intuition. The lemma shows that while the unconditional expectation of $Z_{k,\ell}(\boldsymbol{G}, \hat{\boldsymbol{\sigma}})$ is "too big", the conditional expectation of $Z_{k,\ell}(\boldsymbol{G}, \hat{\boldsymbol{\sigma}})$ given $\mathcal{R}$ is much smaller. Let $\boldsymbol{m}_0 = \boldsymbol{m}_0(\boldsymbol{G}, \hat{\boldsymbol{\sigma}})$ be the total number of negative tests.

▶ **Lemma 16.** *Suppose that $(1 - 1/\log n)k \le \ell < k$ and let $\Gamma_{\min} = \min_{i\in[m]} \Gamma_i$, $\Gamma_{\max} = \max_{i\in[m]} \Gamma_i$. Then*

$$\mathbb{E}[Z_{k,\ell}(\boldsymbol{G}, \hat{\boldsymbol{\sigma}}) \mid \mathcal{G}, \mathcal{R}, \boldsymbol{m}_0] \le$$
$$O(1)\binom{k}{\ell}\binom{n-k}{k-\ell}\left(1 - \left(1 - \frac{k-\ell}{n}\right)^{\Gamma_{\max}}\right)^{\delta\Delta(k-\ell)}\left(\frac{n-2k+\ell}{n-k}\right)^{\Gamma_{\min}\boldsymbol{m}_0} = o(1) \qquad (4.1)$$

The proof of Lemma 16 requires some mildly delicate manoeuvres to cope with the stochastic dependences that are inherent in the random bipartite graph model.

Proof of Proposition 9. Standard tail bound arguments show that $\Gamma_{\max}, \Gamma_{\min} = \min_{i\in[m]} \Gamma_i = \Delta n/m + O(\sqrt{\Delta n/m \log n})$ w.h.p. Plugging these estimates into (4.1) and summing on $\ell > (1 - 1/\log n)k$ completes the proof. ◀

References

1 E. Abbe. Community Detection and Stochastic Block Models: Recent Developments. *Journal of Machine Learning Research*, 18:1–86, 2018.

2 D. Achlioptas and A. Coja-Oghlan. Algorithmic barriers from phase transitions. *Proc. 49th FOCS*, pages 793–802, 2008.

3 D. Achlioptas, A. Coja-Oghlan, and F. Ricci-Tersenghi. On the solution space geometry of random formulas. *Random Structures and Algorithms*, 38:251–268, 2011.

4 D. Achlioptas and C. Moore. Random k-SAT: two moments suffice to cross a sharp threshold. *SIAM Journal on Computing*, 36:740–762, 2006.

5 D. Achlioptas, A. Naor, and Y. Peres. Rigorous location of phase transitions in hard optimization problems. *Nature*, 435:759–764, 2005.

6 D. Achlioptas and Y. Peres. The threshold for random k-SAT is $2^k \log 2 - O(k)$. *Journal of the AMS*, 17:947–973, 2004.

7 M. Aldridge. The capacity of Bernoulli nonadaptive group testing. *IEEE Transactions on Information Theory*, PP, 2015.

8 M. Aldridge. On the optimality of some group testing algorithms. *Proceedings of IEEE International Symposium on Information Theory*, pages 3085–3089, 2017.

9 M. Aldridge. Individual testing is optimal for nonadaptive group testing in the linear regime. *IEEE Transactions on Information Theory*, 65:2058–2061, 2018.

10 M. Aldridge, L. Baldassini, and O. Johnson. Group testing algorithms: bounds and simulations. *IEEE Transactions on Information Theory*, 60:3671–3687, 2014.

11 M. Aldridge, O. Johnson, and J. Scarlett. Improved group testing rates with constant column weight designs. *Proceedings of IEEE International Symposium on Information Theory*, pages 1381–1385, 2016.

12 A. Alleman. An efficient algorithm for combinatorial group testing. *H. Aydinian, F. Cicalese, C. Deppe (eds) Information Theory, Combinatorics, and Search Theory. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg*, 7777:569–596, 2013.

13 R. Benz, S. Swamidass, and P. Baldi. Discovery of power-laws in chemical space. *Journal of Chemical Information and Modeling*, 48:1138–1151, 2008.

14 H. Chen and F. Hwang. A survey on nonadaptive group testing algorithms through the angle of decoding. *Journal of Combinatorial Optimization*, 15:49–59, 2008.

15 A. Coja-Oghlan, F. Krzakala, W. Perkins, and L. Zdeborová. Information-theoretic thresholds from the cavity method. *Advances in Mathematics*, 333:694–795, 2018.

16 A. Coja-Oghlan and K. Panagiotou. The asymptotic k-SAT threshold. *Advances in Mathematics*, 288:985–1068, 2016.

17 A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Phys. Rev. E*, 84:066106, 2011.

18 J. Ding, A. Sly, and N. Sun. Proof of the satisfiability conjecture for large k. *Proc. 47th STOC*, pages 59–68, 2015.

19 R. Dorfman. The detection of defective members of large populations. *Annals of Mathematical Statistics*, 14:436–440, 1943.

20 D. Du and F. Hwang. *Combinatorial group testing and its applications*. World Scientific, 2000.

21 O. Dubois and J. Mandler. The 3-XORSAT Threshold. *Proc. 43rd FOCS*, pages 769–778, 2002.

22 A. Emad and O. Milenkovic. Poisson group testing: a probabilistic model for nonadaptive streaming Boolean compressed sensing. *Proc. ICASSP*, pages 3335–3339, 2014.

23 V. Feldman, E. Grigorescu, L. Reyzin, S. Vempala, and Y. Xiao. Statistical algorithms and a lower bound for detecting planted clique. *Proc. 45th STOC*, pages 655–664, 2013.

24 V. Feldman, W. Perkins, and S. Vempala. On the complexity of random satisfiability problems with planted solutions. *Proc. 48th STOC*, pages 77–86, 2015.

25 D. Gamarnik and M. Sudan. Performance of sequential local algorithms for the random NAE-K-SAT problem. *SIAM J. on Computing*, 46:590–619, 2017.

26 S. Hopkins, P. Kothari, A. Potechin, P. Raghavendra, T. Schramm, and D. Steurer. The power of sum-of-squares for detecting hidden structures. *Proc. 58th FOCS*, pages 720–731, 2017.

27 S. Janson, T. Luczak, and A. Ruci'nski. *Random Graphs*. Wiley, 2000.

28 O. Johnson, M. Aldridge, and J. Scarlett. Performance of group testing algorithms with near-constant tests per item. *IEEE Transactions on Information Theory*, 65:707–723, 2019.

29 A. Kolmogorov, A. Nikolaevich, and B. Gnedenko. Limit distributions for sums of independent random variables. *Addison-Wesley*, 1968.

30 H. Kwang-Ming and D. Ding-Zhu. Pooling designs and nonadaptive group testing: important tools for DNA sequencing. *World Scientific*, 2006.

31 M. Mézard and A. Montanari. *Information, physics and computation*. Oxford University Press, 2009.

32 M. Mézard, M. Tarzia, and C. Toninelli. Group testing with random pools: phase transitions and optimal strategy. *Journal of Statistical Physics*, 131:783–801, 2008.

33 M. Molloy. The freezing threshold for k-colourings of a random graph. *Proc. 43rd STOC*, pages 921–930, 2012.

34 C. Moore. The Computer Science and Physics of Community Detection: Landscapes, Phase Transitions, and Hardness. *Bulletin of the EATCS*, 121, 2017.

35 E. Mossel, J. Neeman, and A. Sly. Reconstruction and estimation in the planted partition model. *Probability Theory and Related Fields*, pages 1–31, 2014.

36 R. Mourad, Z. Dawy, and F. Morcos. Designing pooling systems for noisy high-throughput protein-protein interaction experiments using Boolean compressed sensing. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10:1478–1490, 2013.

37 H. Ngo and D. Du. A survey on combinatorial group testing algorithms with applications to DNA library screening. *Discrete Mathematical Problems with Medical Applications*, 7:171–182, 2000.

38 J. Scarlett and V. Cevher. Phase transitions in group testing. *Proc. 27th SODA*, pages 40–53, 2016.

39 J. Scarlett and V. Cevher. Limits on support recovery with probabilistic models: an information-theoretic framework. *IEEE Transactions on Information Theory*, 63:593–620, 2017.

40 N. Thierry-Mieg. A new pooling strategy for high-throughput screening: the shifted transversal design. *BMC Bioinformatics*, 7:28, 2006.

41 L. Zdeborová and F. Krzakala. Statistical physics of inference: thresholds and algorithms. *Advances in Physics*, 65:453–552, 2016.

On Reachability Problems for Low-Dimensional Matrix Semigroups

Thomas Colcombet
IRIF, CNRS, Université Paris Diderot, France
thomas.colcombet@irif.fr

Joël Ouaknine
The Max Planck Institute for Software Systems, Saarbrücken, Germany
Department of Computer Science, University of Oxford, United Kingdom
joel@mpi-sws.org

Pavel Semukhin
Department of Computer Science, University of Oxford, United Kingdom
pavel.semukhin@cs.ox.ac.uk

James Worrell
Department of Computer Science, University of Oxford, United Kingdom
jbw@cs.ox.ac.uk

Abstract

We consider the Membership and the Half-Space Reachability problems for matrices in dimensions two and three. Our first main result is that the Membership Problem is decidable for finitely generated sub-semigroups of the Heisenberg group over rational numbers. Furthermore, we prove two decidability results for the Half-Space Reachability Problem. Namely, we show that this problem is decidable for sub-semigroups of $\mathrm{GL}(2,\mathbb{Z})$ and of the Heisenberg group over rational numbers.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory; Computing methodologies → Symbolic and algebraic algorithms

Keywords and phrases membership problem, half-space reachability problem, matrix semigroups, Heisenberg group, general linear group

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.44

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1902.09597`, [11].

Funding *Thomas Colcombet*: Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No.670624), and by the DeLTA ANR project (ANR-16-CE40-0007).
Joël Ouaknine: Supported by ERC grant AVS-ISS (648701) and by DFG grant 389792660 as part of TRR 248 (see `https://perspicuous-computing.science`).
Pavel Semukhin: Supported by ERC grant AVS-ISS (648701).
James Worrell: Supported by EPSRC Fellowship EP/N008197/1.

1 Introduction

The algorithmic theory of matrix groups and semigroups is a staple of computational algebra [3] with numerous applications to automata theory and program analysis [7, 10, 12, 19, 20, 27] and has been influential in developing the notion of interactive proofs in complexity theory [1].

Two central decision problems on matrix semigroups are the *Membership* and *Half-Space Reachability* (see, e.g., [6]). For the Membership Problem the input is a finite set of generators $A_1, \ldots, A_k$ and a target matrix A, with all matrices being square and of the same dimension. The question is whether A lies in the semigroup generated by $A_1, \ldots, A_k$. We emphasize that we consider membership in finitely generated *sub-semigroups*, i.e., we seek to recover A as a non-empty product of generators. In a related *subgroup* membership problem one additionally allows to take inverses of generators. The subgroup membership can clearly be reduced to the sub-semigroup membership and tends to be more tractable (e.g., the subgroup membership for polycyclic groups is well-known to be decidable [38], and the subgroup membership for the modular group $\mathrm{PSL}(2,\mathbb{Z})$ is in PTIME [15]). For the Half-Space Reachability Problem the target matrix is replaced by vectors $\boldsymbol{u}$, $\boldsymbol{v}$ and a scalar λ, and the question is now whether there exists a matrix A in the semigroup generated by $A_1, \ldots, A_k$ such that $\boldsymbol{u}^\top A \boldsymbol{v} \geq \lambda$. Geometrically the question is whether the orbit of $\boldsymbol{v}$ under the action of the semigroup reaches a certain half-space with normal $\boldsymbol{u}$. Closely related to these problems are the Vector Reachability and the Hyperplane Reachability[1] problems, which ask whether there exists a matrix A in the semigroup generated by $A_1, \ldots, A_k$ such that $A\boldsymbol{v} = \boldsymbol{u}$ or such that $\boldsymbol{u}^\top A \boldsymbol{v} = \lambda$, respectively.

Undecidability of the Membership Problem has long been known (indeed, this was one of the earliest undecidability results – see A. Markov [28]). Subsequently a number of positive decidability results were obtained in the case of semigroups generated by commuting matrices over infinite fields [2, 20]. More recently, attention has focussed on integer matrices in dimension two. A classical result of [10] shows decidability of the Membership Problem for sub-semigroups of $\mathrm{GL}(2,\mathbb{Z})$ – the group of 2×2 integer matrices with integer inverses (equivalently, with determinants equal to ± 1). Moreover, the semigroup membership for the identity matrix was shown to be NP-complete for $\mathrm{SL}(2,\mathbb{Z})$ [4]. Furthermore, the Membership Problem is decidable for 2×2 integer matrices with nonzero determinant [32] and for 2×2 integer matrices with determinants equal to 0 and ± 1 [33]. However it is still unknown whether the Membership Problem is decidable for all 2×2 integer matrices.

Going beyond dimension two, it has long been known that the Membership Problem is undecidable for general 3×3 integer matrices [31]. However the status of the Membership Problem for $\mathrm{GL}(3,\mathbb{Z})$ is currently an outstanding open problem. Related to this, it was shown in [23] that for a two-element alphabet Σ, the monoid $\Sigma^* \times \Sigma^*$ cannot be embedded in $\mathrm{GL}(3,\mathbb{Z})$. This fact suggests that undecidability proofs of the Membership Problem in other settings (such as [31]), which are based on encodings of the Post Correspondence Problem, are unlikely to carry over to $\mathrm{GL}(3,\mathbb{Z})$. It is classical that the Membership Problem for $\mathrm{GL}(4,\mathbb{Z})$ is undecidable [29, 5, 23]; thus it can reasonably be said that dimension three lies on the borderline between decidability and undecidability.

Our first main result (Theorem 7) concerns the Membership Problem for a simple subgroup of $\mathrm{GL}(3,\mathbb{Z})$: the so-called *Heisenberg group* $\mathrm{H}(3,\mathbb{Z})$, which comprises upper triangular integer matrices with ones along the diagonal. Since the Heisenberg group is polycyclic, the sub*group* membership problem is decidable [38]. It was moreover recently shown in [23] how to decide membership of the identity matrix in finitely generated sub-semigroups of $\mathrm{H}(3,\mathbb{Z})$. Our main theorem strengthens this last result to show decidability of the Membership Problem for $\mathrm{H}(3,\mathbb{Z})$. In fact, like in [23], our argument works for Heisenberg groups of any dimension and even over the field of rational numbers, that is, for $\mathrm{H}(n,\mathbb{Q})$.

[1] In the literature the Hyperplane Reachability Problem is also called the Scalar Reachability Problem.

Our proof relies on arguments developed in [23] but contains several significant new elements, including the use of linear programming, integer register automata, and matrix logarithms. The following algebraic property of $\mathrm{H}(3,\mathbb{Z})$ is important for our construction: the subgroup generated by commutators of matrices from a given subset $\mathcal{G} \subseteq \mathrm{H}(3,\mathbb{Z})$ is isomorphic to a subgroup of $\mathbb{Z}$. Such property does not hold for the direct product of two Heisenberg groups $\mathrm{H}(3,\mathbb{Z})^2$ or for the group of 4×4 upper unitriangular matrices $\mathrm{UT}(4,\mathbb{Z})$. This makes it challenging to generalize our argument to show decidability of the Membership Problem for $\mathrm{H}(3,\mathbb{Z})^2$, $\mathrm{UT}(4,\mathbb{Z})$ or other similar matrix groups.

In [24] a related problem was studied, called the Knapsack Problem. Namely, it was proved that the Knapsack Problem is decidable for $\mathrm{H}(3,\mathbb{Z})$, that is, given matrices $A_1,\ldots,A_k$ and A from $\mathrm{H}(3,\mathbb{Z})$ one can decide whether there are non-negative integers $n_1,\ldots,n_k$ such that $A_1^{n_1}\cdots A_k^{n_k} = A$. Decidability of the Knapsack Problem is shown by reduction to the problem of solving a single quadratic equation in integer numbers (proved to be decidable in [13, 14]). By contrast, our decision procedure for the Membership Problem relies only on linear programming and integer linear arithmetic. As far as we can tell, there is no straightforward reduction in either direction between the Membership and Knapsack Problems for $\mathrm{H}(3,\mathbb{Z})$.

The Vector Reachability, Hyperlane Reachability, and Half-Space Reachabilty Problems are all known to be undecidable in general (see [9, 16, 17]). The Vector and Hyperplane Reachability problems are known to be decidable for $\mathrm{GL}(2,\mathbb{Z})$, as shown in [34]. For matrix semigroups with a single generator, the Half-Space Reachability Problem is equivalent to the *Positivity Problem* for linear recurrence sequences: a longstanding and apparently difficult open problem [30, 36]. Our second main result is that the Half-Space Reachability Problem is decidable for both $\mathrm{GL}(2,\mathbb{Z})$ (Theorem 17) and $\mathrm{H}(n,\mathbb{Q})$ (Theorem 20). For $\mathrm{GL}(2,\mathbb{Z})$ we build on automata-theoretic techniques developed in [10], with the key insight being that the set of matrices in $\mathrm{GL}(2,\mathbb{Z})$ with a positive value in a given entry can be represented as a regular language over the generators of $\mathrm{GL}(2,\mathbb{Z})$. For $\mathrm{H}(n,\mathbb{Q})$ we rely on a nontrivial result about the nonnegativity of quadratic forms over the integers from [13, 14] (related to the result used in [24] to solve the Knapsack Problem).

2 Preliminaries

The Heisenberg Group

We use notations I_n and 0_n for the identity matrix and for the zero matrix of size $n \times n$, respectively. For $n \geq 3$, the *Heisenberg group* of dimension n is the group $\mathrm{H}(n,\mathbb{R})$ of $n \times n$ real matrices of the form

$$A = \begin{pmatrix} 1 & \boldsymbol{a}^\top & c \\ 0 & I_{n-2} & \boldsymbol{b} \\ 0 & 0 & 1 \end{pmatrix}, \tag{1}$$

where $\boldsymbol{a},\boldsymbol{b} \in \mathbb{R}^{n-2}$, $c \in \mathbb{R}$. For brevity, we will often denote a matrix A as in (1) by the triple $(\boldsymbol{a},\boldsymbol{b},c) \in \mathbb{R}^{n-2} \times \mathbb{R}^{n-2} \times \mathbb{R}$. It is easy to check that the product operation is given by

$$(\boldsymbol{a},\boldsymbol{b},c)\cdot(\boldsymbol{a}',\boldsymbol{b}',c') = (\boldsymbol{a}+\boldsymbol{a}',\boldsymbol{b}+\boldsymbol{b}',c+c'+\boldsymbol{a}^\top\boldsymbol{b}').$$

We use ψ to denote the group homomorphism $\psi : \mathrm{H}(n,\mathbb{R}) \to \mathbb{R}^{2n-4}$ given by $\psi(\boldsymbol{a},\boldsymbol{b},c) = (\boldsymbol{a},\boldsymbol{b})$.

The Heisenberg group $\mathrm{H}(n,\mathbb{R})$ is a Lie group whose corresponding Lie algebra $\mathfrak{h}(n,\mathbb{R})$ comprises the vector space of $n \times n$ real matrices of the form

$$B = \begin{pmatrix} 0 & \boldsymbol{a}^\top & c \\ 0 & 0_{n-2} & \boldsymbol{b} \\ 0 & 0 & 0 \end{pmatrix}, \tag{2}$$

where $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^{n-2}$ and $c \in \mathbb{R}$, together with the binary *Lie bracket* operation $[A,B] := AB - BA$ for $A, B \in \mathfrak{h}(n,\mathbb{R})$. Note that $[A,B]$ has only zero entries except for the $(1,n)$-entry. From this it is easy to check that $[[A,B],C] = 0_n$ for all $A, B, C \in \mathfrak{h}(n,\mathbb{R})$.

Given $A \in \mathrm{H}(n,\mathbb{R})$, as shown in (1), we define its logarithm $\log(A) \in \mathfrak{h}(n,\mathbb{R})$ to be

$$\log(A) := (A-I) - \frac{(A-I)^2}{2} = \begin{pmatrix} 0 & \boldsymbol{a}^\top & c - \frac{1}{2}\boldsymbol{a}^\top\boldsymbol{b} \\ 0 & 0_{n-2} & \boldsymbol{b} \\ 0 & 0 & 0 \end{pmatrix}.$$

Conversely, given $B \in \mathfrak{h}(n,\mathbb{R})$, as shown in (2), we define its exponential $\exp(B) \in \mathrm{H}(n,\mathbb{R})$ to be $\exp(B) := I + B + \frac{B^2}{2} = (\boldsymbol{a}, \boldsymbol{b}, c + \frac{1}{2}\boldsymbol{a}^\top\boldsymbol{b})$. It is easy to verify that log and exp are mutually inverse and together induce a bijection between $\mathrm{H}(n,\mathbb{R})$ and $\mathfrak{h}(n,\mathbb{R})$.

The following is a specialisation to $\mathrm{H}(n,\mathbb{R})$ of the Baker-Campbell-Hausdorff product formula (see [18, Chapter 5] for a details). Given a sequence of matrices $B_1, \ldots, B_m \in \mathrm{H}(n,\mathbb{R})$, we have

$$\log(B_1 \cdots B_m) = \sum_{i=1}^{m} \log(B_i) + \frac{1}{2} \sum_{1 \le i < j \le m} [\log(B_i), \log(B_j)]\,. \tag{3}$$

Regular subsets of $\mathbf{GL(2,\mathbb{Z})}$

We will use the notation $\mathrm{GL}(2,\mathbb{Z})$ for the general linear group of 2×2 integer matrices, that is, $\mathrm{GL}(2,\mathbb{Z}) = \{M \in \mathbb{Z}^{2\times 2} : \det(M) = \pm 1\}$. A matrix is called *singular* if its determinant is zero and *nonsingular* otherwise.

We will use the following encoding of the matrices from $\mathrm{GL}(2,\mathbb{Z})$ by words in alphabet $\Sigma = \{X, N, S, R\}$. First, we define a mapping $\varphi : \Sigma \to \mathrm{GL}(2,\mathbb{Z})$ as follows:

$$\varphi(X) = -I_2 = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix},\ \varphi(N) = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},\ \varphi(S) = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix},\ \varphi(R) = \begin{pmatrix} 0 & -1 \\ 1 & 1 \end{pmatrix}.$$

We can extend φ to a morphism $\varphi : \Sigma^* \to \mathrm{GL}(2,\mathbb{Z})$ in a natural way. It is a well-known fact that morphism φ is surjective, that is, for every $M \in \mathrm{GL}(2,\mathbb{Z})$ there is a word $w \in \Sigma^*$ such that $\varphi(w) = M$. This presentation is not unique because of identities such as $\varphi(SS) = \varphi(RRR) = \varphi(X)$. However, as explained below, every matrix $M \in \mathrm{GL}(2,\mathbb{Z})$ is represented by a unique word in the *canonical* form.

In the following definition, for n a positive integer and $V \in \Sigma$, V^n is the word consisting of n copies of V, while V^0 denotes the empty word.

▶ **Definition 1.** *A word $w \in \Sigma^*$ is called* canonical *if it has the form*

$$w = N^\delta X^\gamma S^\beta R^{\alpha_1} S R^{\alpha_2} \cdots S R^{\alpha_n} S^\varepsilon,$$

where $\beta, \gamma, \delta, \varepsilon \in \{0,1\}$ and $\alpha_i \in \{1,2\}$ for $i = 1, \ldots, n$. In other words, w is canonical *if it does not contain subwords SS or RRR. Moreover, letter N may appear only once in the first position, and letter X may appear only once either in the first position or after N.*

The next proposition is a well-known fact.

▶ **Proposition 2** ([25, 26, 32, 35]). *For every matrix $M \in \mathrm{GL}(2,\mathbb{Z})$, there is a unique canonical word w such that $M = \varphi(w)$.*

▶ **Definition 3.** *A subset $\mathcal{S} \subseteq \mathrm{GL}(2,\mathbb{Z})$ is called* regular *if there is a regular language $L \subseteq \Sigma^*$ such that $\mathcal{S} = \varphi(L)$.*

▶ **Definition 4.** *Two words w_1 and w_2 from Σ^* are* equivalent*, denoted $w_1 \sim w_2$, if $\varphi(w_1) = \varphi(w_2)$. Two languages L_1 and L_2 in the alphabet Σ are* equivalent*, denoted $L_1 \sim L_2$, if*

(i) *for each $w_1 \in L_1$, there exists $w_2 \in L_2$ such that $w_1 \sim w_2$, and*
(ii) *for each $w_2 \in L_2$, there exists $w_1 \in L_1$ such that $w_2 \sim w_1$.*

In other words, $L_1 \sim L_2$ if and only if $\varphi(L_1) = \varphi(L_2)$. Two finite automata $\mathcal{A}_1$ and $\mathcal{A}_2$ with alphabet Σ are equivalent*, denoted $\mathcal{A}_1 \sim \mathcal{A}_2$, if $L(\mathcal{A}_1) \sim L(\mathcal{A}_2)$.*

The following theorem is a crucial ingredient of our decidability results.

▶ **Theorem 5** ([32]). *For any automaton $\mathcal{A}$ over the alphabet $\Sigma = \{X, N, S, R\}$, there exists an automaton $\mathrm{Can}(\mathcal{A})$ such that $\mathrm{Can}(\mathcal{A})$ is equivalent to $\mathcal{A}$ and $\mathrm{Can}(\mathcal{A})$ accepts only canonical words. Furthermore, $\mathrm{Can}(\mathcal{A})$ can be constructed from $\mathcal{A}$ in polynomial time.*

The proof of the following corollary is given in the full version [11].

▶ **Corollary 6.** *Regular subsets of $\mathrm{GL}(2,\mathbb{Z})$ are effectively closed under Boolean operations. Namely, given two regular languages $L, L' \subseteq \Sigma^*$, we can algorithmically construct in polynomial time regular languages $L^\cup$, $L^\cap$ and L^c such that $\varphi(L^\cup) = \varphi(L) \cup \varphi(L')$, $\varphi(L^\cap) = \varphi(L) \cap \varphi(L')$, and $\varphi(L^c) = \mathrm{GL}(2,\mathbb{Z}) \setminus \varphi(L)$.*

Decision problems for matrix semigroups

If $\mathcal{G}$ is a finite collection of matrices, then $\langle\mathcal{G}\rangle$ denotes the semigroup generated by $\mathcal{G}$, that is, $A \in \langle\mathcal{G}\rangle$ if and only if there are matrices $A_1, \ldots, A_t \in \mathcal{G}$ such that $A = A_1 \cdots A_t$.

In this paper we will consider the following decision problems for matrix semigroups:

- **The Membership Problem:** Given a finite collection of matrices $\mathcal{G}$ and a "target" matrix A, decide whether A belongs to $\langle\mathcal{G}\rangle$.
- **The Half-Space Reachability Problem:** Given a finite collection of matrices $\mathcal{G}$, two vectors $\boldsymbol{u}, \boldsymbol{v}$ and a scalar λ, decide whether there exists a matrix $A \in \langle\mathcal{G}\rangle$ such that $\boldsymbol{u}^\top A \boldsymbol{v} \geq \lambda$. In other words, decide whether it is possible to reach the half-space $\mathcal{H} = \{\boldsymbol{x} : \boldsymbol{u}^\top \boldsymbol{x} \geq \lambda\}$ using matrices from $\mathcal{G}$ starting from an initial vector $\boldsymbol{v}$.

When we talk about *the Membership Problem for* $\mathrm{GL}(2,\mathbb{Z})$ *or for the Heisenberg group* $\mathrm{H}(n,\mathbb{Q})$, we mean that A and the matrices from $\mathcal{G}$ belong to $\mathrm{GL}(2,\mathbb{Z})$ or $\mathrm{H}(n,\mathbb{Q})$, respectively. Similarly, in *the Half-Space Reachability Problem for* $\mathrm{GL}(2,\mathbb{Z})$ *or* $\mathrm{H}(n,\mathbb{Q})$ we assume that $\mathcal{G}$ is a finite subset of $\mathrm{GL}(2,\mathbb{Z})$ or $\mathrm{H}(n,\mathbb{Q})$, respectively, and furthermore we assume that the vectors $\boldsymbol{u}, \boldsymbol{v}$ have rational coefficients and λ is a rational number.

3 The Membership Problem for the Heisenberg Group

Let $\mathrm{H}(n,\mathbb{Z})$ and $\mathrm{H}(n,\mathbb{Q})$ be subgroups of $\mathrm{H}(n,\mathbb{R})$ comprising all matrices with integer and rational entries, respectively. In this section we will prove our first main result.

▶ **Theorem 7.** *The Membership Problem for $\mathrm{H}(n,\mathbb{Z})$ is decidable.*

We first give an overview of our decision procedure. Let $\mathcal{G} = \{A_1, \ldots, A_k\}$ be a finite set of generators from $\mathrm{H}(n, \mathbb{Z})$ and $A \in \mathrm{H}(n, \mathbb{Z})$ be a target matrix. The idea is to partition the set of generators $\mathcal{G}$ into two sets $\mathcal{G}_+$ and $\mathcal{G}_0$. The definition of $\mathcal{G}_+$ is such that there is a computable upper bound on the number of occurrences of a matrix from $\mathcal{G}_+$ in any string of generators whose product equals the target matrix A. The definition of $\mathcal{G}_0$ is such that the image of the semigroup generated by $\mathcal{G}_0$ under the homomorphism ψ is a sub*group* of $\mathbb{R}^{2n-4}$ (i.e., the image is closed under inverses). We then proceed by a case analysis according to whether or not $\mathcal{G}_0$ is a commutative set of matrices. If $\mathcal{G}_0$ is commutative then the Membership Problem can be reduced to solving a system of linear equations over non-negative integer variables. If $\mathcal{G}_0$ is not commutative then we reduce the Membership Problem to a reachability query in an integer register automaton.

Partitioning the Set of Generators

In the rest of this section we work with an instance of the Membership Problem in which the generators are $A_i = (\boldsymbol{a}_i, \boldsymbol{b}_i, c_i)$, for $i = 1, \ldots, k$, and the target matrix is $A = (\boldsymbol{a}, \boldsymbol{b}, c)$. Recalling the homomorphism $\psi : \mathrm{H}(n, \mathbb{Z}) \to \mathbb{Z}^{2n-4}$, let us define $\boldsymbol{v}_i := \psi(A_i) = (\boldsymbol{a}_i, \boldsymbol{b}_i)$, for $i = 1, \ldots, k$, and $\boldsymbol{v} := \psi(A) = (\boldsymbol{a}, \boldsymbol{b})$.

A set $C \subseteq \mathbb{R}^n$ is called a *cone* if $\sum_{i=1}^{k} r_i \boldsymbol{u}_i \in C$ for all $r_1, \ldots, r_k \in \mathbb{R}_{\geq 0}$ and $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k \in C$. The *dual* of a cone $C \subseteq \mathbb{R}^n$ is the cone defined as

$$C^* := \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{x}^\top \boldsymbol{y} \geq 0 \text{ for all } \boldsymbol{y} \in C\}.$$

We will use the fact that $C = C^{**}$, i.e., a cone is equal to its double dual [8, Chapter 2.6.1].

We write $\mathrm{Cone}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)$ for the cone generated by the vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$. We now partition the set of generators $\mathcal{G}$ into two disjoint sets $\mathcal{G}_0, \mathcal{G}_+$, where

$$\begin{aligned} \mathcal{G}_0 &:= \left\{A_i : \forall \boldsymbol{u} \in \mathrm{Cone}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)^* \quad \boldsymbol{v}_i^\top \boldsymbol{u} = 0\right\} \\ \mathcal{G}_+ &:= \left\{A_i : \exists \boldsymbol{u} \in \mathrm{Cone}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)^* \quad \boldsymbol{v}_i^\top \boldsymbol{u} > 0\right\} . \end{aligned}$$

We can determine the sets $\mathcal{G}_0$ and $\mathcal{G}_+$ using linear programming [37]. Without loss of generality we can assume that $\mathcal{G}_0 = \{A_1, \ldots, A_\ell\}$ for some $\ell \geq 0$.

We show how to compute a bound $\beta > 0$ such that for every sequence $\mathcal{S} = B_1, \ldots, B_m$ of elements of $\mathcal{G}$ whose product is equal to the target matrix A, the number of indices i such that $B_i \in \mathcal{G}_+$ is at most β. By definition of $\mathcal{G}_+$, for each $i \in \{1, \ldots, \ell\}$, there exists $\boldsymbol{u}_i \in \mathrm{Cone}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)^*$ such that $\boldsymbol{v}_i^\top \boldsymbol{u}_i > 0$. Since $\boldsymbol{v}_j^\top \boldsymbol{u}_i \geq 0$ for all $j \neq i$, $\mathcal{S}$ contains at most $\frac{\boldsymbol{v}^\top \boldsymbol{u}_i}{\boldsymbol{v}_i^\top \boldsymbol{u}_i}$ occurrences of matrix A_i (or no occurrences if $\boldsymbol{v}^\top \boldsymbol{u}_i \leq 0$). Thus we may define $\beta := \sum_i \frac{\boldsymbol{v}^\top \boldsymbol{u}_i}{\boldsymbol{v}_i^\top \boldsymbol{u}_i}$ where the sum is take over the indices $i = 1, \ldots, \ell$ such that $\boldsymbol{v}^\top \boldsymbol{u}_i > 0$

We now consider two cases according to whether $\mathcal{G}_0$ is a commutative set of matrices.

Case I: $\mathcal{G}_0$ is commutative

Consider a sequence $\mathcal{S} = B_1, \ldots, B_m$ of elements of $\mathcal{G}$. Let $B_{i_1}, \ldots, B_{i_s}$ be the subsequence of $\mathcal{S}$ containing all occurrences of elements of $\mathcal{G}_+$ in $\mathcal{S}$, where $0 = i_0 < i_1 < \ldots < i_s < i_{s+1} = m + 1$. For $i \in \{1, \ldots, \ell\}$ and $j \in \{1, \ldots, s+1\}$, write $n_{i,j}$ for the number of occurrences of $A_i \in \mathcal{G}_0$ in the subsequence of $\mathcal{S}$ lying strictly between $B_{i_{j-1}}$ and B_{i_j} (where B_0 is interpreted as the beginning of $\mathcal{S}$ and B_{m+1} as the end of $\mathcal{S}$). The idea is to write a formula for $\log(B_1 \cdots B_m)$ that is a linear form in the variables $n_{i,j}$.

Indeed by Equation (3), writing $C_{i_j} := \log(B_{i_j})$ for $j = 1, \ldots, s$ and $D_i := \log(A_i)$ for $i = 1, \ldots, \ell$, we have

$$\log(B_1 \cdots B_m) = \sum_{j=1}^{s} C_{i_j} + \sum_{i=1}^{\ell} \sum_{j=1}^{s+1} n_{i,j} D_i + \sum_{1 \le j < j' \le s} [C_{i_j}, C_{i_{j'}}] + \sum_{i=1}^{\ell} \sum_{1 \le j \le j' \le s} n_{i,j} [D_i, C_{i_{j'}}] + \sum_{i=1}^{\ell} \sum_{1 \le j < j' \le s+1} n_{i,j'} [C_{i_j}, D_i] \quad (4)$$

An important observation is that the above formula has no quadratic terms due to commutativity of $\mathcal{G}_0$. Now $B_1 \cdots B_m = A$ if and only if $\log(B_1 \cdots B_m) = \log(A)$. Setting the right-hand-side of (4) equal to $\log(A)$ yields a linear Diophantine equation in variables $n_{i,j}$. The form of this equation is determined by the subsequence of matrices $B_{i_1}, \ldots, B_{i_s}$ lying in $\mathcal{G}_+$. Recall that we can without loss of generality restrict attention to the case that $s \le \beta$ and thus we reduce the question of whether A lies in the semigroup generated by $\mathcal{G}$ to the solubility of finitely many linear equations in nonnegative integers.

Case II: $\mathcal{G}_0$ is not commutative

Let $\mathcal{G}_0 = \{A_1, \ldots, A_\ell\}$ for some $\ell \ge 2$ such that A_1 and A_2 do not commute. Recall that by definition of $\mathcal{G}_0$ it holds that $\boldsymbol{v}_i^\top \boldsymbol{u} = 0$ for all $\boldsymbol{u} \in \mathrm{Cone}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)^*$ and $i = 1, \ldots, \ell$. Therefore,

$$\mathrm{Span}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_\ell) \subseteq \mathrm{Cone}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)^{**} = \mathrm{Cone}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)\,. \quad (5)$$

Following ideas from [23], we will show that there exist integers $p > 0$ and $q < 0$ such that $M_+ = (\mathbf{0}, \mathbf{0}, p)$ and $M_- = (\mathbf{0}, \mathbf{0}, q)$ and both lie in the semigroup generated by $\mathcal{G}$.

Indeed, from Equation (5) it follows that $-(\boldsymbol{v}_1 + \boldsymbol{v}_2)$ lies in $\mathrm{Cone}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)$. Thus there exist $r_1, \ldots, r_k \in \mathbb{R}_{\ge 0}$ with r_1, r_2 strictly positive such that $\sum_{i=1}^{k} r_i \boldsymbol{v}_i = \mathbf{0}$. But since the vectors $\boldsymbol{v}_i$ have integer coefficients we can solve the above equation in natural numbers $r_1, \ldots, r_k$ with $r_1, r_2 > 0$. Taking a sequence of matrices $B_1, \ldots, B_m$, drawn from $\mathcal{G}$, such that $B_1 = A_1$, $B_2 = A_2$ and such that matrix A_i appears r_i times in the sequence for $i \in \{1, \ldots, k\}$, we obtain $\psi(B_1 \cdots B_m) = \mathbf{0}$. Since ψ is a homomorphism to a commutative group we have that $\psi(B_{\sigma(1)}^t \cdots B_{\sigma(m)}^t) = \mathbf{0}$ for all $t \ge 1$ and permutations $\sigma \in S_m$.

Write $C_i = \log(B_i)$ for $i = 1, \ldots, m$. Applying the Baker-Campbell-Hausdorff Formula (3), we have that for any positive integer t and permutation $\sigma \in S_m$

$$\log(B_{\sigma(1)}^t \cdots B_{\sigma(m)}^t) = t \sum_{i=1}^{m} C_{\sigma(i)} + \frac{t^2}{2} \sum_{i<j} [C_{\sigma(i)}, C_{\sigma(j)}]. \quad (6)$$

We show that we can obtain the desired matrices M_+ and M_- as $M_+ := B_{\sigma(1)}^t \cdots B_{\sigma(m)}^t$ and $M_- := B_{\sigma(m)}^t \cdots B_{\sigma(1)}^t$ for some permutation $\sigma \in S_m$ and large enough t.

Let $\sigma_0 \in S_m$ be the permutation that transposes 1 and 2. Write also $\mathrm{id} \in S_m$ for the identity permutation. Defining $\delta_\sigma := \sum_{i<j} [C_{\sigma(i)}, C_{\sigma(j)}]_{1,n}$, we have $\delta_{\mathrm{id}} - \delta_{\sigma_0} = 2[C_1, C_2]_{1,n} \neq 0$ since B_1, B_2 do not commute. Hence there exists $\sigma \in \{\mathrm{id}, \sigma_0\}$ with $\delta_\sigma \neq 0$. Defining the reverse permutation $\sigma' \in S_m$ by $\sigma'(i) = \sigma(m + 1 - i)$ for $i = 1, \ldots, m$, we moreover have $\delta_{\sigma'} = -\delta_\sigma$, and thus we may suppose that $\delta_\sigma > 0$ and $\delta_{\sigma'} < 0$. It remains to note, by inspection of (6), that for t sufficiently large, if $\delta_\sigma \neq 0$ then the sign of the $(1, n)$-entry of $\log(B_{\sigma(1)}^t \cdots B_{\sigma(m)}^t)$ is equal to the sign of δ_σ. But since $\log(B_{\sigma(1)}^t \cdots B_{\sigma(m)}^t)$ has zeros in all entries, except for the $(1, n)$-entry, this entry is in fact equal to the $(1, n)$-entry of $B_{\sigma(1)}^t \cdots B_{\sigma(m)}^t$.

So, under the assumption that $\mathcal{G}_0$ is not commutative we have shown that one can compute integers $p > 0$ and $q < 0$ such that $M_+ = (\mathbf{0}, \mathbf{0}, p)$ and $M_- = (\mathbf{0}, \mathbf{0}, q)$ are in $\mathcal{G}$. It follows that $\langle \mathcal{G} \rangle$ contains the group $\mathcal{N} = \{(\mathbf{0}, \mathbf{0}, c) \in \mathrm{H}(n, \mathbb{Z}) : c \equiv 0 \pmod{m}\}$, where $m = \gcd(p, q)$. Since

$$(\boldsymbol{a}, \boldsymbol{b}, c) \cdot (\mathbf{0}, \mathbf{0}, c') = (\mathbf{0}, \mathbf{0}, c') \cdot (\boldsymbol{a}, \boldsymbol{b}, c) = (\boldsymbol{a}, \boldsymbol{b}, c + c'),$$

we have the following equivalence for the target matrix $A = (\boldsymbol{a}, \boldsymbol{b}, c)$:

$$\begin{aligned} A = (\boldsymbol{a}, \boldsymbol{b}, c) \in \langle \mathcal{G} \rangle \quad & \text{iff} \quad \exists B \in \langle \mathcal{G} \rangle \text{ such that } AB^{-1} \in \mathcal{N} \\ & \text{iff} \quad \exists B \in \langle \mathcal{G} \rangle \text{ such that } B = (\boldsymbol{a}, \boldsymbol{b}, c') \text{ and } c' \equiv c \pmod{m}. \end{aligned}$$

To decide whether $\langle \mathcal{G} \rangle$ contains a matrix $B = (\boldsymbol{a}, \boldsymbol{b}, c')$ with $c' \equiv c \pmod{m}$, we will use register automata. Let $d = n-2$ and consider the following finite automaton with $2d$ registers:

$$\mathbf{Q} = (\{A_1, \ldots, A_k\}, S, R_1, \ldots, R_d, T_1, \ldots, T_d, s_0, \delta, F),$$

where the alphabet of $\mathbf{Q}$ is equal to the set of generator matrices $\mathcal{G} = \{A_1, \ldots, A_k\}$, and the set of states S is equal to

$$S = \{(s_1, \ldots, s_d, t_1, \ldots, t_d, u) \ : \ s_i, t_i, u \in \{0, \ldots, m-1\} \text{ for } i = 1, \ldots, d\}.$$

Intuitively, $(2d+1)$-tuples from S store the values of a vector $(\boldsymbol{a}, \boldsymbol{b}, c)$ modulo m, and the registers $R_1, \ldots, R_d$ and $T_1, \ldots, T_d$ store the values of $\boldsymbol{a}$ and $\boldsymbol{b}$, respectively.

The initial state of $\mathbf{Q}$ is $s_0 = (0, \ldots, 0)$, and the initial values of all the registers are zeros. The transition function δ is defined as follows. Suppose $\mathbf{Q}$ is in a state $(s_1, \ldots, s_d, t_1, \ldots, t_d, u)$, and the current values of R_i and T_i are r_i and t_i, respectively, for $i = 1, \ldots, d$. If $\mathbf{Q}$ reads a letter $A_\ell = (a_1^\ell, \ldots, a_d^\ell, b_1^\ell, \ldots, b_d^\ell, c^\ell)$, then it moves to the state $(s'_1, \ldots, s'_d, t'_1, \ldots, t'_d, u')$, where for each $i = 1, \ldots, d$:

$s'_i \equiv s_i + a_i^\ell \pmod{m}$ and $t'_i \equiv t_i + b_i^\ell \pmod{m}$,
$u' \equiv u + c^\ell + s_1 b_1^\ell + \cdots + s_d b_d^\ell \pmod{m}$.
Also, the new value of R_i is $r_i + a_i^\ell$ and the new value of T_i is $t_i + b_i^\ell$.

The set F of final states consists of one state that corresponds to the values of the target matrix $A = (\boldsymbol{a}, \boldsymbol{b}, c) = (a_1, \ldots, a_d, b_1, \ldots, b_d, c)$ modulo m, that is

$$F = \{(s_1, \ldots, s_d, t_1, \ldots, t_d, u) \ : \ s_i \equiv a_i, \ t_i \equiv b_i, \ u \equiv c \pmod{m} \ \text{ for } i = 1, \ldots, d\}.$$

The automaton $\mathbf{Q}$ accepts a word $w \in \{A_1, \cdots, A_k\}^*$ if after reading w it reaches the final state from F and the values of the registers $R_1, \ldots, R_d$ and $T_1, \ldots, T_d$ are equal to $a_1, \ldots, a_d$ and $b_1, \ldots, b_d$, respectively. By construction, the language of $\mathbf{Q}$ in non-empty if and only if $\langle \mathcal{G} \rangle$ contains a matrix $B = (\boldsymbol{a}, \boldsymbol{b}, c')$ with $c' \equiv c \pmod{m}$.

Note that after reading any letter the registers of $\mathbf{Q}$ are changed by constant values, and the transitions have no guards or zero checks. Let $\mathcal{S}$ be the set of values that the registers of $\mathbf{Q}$ can have when it reaches the final state. It is well-known that for a register automaton of this type the set $\mathcal{S}$ is effectively semilinear (see [22, 21] for details). In particular, we can decide whether $\mathcal{S}$ contains the vector $(\boldsymbol{a}, \boldsymbol{b})$, and so the emptiness problem for $\mathbf{Q}$ is decidable. Hence, in the case when $\mathcal{G}_0$ is not commutative the Membership Problem for $\mathrm{H}(n, \mathbb{Z})$ is decidable.

▶ **Corollary 8.** *The Membership Problem for* $\mathrm{H}(n, \mathbb{Q})$ *is decidable.*

Proof. Let $A_i = (\boldsymbol{a}_i, \boldsymbol{b}_i, c_i)$, for $i = 1, \ldots, k$, and $A = (\boldsymbol{a}, \boldsymbol{b}, c)$ be the given generators and the target matrix from $\mathrm{H}(n, \mathbb{Q})$. Let N be a natural number such that $A_i = (\frac{1}{N}\boldsymbol{a}'_i, \frac{1}{N}\boldsymbol{b}'_i, \frac{1}{N^2}c'_i)$, for $i = 1, \ldots, k$, and $A = (\frac{1}{N}\boldsymbol{a}', \frac{1}{N}\boldsymbol{b}', \frac{1}{N^2}c')$, where $\boldsymbol{a}'_i, \boldsymbol{b}'_i, c'_i$, for $i = 1, \ldots, k$, and $\boldsymbol{a}', \boldsymbol{b}', c'$ are integer vectors and numbers. It is easy to check that

$$(\tfrac{1}{N}\boldsymbol{x}, \tfrac{1}{N}\boldsymbol{y}, \tfrac{1}{N^2}c) \cdot (\tfrac{1}{N}\boldsymbol{x}', \tfrac{1}{N}\boldsymbol{y}', \tfrac{1}{N^2}c') = (\tfrac{1}{N}(\boldsymbol{x} + \boldsymbol{x}'), \tfrac{1}{N}(\boldsymbol{y} + \boldsymbol{y}'), \tfrac{1}{N^2}(c + c' + \boldsymbol{x}^\top \boldsymbol{y}')) .$$

Hence $A \in \langle A_1, \ldots, A_k \rangle$ iff $A' \in \langle A'_1, \ldots, A'_k \rangle$, where $A' = (\boldsymbol{a}', \boldsymbol{b}', c')$ and $A'_i = (\boldsymbol{a}'_i, \boldsymbol{b}'_i, c'_i)$, for $i = 1, \ldots, k$, are matrices with integer entries, that is, from $\mathrm{H}(n, \mathbb{Z})$. By Theorem 7 we can decide whether $A' \in \langle A'_1, \ldots, A'_k \rangle$. ◀

4 The Half-Space Reachability Problem for $\mathrm{GL}(2, \mathbb{Z})$

In this section we will show that the Half-Space Reachability Problem for $\mathrm{GL}(2, \mathbb{Z})$ is decidable (Theorem 17).

▶ **Definition 9.** *For an integer n, the sign of n as follows:* $\mathrm{sg}(n) = 1$ *if* $n > 0$, $\mathrm{sg}(n) = -1$ *if* $n < 0$, *and* $\mathrm{sg}(n) = *$ *if* $n = 0$.

For a matrix $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathbb{Z}^{2\times 2}$, *define* $\mathrm{sg}(A) := \begin{pmatrix} \mathrm{sg}(a) & \mathrm{sg}(b) \\ \mathrm{sg}(c) & \mathrm{sg}(d) \end{pmatrix}$.

If A and B are two expressions whose values are in the set $\{1, -1, *\}$, then the notation $A \simeq B$ means that $A = B$ or $A = *$ or $B = *$.

▶ **Proposition 10.** *Suppose w is a canonical word of the form $w = SR^{\alpha_1}SR^{\alpha_2}\cdots SR^{\alpha_n}$, where $\alpha_i \in \{1, 2\}$ for $i = 1, \ldots, n$. Then* $\mathrm{sg}(\varphi(w)) \simeq \begin{pmatrix} (-1)^n & (-1)^n \\ (-1)^n & (-1)^n \end{pmatrix}$.

Proof. The proof is by induction on n. For $n = 1$, we have

$$\mathrm{sg}(\varphi(SR)) = \mathrm{sg}\begin{pmatrix} -1 & -1 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} \mathrm{sg}(-1) & \mathrm{sg}(-1) \\ \mathrm{sg}(0) & \mathrm{sg}(-1) \end{pmatrix} \simeq \begin{pmatrix} -1 & -1 \\ -1 & -1 \end{pmatrix} \quad \text{and}$$

$$\mathrm{sg}(\varphi(SR^2)) = \mathrm{sg}\begin{pmatrix} -1 & 0 \\ -1 & -1 \end{pmatrix} = \begin{pmatrix} \mathrm{sg}(-1) & \mathrm{sg}(0) \\ \mathrm{sg}(-1) & \mathrm{sg}(-1) \end{pmatrix} \simeq \begin{pmatrix} -1 & -1 \\ -1 & -1 \end{pmatrix}$$

Suppose the statement of the proposition is true for $w = SR^{\alpha_1}SR^{\alpha_2}\cdots SR^{\alpha_n}$ and consider the words wSR and wSR^2. Assume that $\varphi(w) = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ and $\mathrm{sg}(\varphi(w)) = \begin{pmatrix} \mathrm{sg}(a) & \mathrm{sg}(b) \\ \mathrm{sg}(c) & \mathrm{sg}(d) \end{pmatrix} \simeq \begin{pmatrix} (-1)^n & (-1)^n \\ (-1)^n & (-1)^n \end{pmatrix}$. Then we have

$$\varphi(wSR) = \varphi(w)\varphi(SR) = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} -1 & -1 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} -a & -a-b \\ -c & -c-d \end{pmatrix} \quad \text{and}$$

$$\varphi(wSR^2) = \varphi(w)\varphi(SR^2) = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} -1 & 0 \\ -1 & -1 \end{pmatrix} = \begin{pmatrix} -a-b & -b \\ -c-d & -d \end{pmatrix}.$$

From these formulas it not hard to see that $\mathrm{sg}(\varphi(wSR)) \simeq \begin{pmatrix} (-1)^{n+1} & (-1)^{n+1} \\ (-1)^{n+1} & (-1)^{n+1} \end{pmatrix}$ and $\mathrm{sg}(\varphi(wSR^2)) \simeq \begin{pmatrix} (-1)^{n+1} & (-1)^{n+1} \\ (-1)^{n+1} & (-1)^{n+1} \end{pmatrix}$. ◀

▶ **Proposition 11.** *Let w be a canonical word of the form $w = S^\beta R^{\alpha_1} S R^{\alpha_2} \cdots S R^{\alpha_n} S^\varepsilon$, where $\beta, \varepsilon \in \{0,1\}$ and $\alpha_i \in \{1,2\}$, $i = 1, \ldots, n$. Then* $\operatorname{sg}(\varphi(w)) \simeq \begin{pmatrix} (-1)^n & (-1)^{n+\varepsilon} \\ (-1)^{n-1+\beta} & (-1)^{n-1+\beta+\varepsilon} \end{pmatrix}$.

Proof. First, consider the case when $\varepsilon = 0$. Suppose $\varphi(SR^{\alpha_1} SR^{\alpha_2} \cdots SR^{\alpha_n}) = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$. Then by Proposition 10 we have

$$\operatorname{sg}(\varphi(SR^{\alpha_1} SR^{\alpha_2} \cdots SR^{\alpha_n})) = \begin{pmatrix} \operatorname{sg}(a) & \operatorname{sg}(b) \\ \operatorname{sg}(c) & \operatorname{sg}(d) \end{pmatrix} \simeq \begin{pmatrix} (-1)^n & (-1)^n \\ (-1)^n & (-1)^n \end{pmatrix}.$$

On the other hand, $\varphi(R^{\alpha_1} SR^{\alpha_2} \cdots SR^{\alpha_n}) =$

$$= -\varphi(S)\varphi(SR^{\alpha_1} SR^{\alpha_2} \cdots SR^{\alpha_n}) = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} c & d \\ -a & -b \end{pmatrix}.$$

Hence $\operatorname{sg}(\varphi(R^{\alpha_1} SR^{\alpha_2} \cdots SR^{\alpha_n})) = \begin{pmatrix} \operatorname{sg}(c) & \operatorname{sg}(d) \\ \operatorname{sg}(-a) & \operatorname{sg}(-b) \end{pmatrix} \simeq \begin{pmatrix} (-1)^n & (-1)^n \\ (-1)^{n-1} & (-1)^{n-1} \end{pmatrix}$. Thus, for $\beta \in \{0,1\}$, we showed that

$$\operatorname{sg}(\varphi(S^\beta R^{\alpha_1} SR^{\alpha_2} \cdots SR^{\alpha_n})) \simeq \begin{pmatrix} (-1)^n & (-1)^n \\ (-1)^{n-1+\beta} & (-1)^{n-1+\beta} \end{pmatrix}. \tag{7}$$

Now assume $\varepsilon = 1$ and let $\varphi(S^\beta R^{\alpha_1} SR^{\alpha_2} \cdots SR^{\alpha_n}) = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$. Then

$$\begin{aligned} \varphi(S^\beta R^{\alpha_1} SR^{\alpha_2} \cdots SR^{\alpha_n} S) &= \varphi(S^\beta R^{\alpha_1} SR^{\alpha_2} \cdots SR^{\alpha_n})\varphi(S) \\ &= \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} b & -a \\ d & -c \end{pmatrix}. \end{aligned} \tag{8}$$

From equations (7) and (8) we obtain

$$\operatorname{sg}(\varphi(S^\beta R^{\alpha_1} SR^{\alpha_2} \cdots SR^{\alpha_n} S)) = \begin{pmatrix} \operatorname{sg}(b) & \operatorname{sg}(-a) \\ \operatorname{sg}(d) & \operatorname{sg}(-c) \end{pmatrix} \simeq \begin{pmatrix} (-1)^n & (-1)^{n+1} \\ (-1)^{n-1+\beta} & (-1)^{n-1+\beta+1} \end{pmatrix}. \tag{9}$$

Equations (7) and (9) imply that for any $\beta, \varepsilon \in \{0,1\}$ $\operatorname{sg}(\varphi(S^\beta R^{\alpha_1} SR^{\alpha_2} \cdots SR^{\alpha_n} S^\varepsilon)) \simeq \begin{pmatrix} (-1)^n & (-1)^{n+\varepsilon} \\ (-1)^{n-1+\beta} & (-1)^{n-1+\beta+\varepsilon} \end{pmatrix}$. ◀

From Proposition 11 and the equalities

$$\varphi(X) \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} -a & -b \\ -c & -d \end{pmatrix} \quad \text{and} \quad \varphi(N) \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a & b \\ -c & -d \end{pmatrix}$$

we obtain the following proposition.

▶ **Proposition 12.** *Let w be a canonical word of the form $w = N^\delta X^\gamma S^\beta R^{\alpha_1} SR^{\alpha_2} \cdots SR^{\alpha_n} S^\varepsilon$, where $\beta, \gamma, \delta, \varepsilon \in \{0,1\}$ and $\alpha_i \in \{1,2\}$ for $i = 1, \ldots, n$. Then*

$$\operatorname{sg}(\varphi(w)) \simeq \begin{pmatrix} (-1)^{n+\gamma} & (-1)^{n+\gamma+\varepsilon} \\ (-1)^{n-1+\beta+\gamma+\delta} & (-1)^{n-1+\beta+\gamma+\delta+\varepsilon} \end{pmatrix}.$$

▶ **Theorem 13.** *The set of matrices in* $\mathrm{GL}(2,\mathbb{Z})$ *whose particular entry is nonnegative forms a regular subset. In other words, for all* $i,j \in \{1,2\}$*, the following subset of* $\mathrm{GL}(2,\mathbb{Z})$ *is regular:*

$$\mathrm{Pos}_{ij} = \left\{ \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \in \mathrm{GL}(2,\mathbb{Z}) \ : \ a_{ij} \geq 0 \right\}.$$

Proof. Suppose $i = j = 2$ as other cases are similar. Let A be a matrix from $\mathrm{GL}(2,\mathbb{Z})$ and let $w = N^\delta X^\gamma S^\beta R^{\alpha_1} S R^{\alpha_2} \cdots S R^{\alpha_n} S^\varepsilon$, where $\beta,\gamma,\delta,\varepsilon \in \{0,1\}$ and $\alpha_i \in \{1,2\}$ for $i = 1,\ldots,n$, be a canonical word that represents A, that is, $A = \varphi(w)$. From Proposition 12 we see that $\mathrm{sg}(a_{22}) \simeq (-1)^{n-1+\beta+\gamma+\delta+\varepsilon}$. Hence

$$a_{22} \geq 0 \quad \text{if and only if} \quad n - 1 + \beta + \gamma + \delta + \varepsilon \equiv 0 \pmod 2. \tag{10}$$

To finish the proof, we note that the set of all canonical words is regular. Furthermore, given a canonical word of the form $w = N^\delta X^\gamma S^\beta R^{\alpha_1} S R^{\alpha_2} \cdots S R^{\alpha_n} S^\varepsilon$, a finite automaton can read off the values of $\beta,\gamma,\delta,\varepsilon$ and determine the parity of number n. From this data an automaton can decide whether $a_{22} \geq 0$ by the above mentioned equivalence (10). Hence the set of canonical words w such that $\varphi(w) \in \mathrm{Pos}_{22}$ can be recognised by a finite automaton. ◀

Next theorem was proved in [33].

▶ **Theorem 14.** *For every* $k \in \mathbb{Z}$*, the following subset of* $\mathrm{GL}(2,\mathbb{Z})$ *is regular:*

$$\mathrm{S}_{ij}(k) = \left\{ \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \in \mathrm{GL}(2,\mathbb{Z}) \ : \ a_{ij} = k \right\}.$$

As a corollary from Theorems 13 and 14 we obtain:

▶ **Theorem 15.** *For every* $k \in \mathbb{Z}$*, the following subsets of* $\mathrm{GL}(2,\mathbb{Z})$ *are regular:*

$$\mathrm{S}_{ij}(\geq k) = \left\{ \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \in \mathrm{GL}(2,\mathbb{Z}) \ : \ a_{ij} \geq k \right\} \quad \textit{and}$$

$$\mathrm{S}_{ij}(\leq k) = \left\{ \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \in \mathrm{GL}(2,\mathbb{Z}) \ : \ a_{ij} \leq k \right\}.$$

Proof. Since $\mathrm{S}_{ij}(\leq k)$ is the complement of $\mathrm{S}_{ij}(\geq k+1)$, it suffices to prove that the sets $\mathrm{S}_{ij}(\geq k)$ are regular.

If $k = 0$, then it follows from Theorem 13 that $\mathrm{S}_{ij}(\geq 0) = \mathrm{Pos}_{ij}$ is regular. Furthermore,

$$\mathrm{S}_{ij}(\geq k) = \mathrm{Pos}_{ij} \ \setminus \bigcup_{n=0}^{k-1} M_{ij}(n) \text{ if } k > 0 \quad \text{and} \quad \mathrm{S}_{ij}(\geq k) = \mathrm{Pos}_{ij} \ \cup \bigcup_{n=k}^{-1} M_{ij}(n) \text{ if } k < 0.$$

Since by Corollary 6 regular subsets of $\mathrm{GL}(2,\mathbb{Z})$ are closed under Boolean operations, we conclude that $\mathrm{S}_{ij}(\geq k)$ is a regular set for any $k \in \mathbb{Z}$. ◀

▶ **Theorem 16.** *Let* $\lambda \in \mathbb{Q}$ *and* $\boldsymbol{u},\boldsymbol{v} \in \mathbb{Q} \times \mathbb{Q}$*. Then the set* $\mathcal{S}(\boldsymbol{u},\boldsymbol{v},\lambda) = \{\, M \in \mathrm{GL}(2,\mathbb{Z}) \ : \ \boldsymbol{u}^\top M \boldsymbol{v} \geq \lambda \,\}$ *is a regular subset of* $\mathrm{GL}(2,\mathbb{Z})$*.*

Proof. Note that if $\boldsymbol{u} = \boldsymbol{0}$ or $\boldsymbol{v} = \boldsymbol{0}$, then $\boldsymbol{u}^\top M \boldsymbol{v} = 0$. In this case $\mathcal{S}(\boldsymbol{u},\boldsymbol{v},\lambda)$ equals either the empty set or $\mathrm{GL}(2,\mathbb{Z})$, both of which are regular subsets. Hence we will assume that both $\boldsymbol{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$ and $\boldsymbol{v} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$ are nonzero vectors. By multiplying the inequality $\boldsymbol{u}^\top M \boldsymbol{v} \geq \lambda$

by the least common multiple of the denominators of u_1, u_2, v_1, v_2, we can assume that $\boldsymbol{u}$ and $\boldsymbol{v}$ have integer coefficients. Furthermore, we can divide $\boldsymbol{u}^\top M\boldsymbol{v} \geq \lambda$ by $\gcd(u_1, u_2)$ and $\gcd(v_1, v_2)$ and so assume from now on that $\gcd(u_1, u_2) = \gcd(v_1, v_2) = 1$.

Finally, note that the inequality $\boldsymbol{u}^\top M\boldsymbol{v} \geq \lambda$ is equivalent to $\boldsymbol{u}^\top M\boldsymbol{v} \geq \lceil\lambda\rceil$, where $\lceil\lambda\rceil = \min\{n \in \mathbb{Z} \ : \ n \geq \lambda\}$. So, we can assume that λ is also an integer number.

Since $\gcd(u_1, u_2) = \gcd(v_1, v_2) = 1$, there are integers s_1, s_2, t_1, t_2 such that $s_1u_1 + s_2u_2 = 1$ and $t_1v_1 + t_2v_2 = 1$. Hence the matrices $A = \begin{pmatrix} u_1 & -s_2 \\ u_2 & s_1 \end{pmatrix}$ and $B = \begin{pmatrix} v_1 & -t_2 \\ v_2 & t_1 \end{pmatrix}$ belong to $\mathrm{GL}(2, \mathbb{Z})$, and we have that $\boldsymbol{u} = A\boldsymbol{e}_1$ and $\boldsymbol{v} = B\boldsymbol{e}_1$. Therefore, the inequality $\boldsymbol{u}^\top M\boldsymbol{v} \geq \lambda$ is equivalent to $\boldsymbol{e}_1^\top A^\top MB\boldsymbol{e}_1 \geq \lambda$. In other words,

$$M \in \mathcal{S}(\boldsymbol{u}, \boldsymbol{v}, \lambda) \iff A^\top MB \in \mathrm{S}_{11}(\geq\lambda) \iff M \in (A^\top)^{-1} \cdot \mathrm{S}_{11}(\geq\lambda) \cdot B^{-1}.$$

By Theorem 15, $\mathrm{S}_{11}(\geq \lambda)$ is a regular subset of $\mathrm{GL}(2, \mathbb{Z})$. Let L be a regular language and let w_1, w_2 be canonical words such that $\varphi(L) = \mathrm{S}_{11}(\geq \lambda)$ and $\varphi(w_1) = (A^\top)^{-1}$ and $\varphi(w_2) = B^{-1}$. Then $\{w_1\} \cdot L \cdot \{w_2\}$ is a regular language such that $\varphi(\{w_1\} \cdot L \cdot \{w_2\}) = (A^\top)^{-1} \cdot \mathrm{S}_{11}(\geq\lambda) \cdot B^{-1} = \mathcal{S}(\boldsymbol{u}, \boldsymbol{v}, \lambda)$. ◀

▶ **Theorem 17.** *The Half-Space Reachability Problem for* $\mathrm{GL}(2, \mathbb{Z})$ *is decidable.*

Proof. Let $\mathcal{G} = \{A_1, \ldots, A_k\}$ be a finite collection of matrices from $\mathrm{GL}(2, \mathbb{Z})$, λ be a rational number and $\boldsymbol{u}, \boldsymbol{v}$ be vectors from $\mathbb{Q}^2$. Define $\mathcal{S}(\boldsymbol{u}, \boldsymbol{v}, \lambda) := \{\, M \in \mathrm{GL}(2, \mathbb{Z}) \ : \ \boldsymbol{u}^\top M\boldsymbol{v} \geq \lambda \,\}$. By Theorem 16, $\mathcal{S}(\boldsymbol{u}, \boldsymbol{v}, \lambda)$ is a regular subset of $\mathrm{GL}(2, \mathbb{Z})$. Let $L_{\mathcal{S}}$ be a regular language such that $\mathcal{S}(\boldsymbol{u}, \boldsymbol{v}, \lambda) = \varphi(L_{\mathcal{S}})$. It is not hard to see that the semigroup $\langle\mathcal{G}\rangle$ is also a regular subset. Indeed, consider a regular language $L_{\mathcal{G}} = (w_1 \cup \cdots \cup w_k)^+$, where $w_1, \ldots, w_k$ are canonical words that correspond to the matrices $A_1, \ldots, A_k$, respectively. Then $\langle\mathcal{G}\rangle = \varphi(L_{\mathcal{G}})$.

By Corollary 6, we can algorithmically construct a regular language $L^\cap$ such that $\varphi(L^\cap) = \varphi(L_{\mathcal{S}}) \cap \varphi(L_{\mathcal{G}}) = \mathcal{S}(\boldsymbol{u}, \boldsymbol{v}, \lambda) \cap \langle\mathcal{G}\rangle$. Now we have the following equivalence:

$$\text{there is } M \in \langle\mathcal{G}\rangle \text{ such that } \boldsymbol{u}^\top M\boldsymbol{v} \geq \lambda \quad \text{iff} \quad \mathcal{S}(\boldsymbol{u}, \boldsymbol{v}, \lambda) \cap \langle\mathcal{G}\rangle = \varphi(L^\cap) \neq \emptyset.$$

The last condition is equivalent to $L^\cap \neq \emptyset$. Therefore, we reduced the Half-Space Reachability Problem for $\mathrm{GL}(2, \mathbb{Z})$ to the emptiness problem for regular languages. ◀

5 The Half-Space Reachability Problem for the Heisenberg Group

▶ **Definition 18.** *Let* $\mathcal{S} := B_1, \ldots, B_m$ *be a sequence in* $\mathrm{H}(n, \mathbb{Q})$ *and* A *a particular matrix in* $\mathrm{H}(n, \mathbb{Q})$*. A pair* $i, j \in \{1 \ldots, m\}$ *with* $i \leq j$ *is called an* A-block *of* $\mathcal{S}$ *if*

1. $B_k = A$ *for all* $k \in \{i, \ldots, j\}$,
2. *either* $i = 1$ *or* $B_{i-1} \neq A$,
3. *either* $j = m$ *or* $B_{j+1} \neq A$.

We say that $\mathcal{S}$ *is* pure *if it has at most one* A*-block for every matrix* A*.*

Given a sequence $\mathcal{S} = B_1, \ldots, B_m \in \mathrm{H}(n, \mathbb{Q})$, define $C_i := \log(B_i)$ for $i = 1, \ldots, m$, $\Delta(\mathcal{S}) := \sum_{1 \leq i < j \leq m} [C_i, C_j]$, and $\delta(\mathcal{S}) := \Delta(\mathcal{S})_{1,n}$. Recall that using the Baker-Campbell-Hausdorff formula (3) we can express the product of the sequence $\mathcal{S}$ as follows

$$B_1 \cdots B_m = \exp\Big(\sum_{i=1}^{m} C_i + \frac{1}{2} \underbrace{\sum_{1 \leq i < j \leq m} [C_i, C_j]}_{\Delta(\mathcal{S})}\Big) \tag{11}$$

▶ **Proposition 19.** *For any sequence of matrices $\mathcal{S} = B_1, \ldots, B_m \in \mathrm{H}(n, \mathbb{Q})$, there is a permutation $\pi \in S_m$ such that sequence $\mathcal{S}' := B_{\pi(1)}, \ldots, B_{\pi(m)}$ is pure and $\delta(\mathcal{S}) \leq \delta(\mathcal{S}')$.*

The proof of Proposition 19 can be found in the full version [11].

▶ **Theorem 20.** *The Half-Space Reachability Problem for $\mathrm{H}(n, \mathbb{Q})$ is decidable.*

Proof. Consider an instance of the Half-Space Reachability Problem, given by a finite set $\mathcal{G} = \{A_1, \ldots, A_k\} \subseteq \mathrm{H}(n, \mathbb{Q})$ of generators, vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{Q}^n$ and a scalar $\lambda \in \mathbb{Q}$.

Given a sequence $\mathcal{S} = B_1, \ldots, B_m$ of elements of $\mathcal{G}$ and a permutation $\sigma \in \mathrm{Sym}_m$, define $\mathcal{S}_\sigma = B_{\sigma(1)}, \ldots, B_{\sigma(m)}$. It follows from Equation (11) that the entries of the product $B_{\sigma(1)} \cdots B_{\sigma(m)}$ do not depend on the choice of $\sigma \in \mathrm{Sym}_m$, except for the $(1, n)$-entry which is equal to $\frac{1}{2}\Delta(\mathcal{S}_\sigma)_{1,n}$ plus a constant that also does not depend on σ. So, the permutation σ that maximises $\boldsymbol{u}^\top B_{\sigma(1)} \cdots B_{\sigma(m)} \boldsymbol{v}$ is the same which maximises or minimises $\Delta(\mathcal{S}_\sigma)_{1,n}$ depending on the sign of the coefficient at $\Delta(\mathcal{S}_\sigma)_{1,n}$ in the expression $\boldsymbol{u}^\top \Delta(\mathcal{S}_\sigma) \boldsymbol{v}$, namely, on the sign of $\boldsymbol{u}_1 \boldsymbol{v}_n$. By Proposition 19 we may assume without loss of generality that the optimal permutation σ is such that $\mathcal{S}_\sigma$ is pure.

By the reasoning above, to decide the given instance of the Half-Space Reachability Problem it suffices to restrict attention to pure sequences of generators. Equivalently we must decide whether there exist nonnegative integers $n_1, \ldots, n_k$ and a permutation $\sigma \in \mathrm{Sym}_k$ such that $\boldsymbol{u}^\top A_{\sigma(1)}^{n_1} \cdots A_{\sigma(k)}^{n_k} \boldsymbol{v} \geq \lambda$. Write $C_i = \log A_i$ for $i = 1, \ldots, k$. Then

$$\begin{aligned} \boldsymbol{u}^\top A_{\sigma(1)}^{n_1} \cdots A_{\sigma(k)}^{n_k} \boldsymbol{v} &= \boldsymbol{u}^\top \exp\left(\sum_{i=1}^{k} n_i C_{\sigma(i)} + \frac{1}{2} \sum_{i<j} n_i n_j [C_{\sigma(i)}, C_{\sigma(j)}] \right) \boldsymbol{v} \\ &= Q(n_1, \ldots, n_k) \end{aligned}$$

for some quadratic polynomial $Q(x_1, \ldots, x_k)$ with rational coefficients.

In the work of Grunewald and Segal [14] an algorithm is given for solving the following problem: does there exist integers $n_1, \ldots, n_k$ that satisfy a given quadratic equation $Q(n_1, \ldots, n_k) = 0$ (with rational coefficients) and a finite number of linear inequalities on $n_1, \ldots, n_k$ (also with rational coefficients).

By introducing a "dummy" variable we can use the Grunewald and Segal algorithm to decide whether $Q(n_1, \ldots, n_k) \geq \lambda$ for some nonnegative integers $n_1, \ldots, n_k$. Hence the Half-Space Reachability Problem for $\mathrm{H}(n, \mathbb{Q})$ is decidable. ◀

References

1. László Babai. Trading Group Theory for Randomness. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 421–429, 1985.
2. László Babai, Robert Beals, Jin-yi Cai, Gábor Ivanyos, and Eugene M. Luks. Multiplicative Equations over Commuting Matrices. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '96, pages 498–507, Philadelphia, PA, USA, 1996. Society for Industrial and Applied Mathematics.
3. Robert Beals. Algorithms for Matrix Groups and the Tits Alternative. *J. Comput. Syst. Sci.*, 58(2):260–279, 1999.
4. Paul Bell, Mika Hirvensalo, and Igor Potapov. The Identity Problem for Matrix Semigroups in $\mathrm{SL}(2, \mathbb{Z})$ is NP-complete. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2017.
5. Paul C. Bell and Igor Potapov. On the Undecidability of the Identity Correspondence Problem and its Applications for Word and Matrix Semigroups. *Int. J. Found. Comput. Sci.*, 21(6):963–978, 2010.

6 Paul C. Bell and Igor Potapov. On the Computational Complexity of Matrix Semigroup Problems. *Fundam. Inf.*, 116(1-4):1–13, 2012.

7 V. Blondel, E. Jeandel, P. Koiran, and N. Portier. Decidable and Undecidable Problems about Quantum Automata. *SIAM J. Comput.*, 34(6):1464–1473, 2005. doi:10.1137/S0097539703425861.

8 Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

9 Julien Cassaigne, Vesa Halava, Tero Harju, and François Nicolas. Tighter Undecidability Bounds for Matrix Mortality, Zero-in-the-Corner Problems, and More. *CoRR*, abs/1404.0644, 2014. arXiv:1404.0644.

10 Christian Choffrut and Juhani Karhumäki. Some decision problems on integer matrices. *RAIRO-Theor. Inf. Appl.*, 39(1):125–131, 2005.

11 Thomas Colcombet, Joël Ouaknine, Pavel Semukhin, and James Worrell. On reachability problems for low dimensional matrix semigroups. *CoRR*, abs/1902.09597, 2019. arXiv:1902.09597.

12 H. Derksen, E. Jeandel, and P. Koiran. Quantum automata and algebraic groups. *J. Symb. Comput.*, 39(3-4):357–371, 2005.

13 Fritz J. Grunewald and Daniel Segal. How to solve a quadratic equation in integers. *Mathematical Proceedings of the Cambridge Philosophical Society*, 89(1):1–5, 1981.

14 Fritz J. Grunewald and Daniel Segal. On the integer solutions of quadratic equations. *J. Reine Angew. Math.*, 569:13–45, 2004.

15 Yuri Gurevich and Paul Schupp. Membership Problem for the Modular Group. *SIAM J. Comput.*, 37(2):425–459, May 2007.

16 V. Halava and M. Hirvensalo. Improved matrix pair undecidability results. *Acta Informatica*, 44(3-4):191–205, 2007.

17 Vesa Halava, Tero Harju, and Mika Hirvensalo. Undecidability Bounds for Integer Matrices Using Claus Instances. *Int. J. Found. Comput. Sci.*, 18(5):931–948, 2007.

18 B. Hall. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*, volume 222 of *Graduate Texts in Mathematics*. Springer International Publishing, 2015.

19 Ehud Hrushovski, Joël Ouaknine, Amaury Pouly, and James Worrell. Polynomial Invariants for Affine Programs. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 530–539, 2018.

20 Ravindran Kannan and Richard J. Lipton. Polynomial-time algorithm for the orbit problem. *J. ACM*, 33(4):808–821, 1986.

21 Felix Klaedtke and Harald Rueß. Parikh Automata and Monadic Second-Order Logics with Linear Cardinality Constraints. Technical report, Albert-Ludwigs-Universität Freiburg, 2002.

22 Felix Klaedtke and Harald Rueß. Monadic Second-Order Logics with Cardinalities. In *Automata, Languages and Programming, 30th International Colloquium, ICALP*, pages 681–696, 2003.

23 Sang-Ki Ko, Reino Niskanen, and Igor Potapov. On the Identity Problem for the Special Linear Group and the Heisenberg Group. In *45th International Colloquium on Automata, Languages, and Programming, ICALP*, pages 132:1–132:15, 2018.

24 Daniel König, Markus Lohrey, and Georg Zetzsche. Knapsack and subset sum problems in nilpotent, polycyclic, and co-context-free groups. *CoRR*, abs/1507.05145, 2015. arXiv:1507.05145.

25 Roger C. Lyndon and Paul E. Schupp. *Combinatorial group theory*. Springer-Verlag, Berlin-New York, 1977. Ergebnisse der Mathematik und ihrer Grenzgebiete, Band 89.

26 Wilhelm Magnus, Abraham Karrass, and Donald Solitar. *Combinatorial group theory*. Dover Publications, Inc., New York, revised edition, 1976.

27 A. Mandel and I. Simon. On Finite Semigroups of Matrices. *Theor. Comput. Sci.*, 5(2):101–111, 1977.

28 A. Markov. On certain insoluble problems concerning matrices. *Doklady Akad. Nauk SSSR*, 57(6):539–542, June 1947.

29 K. A. Mihailova. The occurrence problem for a direct product of groups. *Dokl. Akad. Nauk*, 119:1103–1105, 1958.

30 Joël Ouaknine and James Worrell. Positivity Problems for Low-Order Linear Recurrence Sequences. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 366–379, 2014.

31 Michael S. Paterson. Unsolvability in 3×3 matrices. *Studies in Appl. Math.*, 49:105–107, 1970.

32 Igor Potapov and Pavel Semukhin. Decidability of the Membership Problem for 2×2 integer matrices. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 170–186, 2017.

33 Igor Potapov and Pavel Semukhin. Membership Problem in GL(2, Z) Extended by Singular Matrices. In *42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017, August 21-25, 2017 - Aalborg, Denmark*, pages 44:1–44:13, 2017.

34 Igor Potapov and Pavel Semukhin. Vector and scalar reachability problems in $\mathrm{SL}(2, \mathbb{Z})$. *J. Comput. Syst. Sci.*, 100:30–43, 2019.

35 Robert A. Rankin. *Modular forms and functions*. Cambridge University Press, Cambridge-New York-Melbourne, 1977.

36 Grzegorz Rozenberg and Arto Salomaa. *Cornerstones of undecidability*. Prentice Hall International Series in Computer Science. Prentice Hall, 1994.

37 Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

38 Charles Sims. *Computation with Finitely Presented Groups*. Cambridge University Press, 1994.

Independent Sets in Vertex-Arrival Streams

Graham Cormode
University of Warwick, UK
g.cormode@warwick.ac.uk

Jacques Dark
University of Warwick, UK
j.dark@warwick.ac.uk

Christian Konrad
University of Bristol, UK
christian.konrad@bristol.ac.uk

Abstract

We consider the maximal and maximum independent set problems in three models of graph streams:

- In the edge model we see a stream of edges which collectively define a graph; this model is well-studied for a variety of problems. We show that the space complexity for a one-pass streaming algorithm to find a maximal independent set is quadratic (i.e. we must store all edges). We further show that it is not much easier if we only require approximate maximality. This contrasts strongly with the other two vertex-based models, where one can greedily find an exact solution in only the space needed to store the independent set.
- In the "explicit" vertex model, the input stream is a sequence of vertices making up the graph. Every vertex arrives along with its incident edges that connect to previously arrived vertices. Various graph problems require substantially less space to solve in this setting than in edge-arrival streams. We show that every one-pass c-approximation streaming algorithm for maximum independent set (MIS) on explicit vertex streams requires $\Omega(\frac{n^2}{c^6})$ bits of space, where n is the number of vertices of the input graph. It is already known that $\tilde{\Theta}(\frac{n^2}{c^2})$ bits of space are necessary and sufficient in the edge arrival model (Halldórsson *et al.* 2012), thus the MIS problem is not significantly easier to solve under the explicit vertex arrival order assumption. Our result is proved via a reduction from a new multi-party communication problem closely related to pointer jumping.
- In the "implicit" vertex model, the input stream consists of a sequence of objects, one per vertex. The algorithm is equipped with a function that maps pairs of objects to the presence or absence of edges, thus defining the graph. This model captures, for example, geometric intersection graphs such as unit disc graphs. Our final set of results consists of several improved upper and lower bounds for interval and square intersection graphs, in both explicit and implicit streams. In particular, we show a gap between the hardness of the explicit and implicit vertex models for interval graphs.

2012 ACM Subject Classification Theory of computation → Lower bounds and information complexity; Theory of computation → Streaming models

Keywords and phrases streaming algorithms, independent set size, lower bounds

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.45

Category Track A: Algorithms, Complexity and Games

Related Version The full version of this paper is available at: `https://arxiv.org/abs/1807.08331`.

Funding *Graham Cormode*: Supported by European Research Council grant ERC-2014-CoG 647557.
Jacques Dark: Supported by an EMEA Microsoft Research scholarship. Part of the work was done while J.D. was at the Alan Turing Institute, under EPSRC grant EP/N510129/1.
Christian Konrad: C.K. carried out most work on this paper while being at the University of Warwick. He was supported by the Centre for Discrete Mathematics and its Applications (DIMAP) at Warwick University and by EPSRC award EP/N011163/1.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 45; pp. 45:1–45:14

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

1 Introduction

The streaming model supposes that, rather than being loaded into memory all at once, the input is received piece-by-piece over a period of time. Only a sublinear amount of memory (in the input size) is made available, preventing any algorithm from "seeing" even a constant fraction of the whole input at once.

In graph streams (see [19] for an excellent survey), we distinguish between the "edge-arrival" model, where the stream consists of individual edges arriving in any order, and the "vertex-arrival" model, where the stream consists of batches of edges incident to a particular vertex – as each vertex "arrives" we are given all the edges from the new vertex to previously arrived vertices. We will shorten the names to edge streams and vertex streams, respectively. Problems are always at least as hard on edge streams as on vertex streams (as any vertex stream is also a valid edge stream).

There is a further variant which we will call "implicit" vertex streams (as opposed to the normal explicit representation). In this model, the stream consists of a series of small (polylog(n)-sized) identifiers – one per vertex. We are additionally provided with some symmetric function or oracle which maps a pair of identifiers to a Boolean output indicating whether the two vertices are connected or not. This implicitly defines a graph over the list of identifiers received. Geometric intersection graphs, received as a stream of geometric objects, are the most natural members of this class. For example, a unit interval intersection graph can be given by a set of points in $\mathbb{R}$. Then a pair of vertices x, y are adjacent if and only if $|x - y| \leq 1$.

Explicit and implicit vertex streams are closely related but distinct, with neither being strictly "harder" than the other. For example: it is easy to count exactly the number of edges in $\tilde{O}(1)$ space[1] for an explicit vertex stream, however, doing so for an implicit stream requires linear space – otherwise we cannot hope to know how many edges are incident to the final vertex. On the other hand: implicit vertex streams can be stored entirely in $\tilde{O}(n)$ space, whereas explicit vertex streams require $\Omega(n^2)$ space to store the full structure.

Maximum Independent Set (MIS) is an important problem on graphs. The task is to find a largest subset of vertices which have no edges between them. The size of a MIS in a graph G is denoted $\alpha(G)$, the independence number of G. Unfortunately, it is NP-hard to find a maximum independent set in a general graph [16], and even hard to approximate within a factor of $n^{1-\epsilon}$, for any $\epsilon > 0$ [20]. It is also known to be hard in the edge-arrival streaming model: Halldórsson *et al.* [15] showed that space $\tilde{\Theta}(\frac{n^2}{c^2})$ is necessary (and sufficient) for computing a c-approximation on an n-vertex graph, despite being allowed unlimited computation.

Our Results. In this paper, we study the hardness of approximate MIS in the explicit and implicit vertex streaming models. Since many problems are significantly easier to solve in vertex streams than in edge streams, we ask whether this is also the case for the MIS problem. As our main result, we answer this question in the negative:

▶ **Theorem 1.** *Any constant error one-pass c-approximation streaming algorithm for* MIS *(or the size of a* MIS*) in the explicit vertex stream model requires $\Omega\left(\frac{n^2}{c^6}\right)$ bits of space.*

[1] All space bounds in this paper are given as number of bits. We use $\tilde{O}$, $\tilde{\Theta}$, and $\tilde{\Omega}$ to mean O, Θ, and Ω (respectively) with log factors suppressed.

Space Bound	Approx. MIS $\tilde{O}(\alpha(G))$	Approx. $\alpha(\mathbf{G})$ poly$(\log n)$	$\Omega(n)$
Unit Interval	2 (Greedy alg.)	$O\left(\frac{\log^2 n}{\log\log n}\right)$ [9]	$< 5/3$

Figure 1 Approximation factors for explicit vertex streams.

Space Bound	Approx. MIS $\tilde{O}(\alpha(G))$	Approx. $\alpha(\mathbf{G})$ poly$(\log n, \epsilon^{-1})$	$\Omega(n)$
Unit Interval	3/2 [11]	$3/2+\epsilon$ [7]	$< 3/2$ [11]
Interval	2 [11]	$2+\epsilon$ [7]	< 2 [11]
Unit Square	3	$3+\epsilon$	$< 5/2$

Figure 2 Approximation factors for implicit vertex streams. The first column concerns algorithms that output independent sets themselves, while the second columns concerns algorithms that output estimations of the maximum independent set size. Results from this paper are highlighted.

Our lower bound also holds for the Minimum Vertex Coloring (MVC) problem, where the objective is to color the vertices of the input graph such that adjacent vertices have different colors, using the fewest colors possible. This quantity is the chromatic number and denoted by $\chi(G)$. Our result is the first lower bound known for this problem, even for edge streams (in particular, the work by Halldórsson *et al.* [15] does not imply such a result).

Next, we show that the situation is very different for the related *maximal* independent set problem, where we need to find a subset of non-adjacent vertices that cannot be enlarged. While it is easy to maintain a maximal independent set in vertex arrival streams (both explicit and implicit) using space $\tilde{O}(\alpha(G)) = \tilde{O}(n)$, we prove that $\Omega(n^2)$ space is required in the edge-arrival model. We further show that even if we relax the maximality constraint to *approximate maximality* and allow for a slightly sublinear number of vertices that are not adjacent to vertices of the independent set then space $\Omega(n^{2-o(1)})$ is still required.

Finally, we show various improved upper and lower bounds for certain geometric intersection graph classes in both vertex streaming models: unit interval intersection graphs given as explicit vertex streams require $\Omega(n)$ space to get a better than $\frac{5}{3}$-approximation to $\alpha(G)$, making them harder than their implicit vertex stream equivalents; and we can 3-approximate MIS for a stream of unit squares in the plane using $\tilde{O}(\alpha(G))$ space, but achieving better than a $\frac{5}{2}$-approximation to $\alpha(G)$ requires $\Omega(n)$ space. Figures 2 and 1 shows these results in the context of previously known bounds.

Techniques. Halldórsson *et al.* [15] proved their space lower bound for MIS in edge streams via the *one-way two-party communication framework*. Two parties, denoted Alice and Bob, each hold a subset of the edges of the input graph. Alice sends a single message to Bob, who, upon receipt, outputs a large independent set. Via a reduction from a well-known communication problem, they showed that if Bob outputs a c-approximate MIS then Alice must send a message of size $\tilde{\Omega}(\frac{n^2}{c^2})$ to Bob. A common reduction then implies that the same lower bound holds for the space complexity of one-pass streaming algorithms.

Proving a similar result in vertex streams is significantly harder since the two-party communication abstraction cannot yield the desired result. When partitioning the vertices of a vertex stream between Alice and Bob both parties hold *vertex-induced subgraphs*, as opposed to the *spanning subgraphs* obtained when partitioning the edges of an edge stream. Since an independent set in an induced subgraph is also an independent set in the whole

graph, and since either Alice or Bob holds at least half the vertices of any MIS, one of them must already know a 2-approximation to the MIS. Using the same reasoning, it is trivial to compute a p-approximation in the one-way p-party communication setting. To obtain our lower bound result, we therefore need to consider multi-party communication with $\Omega(c)$ parties.

To this end, we define a new k-party communication problem denoted $\textsc{Chain}_k$ – which can be seen as chaining together multiple two-party instances of the well-known Index problem (see Definition 2) that are guaranteed to have the same answer. We first give a $\Omega(\frac{n}{k^2})$ lower bound for $\textsc{Chain}_k$ by showing a reduction from a multi-party pointer jumping problem [8]. We then improve this lower bound to $\Omega(\frac{n}{k})$ for k up to $\tilde{O}\left(\sqrt[4]{n}\right)$ using the same party elimination techniques described in [8]. The actual reduction from $\textsc{Chain}_k$ to MIS relies on an involved graph construction using erasure codes based on affine planes.

Our lower bound for the computation of a maximal independent set in edge streams is obtained via a reduction from the Index problem in the two-party communication framework. This construction is then extended to yield results for approximate maximality via a construction involving Ruzsa-Szemerédi graphs.

Our upper bound results on 2D geometric intersection graphs are obtained by generalizing 1D bounds, with more work to cover the increased number of cases that occur in 2D. The lower bounds involve intricate packing arguments to show that knowledge of $\alpha(G)$ can be used to recover encoded information, which is used in conjunction with our multiparty $\textsc{Chain}_k$ problem to demonstrate approximation hardness.

Further Related Work. Grouping the three streaming models:

- *Edge Streams.* As previously mentioned, Halldórsson *et al.* [15] showed that for general graphs in the edge-arrival model $\tilde{\Omega}\left(\frac{n^2}{c^2}\right)$ space is required to obtain a c-approximation to the maximum independent set size (or maximum clique size). A corresponding $\tilde{O}\left(\frac{n^2}{c^2}\right)$ space random sampling algorithm shows that this is tight up to logarithmic factors. Braverman *et al.* [6] showed that space $\Omega(\frac{m}{c^2})$ is needed, even if $c = o(\log n)$, where m is the number of edges of the input graph, though this bound only holds for small m.
- *Explicit Vertex Streams.* The work of Halldórsson *et al.* [13] gives an $O(n \log n)$ space streaming algorithm which can find an independent set of expected size at least $\beta(G) = \sum_{v \in V} \frac{1}{\deg(v)+1}$. On general graphs, this only gives a $\Theta(n)$-approximation, but for polynomially bounded independence graphs, this gives a polylog(n)-approximation [14]. In our prior work, we showed how to return an estimate $\gamma \in \Omega\left(\frac{\beta(G)}{\log n}\right)$ with $\gamma \leq \alpha(G)$ from an explicit vertex arrival stream using only $O(\log^3 n)$ space [9]. This result, for example, gives a $O(\frac{\log^2 n}{\log \log n})$-approximation on unit interval graphs (see Figure 1). However, the technique samples vertices based on their degree and does not extend to implicit vertex streams.

  Braverman *et al.* [6] showed that in a variant of the vertex arrival model, where every vertex arrives together with *all* its incident edges (as opposed to only the edges incident to previously arrived vertices), space $\Omega(\frac{m}{c^3})$ is required for computing a c-approximate MIS. In their construction the input graph has $\Theta(nc)$ edges, which thus yields a lower bound of $\Omega(\frac{n}{c^2})$. Observe that our lower bound for explicit vertex streams is $\Omega(\frac{n^2}{c^6})$, a quadratic improvement for constant c.
- *Implicit Vertex Streams.* In [11], it was shown that it is possible to $\frac{3}{2}$-approximate MIS for the intersection graph of a unit interval stream using $\tilde{O}(\alpha(G))$ space. In the same space, a 2-approximation is possible for arbitrary interval streams. Both are shown to

be tight: any $(\frac{3}{2}-\epsilon)$-approximation for unit intervals, or $(2-\epsilon)$ for general intervals, requires $\Omega(n)$ space. By clever use of sampling, the result can be adapted to provide an approximation of $\alpha(G)$ of $\frac{3}{2}+\epsilon$ for unit intervals and $2+\epsilon$ for general intervals with only polylog(n, ϵ^{-1}) space [7].

Concurrent Work. Independently of, and concurrently with, an earlier version of this paper [10, v1], Assadi *et al.* [3] also gave an $\Omega(n^2)$ lower bound for maximal independent set in edge streams using a similar construction.

Outline. We present our main result, the lower bound for MIS in vertex streams, in Section 2. Our lower bounds for maximal and approximately maximal independent sets in edge streams are given in Section 3. Section 4 covers our results on interval and square graphs, and we give a brief conclusion in Section 5.

2 Maximum Independent Set in Explicit Vertex Streams

We first introduce and show the hardness of a "chained index" problem, which we then use to show the hardness of approximating the size $\alpha(G)$ – and hence also for finding an approximate MIS.

2.1 Chained Index Communication Problem

We define a multi-party communication problem $\textsc{Chain}_k$, which allows us to prove new lower bounds on several streaming problems. The problem is closely related to pointer jumping and generalizes the classic two-party Index communication problem to more parties by "chaining" together multiple instances which have the same answer but are otherwise independent. Index is defined as follows:

▶ **Definition 2.** *In the two-party communication problem* Index*, Alice holds an n-bit string $X \in \{0,1\}^n$ and Bob holds an index $\sigma \in [n]$. Alice sends a single message to Bob who, upon receipt, outputs X_σ.*

It is well known that Alice essentially needs to send all n bits to Bob (see [18]):

▶ **Theorem 3.** *The randomized constant error communication complexity of* Index *is $\Omega(n)$.*

In $\textsc{Chain}_k$, each party (except the last) holds a binary vector that contains a special bit which is the answer to the instance. Each party (except the first) knows where the answer bit is located in the previous party's vector. Communication is one-way and private, with each player receiving a message from the previous player and then sending a message to the next player. Formally:

▶ **Definition 4.** *The k-party chained index problem $\textsc{Chain}_k$ consists of $(k-1)$ n-bit binary vectors $\{X^{(i)}\}_{i=1}^{k-1}$, along with corresponding indices $\{\sigma_i\}_{i=1}^{k-1}$ from the range $[n]$. We have the promise that the entries $\{X^{(i)}_{\sigma_i}\}_{i=1}^{k-1}$ are all equal to the desired answer bit $z \in \{0,1\}$. The input is initially allocated as follows:*

- *The first party P_1 knows $X^{(1)}$*
- *Each intermediate party P_p for $1 < p < k$ knows $X^{(p)}$ and σ_{p-1}*
- *The final party P_k knows just σ_{k-1}*

Communication proceeds as follows: P_1 sends a single message to P_2, then P_2 communicates to P_3, and so on, with each party sending exactly one message to its immediate successor. After all messages are sent, P_k must correctly output z, succeeding with probability at least $2/3$. If the promise condition is violated, any output is considered correct.

There is a trivial communication upper bound of $O(n)$ bits: for instance, simply have the penultimate party send $X^{(k-1)}$ to the final party who can then return $X^{(k-1)}_{\sigma_{k-1}}$.

We claim two bounds on the communication complexity of this problem.

▶ **Theorem 5.** *Any communication scheme $\mathcal{B}$ which solves $\textsc{Chain}_k$ must communicate at least $\Omega(\frac{n}{k^2})$ bits in total.*

This first bound is shown by reducing instances of another problem (conservative pointer jumping [8]) to instances of our problem.

▶ **Theorem 6.** *There is a constant $C > 0$ such that any communication scheme $\mathcal{B}$ which solves $\textsc{Chain}_k$ for $k \leq C\left(\frac{n}{\log n}\right)^{\frac{1}{4}}$ must communicate at least $\Omega(\frac{n}{k})$ bits in total.*

This second bound is shown by a lengthy and technical proof based on the structure of the pointer jumping bound given in [8]. Due to space restrictions we omit both proofs here – they can be found in the full paper.

In particular, for constant k, we have a tight bound on the communication complexity of the k-party chained index problem of $\Theta(n)$. We conjecture that a dependence on k is not necessary.

▶ **Conjecture 1.** *Any communication scheme for $\textsc{Chain}_k$ requires $\Omega(n)$ communication.*

2.2 MIS Hardness in Explicit Vertex Streams

We show a new lower bound for the vertex streaming space complexity of approximate MIS.

▶ **Theorem 1** (restated)**.** *Any algorithm for the explicit vertex stream model which finds a c-approximation to $\alpha(G)$ with probability at least $2/3$ requires $\Omega\left(\frac{n^2}{c^6}\right)$ space.*

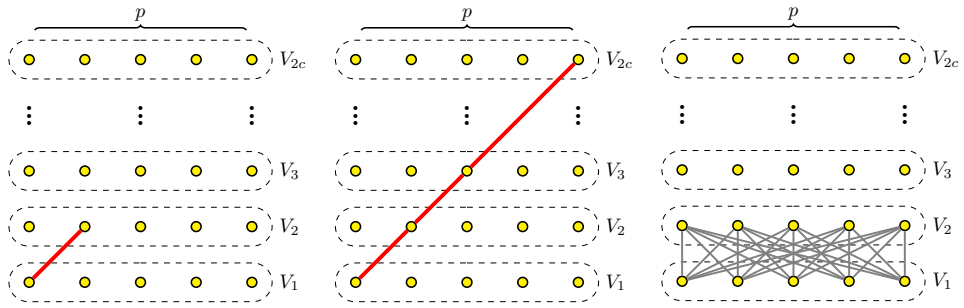
For ease of argument, we will actually prove an equivalent result for the problem of clique number approximation, and then note that the complement of the constructed graph can be used with the same arguments to prove Theorem 1. To see this equivalence, note that an MIS of a graph is a maximum clique in its complement.

▶ **Theorem 7.** *Any algorithm for the explicit vertex stream model which finds a c-approximation to the size of the largest clique $\omega(G)$ with probability at least $2/3$ requires $\Omega\left(\frac{n^2}{c^6}\right)$ space.*

The heart of our construction is to use an erasure code to encode a length $\Theta(\frac{n^2}{c^4})$ binary vector on $\Theta(\frac{n}{c})$ vertices, with each bit corresponding to the presence or absence of a clique of size $2c$. The use of the erasure code is to ensure that no pair of these cliques can share an edge. We can then chain together $2c$ such gadgets to encode an instance of $\textsc{Chain}_{2c}$ such that if the correct answer is 1, the resulting graph has an independent set of size $4c^2$, while if the correct answer is 0 the graph has no independent set larger than $4c - 1$. Any (one-sided) c-approximation algorithm could distinguish these two cases, which proves the result.

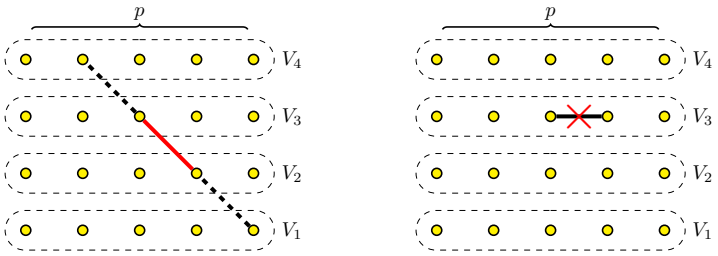
First we define our clique gadget.

▶ **Lemma 8.** *For any positive integers n and $c^2 < \frac{n}{8}$, there exists a graph on n vertices containing $\frac{n^2}{16c^2}$ edge-disjoint cliques of size $2c$ and no cliques of size larger than $2c$.*



(a) Select an edge. **(b)** Extend linearly to size $2c$. **(c)** Can make p^2 such cliques.

Figure 3 Clique gadget construction in Lemma 8.



(a) No edge belongs to two different lines (cliques).

(b) No edges within layers, so no cliques larger than $2c$.

Figure 4 Clique gadget proof sketch for Theorem 7.

Proof. We construct the sets from an erasure code with block size $2c$ and message size 2. Choose a prime p such that $\frac{n}{4c} \leq p \leq \frac{n}{2c}$ (which is guaranteed to exist). Now take $2c < p$ groups of vertices, each of size p. Label the groups V_i (for $i \in [2c]$) and label the items in each group V_i as v_j^i (for $j \in [p]$). Leftover vertices are added to the final graph as isolated vertices.

For each polynomial $\mathcal{P} \in \mathrm{GF}(p^2)$ we define $K_{\mathcal{P}}$ to be the clique over vertices $\{v^i_{\mathcal{P}(i)} | i \in [2c]\}$. This can be viewed as taking each of the p^2 possible edges between V_1 and V_2 and extending them "linearly" to the other layers (see Figure 3). In other words, the cliques correspond to non-horizontal lines in the affine plane of order p. Clearly $\mathcal{K} = \{K_{\mathcal{P}} | \mathcal{P} \in \mathrm{GF}(p^2)\}$ consists of $p^2 > \frac{n^2}{16c^2}$ cliques, each of size $2c$. We next show that they are pairwise edge-disjoint and that their union contains no larger cliques.

Each clique contains exactly one vertex from each group V_i, so for two cliques to share an edge there must be distinct polynomials $\mathcal{P}, \mathcal{Q} \in GF(p^2)$ that have the same value at two different points: $\mathcal{P}(i) = \mathcal{Q}(i)$ and $\mathcal{P}(j) = \mathcal{Q}(j)$ for $i \neq j$ – a contradiction. Finally, because no clique contains a pair of vertices from a single V_i, their union can contain no internal edges on any V_i. So any clique can contain at most 1 vertex from each V_i, giving a maximum size of $2c$. Hence, $\bigcup_{\mathcal{P} \in \mathrm{GF}(p^2)} K_{\mathcal{P}}$ is a graph with the required properties. ◀

Proof of Theorem 7. Suppose we have an algorithm $\mathcal{C}$ for explicit vertex streams which can, with probability at least $\frac{2}{3}$, produce a c-approximation to $\omega(G)$, the size of the largest clique. We will show that such an algorithm can be used to solve $\textsc{Chain}_{2c}$, by communicating its state $2c - 1$ times.

Fix an instance of $\textsc{Chain}_{2c}$ with vectors of length $b = \frac{n^2}{64c^4}$. Our lower bound in Theorem 6 implies that any algorithm that can solve this must send at least one message of size $\Omega\left(\frac{b}{c^2}\right) = \Omega\left(\frac{n^2}{c^6}\right)$ bits. Take n vertices and partition the nodes into $2c$ groups of size $\frac{n}{2c}$. Each group will be added to the stream by one of the parties.

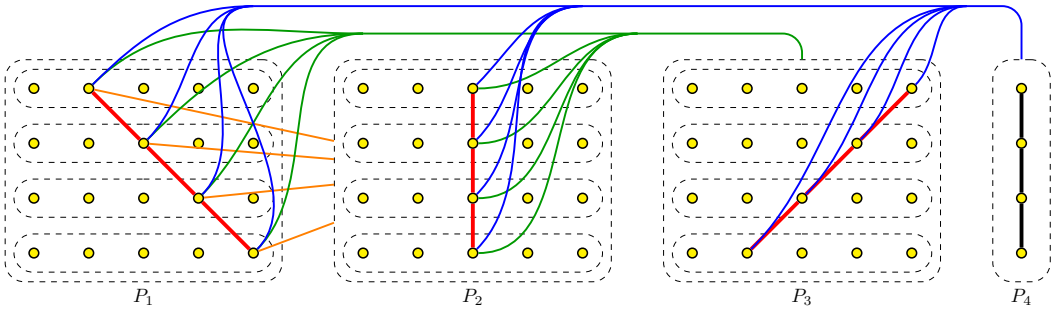


Figure 5 Example lower bound instance with 4 players for Theorem 7. Cliques corresponding to σ_1, σ_2, and σ_3 are shown in bold red – other cliques are omitted.

Intra-party edges. First, consider the group of nodes associated with party P_i. We will encode the bits of $X^{(i)}$ onto the internal edges of this group using the construction from Lemma 8. The size $\frac{n}{2c}$ sub-graph can fit b cliques of size $2c$. We include the edges of clique j if and only if $X^{(i)}_j = 1$. This is well defined as the cliques are edge-disjoint. Label the clique in party P_i corresponding to bit j of $X^{(i)}$ as $\mathcal{K}^i_j$. The final party P_{2c} has no associated vector. Instead, it constructs a single clique of size $2c$ and leaves the other vertices isolated.

Inter-party edges. We also need edges between the sub-graphs associated with different parties. Each party P_i will connect all its vertices to some of the vertices belonging to previous parties (P_j for $j < i$). These edges are considered to belong to party P_i, as they will be added by this party in the vertex streaming model. For each $j < i$ the party P_i connects every one of its vertices to all of $\mathcal{K}^j_{\sigma_j}$ (the clique corresponding to index σ_j). For this to happen, P_i must know all σ_j for $j < i$. This information is not known initially, but can be appended to the communications between players with only $O(c)$ overhead.

Now that we have our construction, we need to show bounds on $\omega(G)$ for the two cases. First, consider when every $X^{(i)}_{\sigma_i} = 1$. In this case we have each of the cliques $\mathcal{K}^i_{\sigma_i}$ present and connected together, forming a clique of size $4c^2$. Now consider the case when every $X^{(i)}_{\sigma_i} = 0$. Consider a clique $\mathcal{K}$ in the graph. If $\mathcal{K}$ contains multiple vertices belonging to one party P_i, then it can contain none from any subsequent party P_j ($j > i$), and at most one from each preceding party P_l ($l < i$). Hence the size of any clique is bounded by $4c - 1$. To see why this holds, observe that for any $i < 2c$, our clique can contain only one vertex from $\mathcal{K}^i_{\sigma_i}$, as none of its edges are included in the graph. So to contain multiple vertices from party P_i, the clique $\mathcal{K}$ must contain a vertex v from some $\mathcal{K}^i_j$ with $j \neq \sigma_i$. But then all subsequent parties P_j ($j > i$) will have no vertices adjacent to v, so cannot contribute anything to $\mathcal{K}$. So the best we can do is include one vertex from each $\mathcal{K}^i_{\sigma_i}$ and then $2c$ from party P_{2c} giving a clique of size $4c - 1$.

To complete the proof, observe that this gap in clique sizes can be distinguished by a c-approximation algorithm, and any streaming algorithm gives a communication protocol by having each party update the algorithm state with their information and then pass it to the next party. ◀

Interestingly, the same construction gives us hardness for approximating the chromatic number of a graph. This is notably not possible in the 2-party edge stream construction in [15], as the random graphs used as gadgets have large chromatic number w.h.p. (see [5]).

▶ **Corollary 9.** *Any explicit vertex streaming algorithm to find a c-approximation to $\chi(G)$ (the chromatic number), succeeding with probability at least 2/3 requires $\Omega\left(\frac{n^2}{c^6}\right)$ space.*

Proof. Consider the construction in the proof of Theorem 1. In the case of all $X_{\sigma_i}^{(i)} = 1$, the graph contains a clique of size $4c^2$, so it requires at least as many colours.

Conversely, in the case of every $X_{\sigma_i}^{(i)} = 0$, we can construct a $4c$-coloring of the graph. First color each of the nodes in each $\mathcal{K}_{\sigma_i}^i$ with the i^{th} color (this is allowed, as they have no internal edges). The remaining vertices in each party are then not adjacent to any uncolored vertices from other parties, so we simply need to be able to complete the coloring of each party in isolation with $2c$ new colors and we are finished. This is easily done, as each party's sub-graph is $2c$-partite by construction. ◀

3 Maximal Independent Set in Edge Streams

In this section, we consider streaming algorithms for the *maximal* independent set problem. Vertex streams (both explicit and implicit) are well-suited to the maximal independent set problem, since they allow the implementation of the Greedy algorithm for independent sets, which greedily adds every incoming vertex v to an initially empty independent set I if this is possible, i.e., if $I \cup \{v\}$ is an independent set. The algorithm only stores the computed independent set. This yields the following result:

▶ **Fact 1.** *The* Greedy *algorithm for independent sets is a one-pass $\tilde{O}(\alpha(G)) = \tilde{O}(n)$ space maximal independent set algorithm in vertex streams (both implicit and explicit).*

This raises the question of how well we can solve the maximal independent set problem in edge streams. We show that computing a maximal independent set in one pass in the edge-arrival model is not possible using sublinear space, i.e., space $\Omega(n^2)$ is required. This result is obtained through a reduction from the Index problem in two-party communication complexity. This proof is available in the full version of the paper.

▶ **Theorem 10.** *Every randomized constant error one-pass streaming algorithm in the edge-arrival model that computes a maximal independent set requires $\Omega(n^2)$ space.*

Since computing a maximal independent set with sublinear space is impossible in edge streams, we ask whether we can compute an *approximately maximal* independent set instead:

▶ **Definition 11** (Approximate Maximality)**.** *Let $G = (V, E)$ be an n-vertex graph, and let $I \subseteq V$ be an independent set. Then I is δ-maximal, if $|I \cup \Gamma_G[I]| \geq \delta n$.*

A δ-maximal independent set I covers a δ-fraction of the vertices, or, in other words, when removing I and its neighbors $\Gamma_G[I]$ from the graph, then at most $(1-\delta)n$ vertices are remaining. We will next show that establishing approximate maximality in edge streams requires strictly more space than computing a maximal independent set in vertex streams (i.e., $\omega(n)$ space), even if $\delta = \frac{24}{25}$. Regarding stronger approximate maximality, our lower bound yields that computing a $(1 - \frac{1}{n^\epsilon})$-maximal independent set requires space $\Omega(n^{2-o(1)})$, for every $\epsilon > 0$.

Central to our construction are *Ruzsa-Szemerédi graphs*, which have previously been used for the construction of streaming space lower bounds for maximum matching [12, 17, 4]:

▶ **Definition 12** (Ruzsa-Szemerédi graph)**.** *A bipartite graph G is an (r, s)*-Ruzsa-Szemerédi graph *if its edge set can be partitioned into r induced matchings each of size s.*

Recall that a matching $M \subseteq E$ in a graph $G = (V, E)$ is induced, if the edge set of the vertex-induced subgraph $G[V(M)]$ equals M, i.e., there are no other edges interconnecting $V(M)$ different from M.

Our lower bound for approximate maximality is obtained by a reduction from the two-party communication problem RS-Index, defined as follows:

▶ **Definition 13** (RS-Index). *Let H be an (r, s)-Ruzsa-Szemerédi graph with induced matchings $M_1, M_2, \ldots, M_r$. For each induced matching M_i, let $M_i' \subseteq M_i$ be a uniform random subset of size $s/2$ (we assume that s is even). The* RS-Index *problem is a one-way two-party communication problem, where H, and, in particular, $M_1, M_2, \ldots, M_r$ are known by both parties. In addition, Alice holds the graph $G = H[\cup_i M_i']$, and Bob holds a uniform random index $i \in \{1, 2, \ldots, r\}$. Alice sends a single message to Bob, who, upon receipt, outputs at least $C \cdot s$ edges of M_i', for an arbitrary small constant C.*

Observe that this problem is similar in spirit to Index: In Index, Bob needs to learn one uniform random bit, while in RS-Index, Bob needs to learn the presence of many edges of M_i'. A lower bound on the communication complexity of RS-Index is implicit in [12] [2]:

▶ **Theorem 14** ([12]). *The randomized constant error communication complexity of* RS-Index *is $\Omega(r \cdot s)$.*

Equipped with the RS-Index problem, we now give a reduction to approximate maximality from RS-Index, which yields our lower bound for streaming algorithms:

▶ **Lemma 15.** *Let r, s, n be integers such that there is an n-vertex (r, s)-Ruzsa-Szemerédi graph. Then, every randomized constant error one-pass streaming algorithm in the edge-arrival model that computes a $(1 - \frac{s}{6n})$-maximal independent set requires $\Omega(r \cdot s)$ space.*

Proof. Let H be an n-vertex (r, s)-Ruzsa-Szemerédi graph, and let G be Alice's input graph for the RS-Index problem derived from H. Let $M_1, M_2, \ldots, M_r$ denote the induced matchings in H, let $V_i = V(M_i)$, and let $M_i' \subseteq M_i$ denote the subset of edges of matching M_i that is included in G. Let i be Bob's input. Furthermore, let $\mathcal{A}$ be a constant error randomized one-pass streaming algorithm for the edge-arrival model that computes a $(1-\frac{s}{6N})$-maximal independent set on a graph on N vertices. We now show how $\mathcal{A}$ can be used to solve RS-Index:

Given G, let $\tilde{G}$ be the graph obtained from G, where every induced matching M_i' in G is replaced by edges $\tilde{M}_i' := M_i \setminus M_i'$ (observe that $E(G) \cup E(\tilde{G}) = E(H)$). Alice now constructs two disjoint copies G_1 and G_2 of $\tilde{G}$, runs algorithm $\mathcal{A}$ on $G_1 \dot{\cup} G_2$ (on an arbitrary ordering of their edges), and sends the memory state to Bob. Bob constructs the edge set F that connects every vertex $v_1 \in V(G_1) \setminus V_{i1}$ with every vertex $v_2 \in V(G_2) \setminus V_{i2}$, where V_{i1} and V_{i2} are the copies of the vertices V_i in graphs G_1 and G_2, respectively, and continues the execution of $\mathcal{A}$ on F. Let I be the independent set produced by algorithm $\mathcal{A}$.

Observe that the graph processed by algorithm $\mathcal{A}$ contains $N = 2n$ vertices. Since I is $(1 - \frac{s}{6N})$-maximal, we have $|V \setminus \Gamma[I]| \leq N - (1 - \frac{s}{6N})N = s/6$. This allows us to identify $\Omega(s)$ edges of M_i' as follows:

Let a, b be the incident vertices to an arbitrary edge of M_i', let a_1, b_1 be the copies of a, b in G_1, and let a_2, b_2 be the copies of a, b in G_2. Observe that a_1 and b_1 are not connected in G_1, and a_2 and b_2 are not connected in G_2. We now claim that if all vertices a_1, b_1, a_2, b_2

[2] In [12] a lower bound is given for the task of computing a maximum matching. Their hardness stems from the fact that it is hard to learn many edges of M_i' under the distribution described in the definition of RS-Index.

are covered by I, i.e., $\{a_1, b_1, a_2, b_2\} \subseteq \Gamma[I]$, then either $\{a_1, b_1\} \subseteq I$ or $\{a_2, b_2\} \subseteq I$ (or both). Indeed, suppose that this is not the case. Then there are vertices $x_1 \in \{a_1, b_1\}$ and $x_2 \in \{a_2, b_2\}$ with $x_1, x_2 \notin I$. Let $y_1 \in I$ be a vertex incident to x_1, and let $y_2 \in I$ be a vertex incident to x_2. By the construction of the input graph, $y_1 \in V(G_1) \setminus V_{i1}$, and $y_2 \in V(G_2) \setminus V_{i2}$. Observe, however, that the edge y_1y_2 was included by Bob, which implies that y_1, y_2 are not independent: a contradiction. Hence, either $\{a_1, b_1\} \subseteq I$ or $\{a_2, b_2\} \subseteq I$ (or both) hold. This implies that the algorithm identified that there is no edge between a_1, b_1, which in turn implies that we learned one edge of M'_i. Hence, for every pair of vertices a, b of M'_i, either at least one vertex among $\{a_1, b_1, a_2, b_2\}$ is not covered by I, or we learn one edge of M'_i. Since there are $s/2$ edges in M'_i, and at most $s/6$ vertices of the input graph are not covered by I, we learn at least $s/2 - s/6 = \Omega(s)$ edges of M'_i, which thus solves RS-Index. By Theorem 14, algorithm $\mathcal{A}$ therefore requires space $\Omega(r \cdot s)$. ◀

In [12] it is shown that there are n-vertex $(n^{\Theta(\frac{1}{\log\log n})}, (\frac{1}{4} - \epsilon)n)$ Ruzsa-Szemerédi graphs, for every $\epsilon > 0$, and in [2], it is shown that there are such graphs with $\Theta(n^{2-o(1)})$ edges such that each matching is of size $n^{1-o(1)}$. Combined with Lemma 15, we obtain:

▶ **Theorem 16.** *Every randomized constant error one-pass streaming algorithm that computes a $\frac{24}{25}$-maximal independent set requires space $n^{1+\Omega(\frac{1}{\log\log n})}$, and every such algorithm computing a $(1 - \frac{1}{n^\epsilon})$-maximal independent set requires space $\Omega(n^{2-o(1)})$, for every $\epsilon > 0$.*

Last, interestingly, if we allow an algorithm to perform multiple passes, then sublinear space algorithms can be obtained. Such algorithms are in fact immediately implied by the correlation clustering algorithms given in [1]. Their result yields the following theorem:

▶ **Theorem 17.** *There is a $O(\log\log n)$-pass streaming algorithm for maximal independent set that uses space $\tilde{O}(n)$.*

4 Maximum Independent Set in Geometric Intersection Graphs

We now present a collection of results around geometric intersection graphs, in one and two dimensions, given as explicit or implicit vertex streams. We consider intervals and squares.

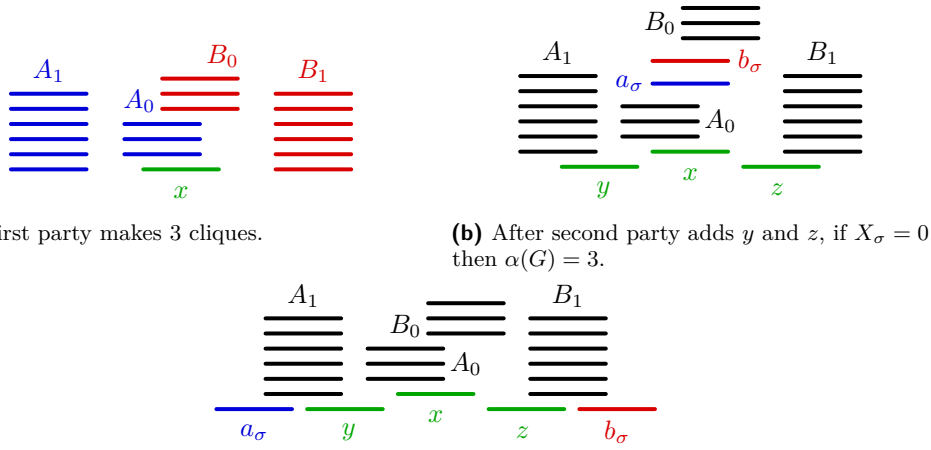
A geometric intersection graph is a graph where nodes correspond to geometric objects, and edges indicate whether or not a particular pair of objects intersect. These graphs can be described implicitly as the collection of geometric objects, or explicitly as a collection of vertices and edges under the promise that some geometric representation exists.

For implicit representations, we assume that intervals and squares are presented by their centers and their lengths. We assume that the center is a value in $[M]^d$ ($d = 1$ for intervals, and $d = 2$ for squares), and the length is in $[M]$, for some $M \in$ poly n.

4.1 Unit Interval Graphs: $\mathrm{d} = 1$

As discussed in Section 1, given a stream of unit intervals we can compute a $\frac{3}{2}$-approximation to MIS in $\tilde{O}(\alpha(G))$ space, and any better approximation requires $\Omega(n)$ space. A natural question is how this compares with the space complexity for an interval intersection graph given as an explicit vertex stream:

▶ **Theorem 18.** *Any algorithm with constant error probability that returns a $(\frac{5}{3} - \epsilon)$-approximation of $\alpha(G)$ for a unit interval intersection graph given as an explicit vertex stream requires $\Omega(n)$ space.*



(a) First party makes 3 cliques.

(b) After second party adds y and z, if $X_\sigma = 0$ then $\alpha(G) = 3$.

(c) Conversely, if $X_\sigma = 1$ then $\alpha(G) = 5$.

Figure 6 Interval representations for the construction in theorem 18. Horizontal positioning represents the location of the intervals in $\mathbb{R}$, vertical positioning is for clarity only.

Proof. We will show this bound by a reduction from the 2-party INDEX communication problem. Consider an instance of INDEX with bit vector $X \in \{0,1\}^n$ and index to be queried $\sigma \in [n]$. We will construct a $2n+3$ vertex graph as an explicit vertex stream.

Label the vertices x, y, z and a_i, b_i for $i \in [n]$. Split the a_i's into two sets based on the bit vector X: $A_1 = \{a_i\}_{X_i=1}$ and $A_0 = \{a_i\}_{X_i=0}$. Similarly let $B_1 = \{b_i\}_{X_i=1}$ and $B_0 = \{b_i\}_{X_i=0}$. Now the first party creates the following subgraph in the stream: a clique consisting of all the vertices in A_1, a second clique made from B_1, and a third clique containing $A_0 \cup B_0 \cup \{x\}$.

So far this represents a valid interval graph, which can be interpreted as three adjacent "stacks" of intervals. Now, the second player adds y with edges to every a_i except a_σ and then adds z with edges to every b_i except b_σ. This can still be viewed as a valid interval graph, but we now require some intervals from each stack to be "shifted" to overlap with the two new intervals.

In the case of $X_\sigma = 0$, the resulting graph has $\alpha(G) = 3$. Otherwise, $\alpha(G) = 5$. Hence, any algorithm giving a better than $\frac{5}{3}$-approximation factor could distinguish them and solve INDEX. ◀

This shows that MIS for interval graphs is strictly more difficult in explicit vertex streams than implicit ones.

4.2 Square Graphs: $\mathrm{d} = 2$

We obtain several improved bounds for the 2D case. Full details can be found in the extended paper, but we briefly summarise here.

Our first result for 2D is a 3-approximation algorithm for MIS on a unit square stream. This is a generalization of the algorithm of [7] for unit interval streams – we perform a decomposition of the plane into 2-by-3 strips, similar to their decomposition of the line into length 3 segments.

▶ **Theorem 19.** *There is a 3-approximation streaming algorithm for* MIS *on a stream of unit squares (implicit vertex stream) using $\tilde{O}(\alpha(G))$ space.*

As in [7] for unit intervals, this immediately leads to a sublinear space algorithm for estimating $\alpha(G)$ with only a $(1+\epsilon)$ factor loss in approximation factor, through a combination of counting distinct elements and clever sampling.

▶ **Corollary 20.** *We can $(3+\epsilon)$-approximate $\alpha(G)$ with constant probability in a stream of unit squares using $O(\epsilon^{-2}\log\epsilon^{-1}+\log n)$ space.*

One might speculate whether this decomposition approach could afford a better approximation factor based on some different partitioning of the plane. We give evidence for the negative, since any larger strip size results in the fixed-size sub-problems not being solvable exactly, as the following result shows.

▶ **Theorem 21.** *Given a stream of w-by-w squares contained in a $(2+\delta)w$-by-$(2+\delta)w$ region, achieving a $(\frac{3}{2}-\epsilon)$-approximation to $\alpha(G)$, with constant probability of success for any $\epsilon,\delta>0$ requires $\Omega(n)$ space.*

Our next result for two dimensions is a stronger lower bound for approximating $\alpha(G)$ of a stream of unit squares in an unrestricted region, based on a reduction from the chained index communication problem used in our main result in Section 2.

▶ **Theorem 22.** *Achieving a $(\frac{5}{2}-\epsilon)$-approximation of $\alpha(G)$, with constant probability of success, on a unit square stream requires $\Omega(n)$ space for any $\epsilon>0$.*

If we are allowed a combination of large and small balls, we can slightly improve the lower bound up to the maximum possible for a 3-party construction.

▶ **Theorem 23.** *Achieving a $(3-\epsilon)$-approximation of $\alpha(G)$, with constant probability of success, on a stream of squares or arbitrary side lengths requires $\Omega(n)$ space for any $\epsilon>0$.*

5 Conclusion

We have looked at the complexity of Maximal and Maximum Independent Set (and various relaxations and related problems) under three natural models of graph streams: edge-arrival, explicit vertex-arrival, and implicit vertex-arrival.

By making use of a new communication problem $\textsc{Chain}_k$, we showed that MIS is not significantly easier on explicit vertex streams than edge streams. However, the question of whether they have exactly the same complexity is left open. Improving the communication bound on $\textsc{Chain}_k$ to $\Omega(n)$, as we conjectured, would improve our MIS lower bound to $\Omega\left(\frac{n^2}{c^5}\right)$, but we do not know of any vertex stream upper bounds better than the $\tilde{O}\left(\frac{n^2}{c^2}\right)$ algorithm for general edge streams.

There are a number of other open questions that naturally follow from our study:

- Is there a multi-pass lower bound for *maximal* independent set in edge streams?
- Are there $o(\alpha(G))$ space algorithms for achieving constant factor approximations to $\alpha(G)$ for classes of geometric intersection graphs given as *explicit* vertex streams?
- Can we close the gap between the 3 and 5/2 factors of the upper and lower bounds for approximating MIS in a unit square stream?
- Is there an $O(\alpha(G))$ space constant factor approximation algorithm for MIS on streams of arbitrary sized squares?
- Can $\textsc{Chain}_k$ be used to form novel lower bounds for other kinds of problems?

References

1 Kook Jin Ahn, Graham Cormode, Sudipto Guha, Andrew McGregor, and Anthony Wirth. Correlation Clustering in Data Streams. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 2237–2246. JMLR.org, 2015. URL: `http://dl.acm.org/citation.cfm?id=3045118.3045356`.

2 Noga Alon, Ankur Moitra, and Benny Sudakov. Nearly Complete Graphs Decomposable into Large Induced Matchings and Their Applications. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 1079–1090, New York, NY, USA, 2012. ACM. doi:10.1145/2213977.2214074.

3 Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear Algorithms for $(\Delta + 1)$ Vertex Coloring. In *SODA*, 2019.

4 Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum Matchings in Dynamic Graph Streams and the Simultaneous Communication Model. In *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 1345–1364, Philadelphia, PA, USA, 2016. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=2884435.2884528.

5 Béla Bollobás. The chromatic number of random graphs. *Combinatorica*, 8(1):49–55, 1988.

6 Vladimir Braverman, Zaoxing Liu, Tejasvam Singh, N. V. Vinodchandran, and Lin F. Yang. New Bounds for the CLIQUE-GAP Problem Using Graph Decomposition Theory. *Algorithmica*, 80(2):652–667, February 2018. doi:10.1007/s00453-017-0277-5.

7 Sergio Cabello and Pablo Pérez-Lantero. Interval selection in the streaming model. *Theoretical Computer Science*, 702:77–96, 2017.

8 Amit Chakrabarti. Lower bounds for multi-player pointer jumping. In *Computational Complexity, 2007. CCC'07. Twenty-Second Annual IEEE Conference on*, pages 33–45. IEEE, 2007.

9 Graham Cormode, Jacques Dark, and Christian Konrad. Approximating the Caro-Wei Bound for Independent Sets in Graph Streams. In Jon Lee, Giovanni Rinaldi, and A. Ridha Mahjoub, editors, *Combinatorial Optimization*, pages 101–114, Cham, 2018. Springer International Publishing.

10 Graham Cormode, Jacques Dark, and Christian Konrad. Independent Sets in Vertex-Arrival Streams. *CoRR*, abs/1807.08331, 2018. arXiv:1807.08331.

11 Yuval Emek, Magnús M Halldórsson, and Adi Rosén. Space-constrained interval selection. *ACM Transactions on Algorithms (TALG)*, 12(4):51, 2016.

12 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 468–485, 2012.

13 Bjarni V Halldórsson, Magnús M Halldórsson, Elena Losievskaja, and Mario Szegedy. Streaming algorithms for independent sets. In *International Colloquium on Automata, Languages, and Programming*, pages 641–652. Springer, 2010.

14 Magnús M. Halldórsson and Christian Konrad. Computing Large Independent Sets in a Single Round. *Distrib. Comput.*, 31(1):69–82, February 2018. doi:10.1007/s00446-017-0298-y.

15 Magnús M Halldórsson, Xiaoming Sun, Mario Szegedy, and Chengu Wang. Streaming and communication complexity of clique approximation. In *International Colloquium on Automata, Languages, and Programming*, pages 449–460. Springer, 2012.

16 R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

17 Christian Konrad. Maximum Matching in Turnstile Streams. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms - ESA 2015*, pages 840–852, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

18 I. Kremer, N. Nisan, and D. Ron. On Randomized One-round Communication Complexity. *computational complexity*, 8(1):21–49, 1999. doi:10.1007/s000370050018.

19 Andrew McGregor. Graph Stream Algorithms: A Survey. *SIGMOD Rec.*, 43(1):9–20, May 2014. doi:10.1145/2627692.2627694.

20 David Zuckerman. Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number. *Theory of Computing*, 3(1):103–128, 2007.

Approximation Algorithms for Min-Distance Problems

Mina Dalirrooyfard
MIT, Cambridge, MA, USA
minad@mit.edu

Virginia Vassilevska Williams
MIT, Cambridge, MA, USA
virgi@mit.edu

Nikhil Vyas
MIT, Cambridge, MA, USA
nvyas@mit.edu

Nicole Wein
MIT, Cambridge, MA, USA
nwein@mit.edu

Yinzhan Xu
MIT, Cambridge, MA, USA
xyzhan@mit.edu

Yuancheng Yu
MIT, Cambridge, MA, USA
ycyu@mit.edu

Abstract

We study fundamental graph parameters such as the Diameter and Radius in directed graphs, when distances are measured using a somewhat unorthodox but natural measure: the distance between u and v is the *minimum* of the shortest path distances from u to v and from v to u. The center node in a graph under this measure can for instance represent the optimal location for a hospital to ensure the fastest medical care for everyone, as one can either go to the hospital, or a doctor can be sent to help.

By computing All-Pairs Shortest Paths, all pairwise distances and thus the parameters we study can be computed exactly in $\tilde{O}(mn)$ time for directed graphs on n vertices, m edges and nonnegative edge weights. Furthermore, this time bound is tight under the Strong Exponential Time Hypothesis [Roditty-Vassilevska W. STOC 2013] so it is natural to study how well these parameters can be *approximated* in $O(mn^{1-\varepsilon})$ time for constant $\varepsilon > 0$. Abboud, Vassilevska Williams, and Wang [SODA 2016] gave a polynomial factor approximation for Diameter and Radius, as well as a constant factor approximation for both problems in the special case where the graph is a DAG. We greatly improve upon these bounds by providing the first constant factor approximations for Diameter, Radius and the related Eccentricities problem in general graphs. Additionally, we provide a hierarchy of algorithms for Diameter that gives a time/accuracy trade-off.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

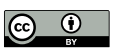
Keywords and phrases fine-grained complexity, graph algorithms, diameter, radius, eccentricities

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.46

Category Track A: Algorithms, Complexity and Games

Related Version `https://arxiv.org/abs/1904.11606`

Acknowledgements The authors would like to thank the members of the MIT course 6.S078 open problem sessions, especially Thuy-Duong Vuong, Robin Hui, and Ali Vakilian. These sessions were organized by Erik Demaine, Ryan Williams, and Virginia Vassilevska Williams.



46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 46; pp. 46:1–46:14

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

1 Introduction

The diameter, radius and eccentricities of a graph are fundamental parameters that have been extensively studied [13, 20, 12, 18, 3, 14, 11, 17, 5, 6, 26, 27, 9, 19, 24, 23, 10, 1, 7] (and many others). The eccentricity of a vertex v is the largest distance between v and any other vertex. The diameter is the maximum eccentricity of a vertex in the graph, thus measuring how far apart two nodes can be, and the radius is the minimum eccentricity, measuring the maximum distance to the most central node.

The distance between two vertices in an undirected graph is just the shortest path distance $d(\cdot,\cdot)$ between them. For directed graphs, however, this notion of distance d is no longer necessarily symmetric, and rather than being a distance *between* two nodes, it measures the distance in a given direction. Several related notions of pairwise distance that are symmetric have been studied. These include the roundtrip distance [15] which for two vertices u and v is just $d(u,v)+d(v,u)$, the max-distance [2] which is $\max\{d(u,v),d(v,u)\}$, and the min-distance [2] which is $\min\{d(u,v),d(v,u)\}$.

Each of these notions of distance has a particular application. For instance, one would have to pay the roundtrip distance when going to the store and back. On the other hand, if one needs medical assistance, one could either go to the hospital, or have a physician come to the home – the time to receive care is then measured by the min-distance. Another example of min-distance is in symmetric-key encryption: any pair of parties can create a shared private key by using only one-way communication.

For each notion of distance, the diameter, radius and eccentricity parameters are well-defined. Given the shortest path distances $d(\cdot,\cdot)$ for all vertices, the parameters for each distance measure can be computed in $O(n^2)$ time in n vertex graphs. The fastest known algorithms for All-Pairs Shortest Paths (APSP) [25, 21, 22] give the fastest known algorithms to compute these parameters exactly, running in $n^3/\exp(\sqrt{\log n})$ time and $O(mn+n^2\log\log n)$, respectively on m-edge, n-vertex graphs. Furthermore, under the Strong Exponential Hypothesis, there is no $O(m^{2-\varepsilon})$ time algorithm for Diameter in unweighted graphs (and thus also for any of these notions of Diameter and Eccentricities in directed graphs) [23]. For Radius, the same lower bound holds but under the "Hitting Set" conjecture [2].

As exact computation is expensive, it makes sense to resort to approximation algorithms. For the shortest path distance versions of Diameter, Eccentricities and Radius, there are several fast algorithms that achieve various small constant approximation ratios [23, 10, 8, 4]. For instance, for Diameter, a folklore linear time algorithm can achieve a 2-approximation, and an $\tilde{O}(m^{3/2})$ time[1] algorithm can achieve a 3/2-approximation [23, 10].

Many of these algorithms [23, 10, 4] work for any distance measure that satisfies the triangle inequality. Thus they work for the shortest paths distance, max-distance and roundtrip distance. The min-distance however does not satisfy the triangle inequality: e.g. you might have edges (x,y) and (z,y), and thus the min-distance between x and y and between y and z are both 1, yet there may be no directed path between x and z in any direction, so that the min-distance between them may be ∞.

This issue makes it much more difficult to design fast approximation algorithms for Min-Diameter, Min-Radius and Min-Eccentricities (the parameters of interest under the min-distance). The only known nontrivial algorithms are by Abboud et al. [2]. For Min-Diameter [2] gives a near-linear time 2-approximation algorithm if the input is a directed acyclic graph. For general graphs, the only nontrivial fast approximation algorithm is an

[1] We use $\tilde{O}$ notation to hide polylogarithmic factors.

$\tilde{O}(mn^{1-\varepsilon})$ time n^{ε}-approximation algorithm for any constant $\varepsilon > 0$. (No constant factor approximation algorithm is known that runs significantly faster than just computing APSP.) For Min-Radius, [2] gives an $\tilde{O}(m\sqrt{n})$ time 3-approximation algorithm for directed acyclic graphs. For general graphs, they only achieve a very weak n-approximation in near-linear time that checks if the Min-Radius is finite. There are no known approximation algorithms for Min-Eccentricities faster than just computing APSP.

1.1 Our Results

The main goal of our paper is to obtain new fast, $O(mn^{1-\varepsilon})$ time for some constant $\varepsilon > 0$, algorithms for Min-Diameter, Min-Radius and Min-Eccentricities (thus beating the $\tilde{O}(mn)$ time of exact computation). We achieve this by developing powerful new techniques that can handle the complications that arise due to the fact that the min-distance does not satisfy the triangle inequality.

Our results are as follows. For Min-Diameter we achieve a hierarchy of algorithms trading off running time with approximation accuracy.

▶ **Theorem 1.** *For any integer $0 < \ell \leq O(\log n)$, there is an $\tilde{O}(mn^{1/(\ell+1)})$ time randomized algorithm that, given a directed weighted graph G with edge weights non-negative and polynomial in n, can output an estimate $\tilde{D}$ such that $D/(4\ell - 1) \leq \tilde{D} \leq D$ with high probability, where D is the min-diameter of G.*

When we set $\ell = 1$, we obtain an $\tilde{O}(m\sqrt{n})$ time 3-approximation algorithm, and when we set $\ell = \lceil \log n \rceil$, we get an $\tilde{O}(m)$ time $O(\log n)$-approximation.

Our tradeoff achieves the first constant factor approximation algorithms for Min-Diameter in general graphs that run in $O(mn^{1-\varepsilon})$ time for constant $\varepsilon > 0$. Such a result was only known for directed acyclic graphs, whereas for general graphs the only known efficient algorithm could achieve an n^{ε}-approximation.

For Min-Radius, we also achieve the first constant factor approximation algorithm for general graphs running in $O(mn^{1-\varepsilon})$ time for some constant $\varepsilon > 0$. Such a result was only known for directed acyclic graphs, whereas for general graphs the only known efficient algorithm could only check if the Min-Radius is finite.

▶ **Theorem 2.** *For any constant δ with $1 > \delta > 0$, there is an $\tilde{O}(m\sqrt{n}/\delta)$ time randomized algorithm, that given a directed weighted graph G with edge weights positive and polynomial in n, can output an estimate R' such that $R \leq R' \leq (3+\delta)R$ with high probability, where R is the min-radius of G.*

Finally, we obtain the first $O(mn^{1-\varepsilon})$ time (for constant $\varepsilon > 0$) constant factor approximation algorithms for the Min-Eccentricities of all vertices in a graph. For unweighted graphs we are able to obtain a close to 3 approximation in $\tilde{O}(m\sqrt{n})$ time. For weighted graphs, our approximation factor grows to 5, while the running time is the same. Previously, the only algorithm to approximate the Min-Eccentricities computed them exactly via an APSP computation.

▶ **Theorem 3.** *For any constant δ with $1 > \delta > 0$, there is an $\tilde{O}(m\sqrt{n}/\delta)$ time randomized algorithm, that given a directed weighted graph $G = (V, E)$ with weights positive and polynomial in n, can output an estimate $\varepsilon'(s)$ for every vertex $s \in V$ such that $\varepsilon(s) \leq \varepsilon'(s) \leq (5+\delta)\varepsilon(s)$ with high probability, where $\varepsilon(s)$ is the min-eccentricity of vertex s in G.*

▶ **Theorem 4.** *For any constant δ with $1 > \delta > 0$, there is an $\tilde{O}(m\sqrt{n}/\delta^2)$ time randomized algorithm, that given a directed unweighted graph $G = (V, E)$, can output an estimate $\varepsilon'(s)$ for every vertex $s \in V$ such that $\varepsilon(s) \leq \varepsilon'(s) \leq (3+\delta)\varepsilon(s)$ with high probability, where $\varepsilon(s)$ is the min-eccentricity of the vertex s in G.*

1.2 Our Techniques

To obtain our results, we develop powerful new techniques which we outline below.

Partial search graphs. The idea of partial search graphs is used in the algorithms of [2] for Min-Radius and Min-Diameter on DAGs. These algorithms use the following high-level framework: perform Dijkstra's algorithm from some vertices and then perform a *partial* Dijkstra's algorithm from *every* vertex. The partial search from a vertex v is with respect to a carefully defined partial search graph $G_v \subset G$. The crux of the analysis for the algorithms on DAGs is to argue that if the executions of Dijkstra's algorithm on the full graph did not find a good estimate for the desired quantity (either min-diameter or min-radius), then the partial search from some vertex v returns a good estimate of the min-eccentricity of v, which in turn is a good estimate for the desired quantity. In DAGs it is natural to define the partial search graphs G_v by considering a topological ordering of the vertices and letting each G_v be some interval containing v (though defining the exact intervals requires some work). For general graphs it is completely unclear how to even define such intervals since there is no natural notion of an ordering of the vertices, and thus figuring out what the G_v's should be is nontrivial. Our approach to overcoming this hurdle is to carefully define a DAG-like structure in general graphs. Such a structure may be of independent interest.

Defining a DAG-like structure in general graphs. It would be ideal to directly reduce the problem on general graphs to the problem on DAGs, however it is very unclear how to do this. Instead, we recognize that it suffices to define a *DAG-like* structure in general graphs. As a first step, we use the following idea. Suppose we have performed Dijkstra's algorithm from a vertex v. We let $S_v = \{u : d(u,v) < d(v,u)\}$ and we let $T_v = \{u : d(u,v) > d(v,u)\}$[2]. Then, we partially order the vertices so that the vertices in S_v appear before v and those in T_v appear after v. We note that this partial ordering is "DAG-like" because it is consistent with the topological ordering of a DAG; that is, if we apply this partition into S_v and T_v to a DAG then there trivially exists a topological ordering such that every vertex in S_v appears before v and every vertex in T_v appears after v. After partitioning into S_v and T_v, we recursively partition each set to create a more precise partial ordering. Importantly, we show that by recursively sampling vertices randomly, we can guarantee that our partitioning is approximately balanced which is crucial for the runtime analysis. The obtained partial ordering is the starting point for all of our algorithms.

Min-Diameter: graph augmentation. The Min-Diameter algorithm on DAGs from [2] relies heavily on the following key property of DAGs. Consider a topological ordering and the graphs induced by the first and second halves of the ordering; which are defined with respect to the middle vertex in the ordering. For all pairs of vertices in the same half of the ordering, their min-distance in the graph induced by this half is the same as their min-distance in the full graph. As previously mentioned, if we sample a vertex v, we can make sure that S_v and T_v are approximately balanced, so that we can think of S_v and T_v as corresponding to the first and second half of a DAG topological ordering, respectively. However it is unclear how to obtain a property of S_v and T_v analogous to the above key property of DAGs. In particular, the min-distance between a pair of vertices in the graph induced by S_v could be wildly different from their min-distance in the full graph, since paths whose endpoints are

[2] u's with $d(u,v) = d(v,u)$ are added to either S_v or T_v as specified in the formal definition later

in S_v can contain vertices outside of S_v. To overcome this hurdle, we *augment* the graph induced by S_v and the graph induced by T_v by carefully adding edges so that distances within these augmented graphs approximate distances in the original graph.

Min-Radius: refined DAG-like structure. Our Min-Radius algorithm is much more delicate than our Min-Diameter algorithm due to the fact that for Min-Radius we care about small distances instead of large distances. In particular, the graph augmentation idea from our Min-Diameter algorithm does not help for Min-Radius because although the augmentations do not distort large distances much, they heavily distort small distances. Furthermore, the previously mentioned DAG-like structure for general graphs does not suffice for Min-Radius. However we use it as a starting point to define a more refined DAG-like partial ordering. Most of our algorithm is concerned with precisely arranging vertices in this partial ordering. Specifically, we structure the partial ordering to satisfy *roughly* the following property: for every pair of vertices u, v such that u appears before v in the partial ordering, $d(v,u)$ is large while $d(u,v)$ is small.

1.3 Notation

Given a graph $G=(V,E)$, $n=|V|$ and $m=|E|$. Graphs are directed and have non-negative weights polynomial in n unless otherwise specified. For any pair of vertices u and v, the *distance from u to v* $d(u,v)$ is the length of the shortest directed path from u to v. When the context is not clear, we write $d_G(u,v)$ to specify the graph G. The *min-distance* between a pair of vertices u and v is $d_{min}(u,v)=\min\{d(u,v),d(v,u)\}$. The *min-diameter* of a graph is $\max_{u,v\in V} d_{min}(u,v)$. The *min-radius* of a graph is $\min_{v\in V}\max_{u\in V} d_{min}(u,v)$. For any vertex v, the *min-eccentricity* of v is $\varepsilon(v)=\max_{u\in V} d_{min}(u,v)$. When the context is not clear, we say $\varepsilon_G(v)$ to specify the graph G. Note that we do not use the *min* subscript to denote the min-eccentricity of a vertex. For an algorithm with input size n we use *with high probability* to denote the probability $>1-1/n^c$ for all constants c. We say some quantity is *poly*(n) to mean it is $O(n^c)$ for some fixed constant c. We use $\tilde{O}$ notation to hide polylogarithmic factors.

1.4 Organization

In Section 2 we give an overview of all of our algorithms, in Section 3 we describe a graph partitioning procedure that begins all of our algorithms, in Section 4 we describe our Min-Diameter algorithms. We defer the time/accuracy tradeoff algorithm for Min-Diameter, the Min-Radius algorithm and the Min-Eccentricities algorithm to the full version [16].

2 Overview of Algorithms

We use the algorithms from [2] for Min-Diameter and Min-Radius on DAGs as inspiration. For each problem, we first outline the DAG algorithm and then provide intuition for how to apply these ideas to general graphs.

2.1 Min-Diameter

Algorithm for DAGs

We begin by outlining the $\tilde{O}(n+m)$ time 2-approximation algorithm for Min-Diameter on DAGs from [2]. Consider a topological ordering of the vertices and perform Dijkstra's algorithm from the middle vertex v. Then recurse on the graphs induced by the vertices in

the first half (before v) and in the second half (after v). A key observation in the analysis is that if the true endpoints s^* and t^* of the min-diameter fall on opposite sides of v in the ordering, then the min-eccentricity $\varepsilon(v)$ of v is a 2-approximation for the min-diameter D. This is because if $\varepsilon(v) < D/2$ and s^* and t^* fall on opposite sides of v in the ordering, then $d(s^*, v) < D/2$ and $d(v, t^*) < D/2$ so $d(s^*, t^*) < D$, a contradiction. So, suppose (without loss of generality) that s^* and t^* both fall before v in the ordering. Since the graph is a DAG, every path between s^* and t^* only uses vertices before v in the ordering. Thus, the min-distance between s^* and t^* in the graph induced by the first half of the graph is still D.

Algorithm for general graphs

We now outline a precursor to our Min-Diameter algorithm for general graphs that mimics the algorithm for DAGs. This $\tilde{O}(n+m)$ time algorithm does not achieve a constant approximation factor, however it provides intuition for our constant-factor approximation algorithms. We begin by performing Dijkstra's algorithm from a vertex v and constructing S_v and T_v as defined in the previous section. Analogously to the DAG algorithm if the true min-diameter endpoints s^* and t^* fall into different sets S_v, T_v then the min-eccentricity $\varepsilon(v)$ is a 2-approximation. This is because if $\varepsilon(v) < D/2$, $s^* \in S_v$, and $t^* \in T_v$ then $d(s^*, v) < D/2$ and $d(v, t^*) < D/2$ so $d(s^*, t^*) < D$, a contradiction. However, unlike the DAG algorithm, we cannot simply recurse independently on the graphs induced by S_v and T_v since the shortest path between a pair of vertices in S_v may not be completely contained in S_v (and analogously for T_v).

To overcome this hurdle, before recursing we first augment the graphs induced by S_v and T_v by carefully adding edges so that distances within these augmented graphs approximate distances in the original graph. Specifically, for every vertex $u \in S_v$, we add the directed edge (u, v) with weight 0 and the directed edge (v, u) with weight $\max\{0, d(v, u) - \varepsilon(v)\}$. This choice of edges allows us to argue that the distances within the augmented graphs are approximations of the distances in G up to an additive error of $2\varepsilon(v)$. Then, by returning the maximum of $\varepsilon(v)$ and the min-diameter estimates from recursing on the augmented graphs, we get an approximation guarantee, which turns out to be a logarithmic factor. Intuitively, the approximation factor is not constant because the recursion causes the distance distortion to compound at each level of recursion.

To reduce the approximation factor to a constant, we would like to decrease the number of recursion levels. To achieve this, we initially partition the graph into more than just two parts S_v and T_v, by sampling more vertices. For our $\tilde{O}(m\sqrt{n})$ time 3-approximation, we perform a full Dijkstra's algorithm from $\tilde{O}(\sqrt{n})$ vertices to define an ordered partition of the vertices into $\tilde{O}(\sqrt{n})$ parts of $\tilde{O}(\sqrt{n})$ vertices each. Then we apply the above idea of adding weighted edges within each part, however we must refine the definition of the graph augmentation to take into account *all* of the $\tilde{O}(\sqrt{n})$ vertices we initially perform Dijkstra's algorithm from, instead of just v. Finally we use brute force (without recursion) on each part in the partition by running an exact all-pairs shortest paths algorithm.

To achieve our time-accuracy trade-off algorithm, we carefully combine ideas from the logarithmic factor approximation and the 3-approximation algorithms. Specifically, we initially perform Dijkstra's algorithm from fewer than $\sqrt{n}$ vertices to define an ordered partition with larger parts than in the 3-approximation. Then we augment the graph induced by each part and carry out a constant number of recursion levels to further partition the graph before applying brute-force.

2.2 Min-Radius

Algorithm for DAGs

We begin by outlining the $\tilde{O}(m\sqrt{n})$ time 3-approximation algorithm for Min-Radius on DAGs from [2], which is very different from and more involved than the Min-Diameter algorithm on DAGs. We begin by considering a topological ordering of the vertices and performing Dijkstra's algorithm from a set W of $\tilde{O}(\sqrt{n})$ evenly spaced vertices including the first and last vertex. If a vertex $v \in W$ has min-eccentricity at most twice the true min-radius R then we have obtained a 2-approximation. (We do not know R in advance but we repeatedly run the algorithm with different values of R to perform a binary search on R.)

Otherwise, we will define intervals in the ordering such that the min-center c cannot be contained in any of these intervals. A key observation is that if there is a pair of vertices (u, v) such that u appears before v in the topological ordering and $d(u, v) > 2R$, then the min-center c cannot fall between u and v in the topological ordering. This is because if it did, then $d(u, c) \leq R$ and $d(c, v) \leq R$, so $d(u, v) \leq 2R$, a contradiction. We define the intervals that cannot contain c as follows: for all $v \in W$ we let a_v be the first vertex in the ordering such that $d(a_v, v) > 2R$ (if it exists, otherwise $a_v = v$) and define b_v to be the last vertex in the ordering such that $d(v, b_v) > 2R$ (if it exists, otherwise $b_v = v$). Then, the key observation implies that c cannot fall in the interval $[a_v, b_v]$ in the ordering. Now, we have a set of possibly overlapping intervals that cannot contain c. We take the union of these intervals to get a set of disjoint intervals that cannot contain c.

Every vertex u that does not appear in such an interval, falls between two consecutive intervals I_u and I'_u. We define the partial search graph of u to be the graph induced by the set of vertices in I_u or I'_u or between I_u and I'_u. After performing the partial searches, the algorithm returns 3 times the minimum min-radius of all partial search graph. Next we give the idea of the analysis, which demystifies the factor of 3 in the returned value.

We claim that if the min-eccentricity of a vertex with respect to its partial search graph is at most R, then its min-eccentricity with respect to the full graph is at most $3R$, and the min-eccentricity of the true min-center with respect to its partial search graph is at most R (because for any path in a DAG whose starting and ending points are in a certain interval, every vertex in the path is in that interval). Thus, assuming the claim, $3R$ is a 3-approximation for the min-radius. We now outline the proof of the claim. Let u be the min-center with the minimum min-radius R of all partial search graphs. Let $v \in W$ such that a_v is the first vertex (in the topological order) of I_u, then $v \in I_u$ and $d(v, u) \leq R$. Furthermore, by the definition of a_v, all vertices that appear before the beginning of the interval I_u have distance at most $2R$ to v, and thus distance at most $3R$ to u. A symmetric argument holds for vertices that appear after the end of the interval I'_u. Hence the min-eccentricity of u with respect to the full graph is at most $3R$.

This algorithm runs in time $O(m\sqrt{n})$ because the vertices of W are evenly spaced so there are no more than $\sqrt{n}$ vertices between each pair of consecutive intervals. This implies that in the partial searches, each edge is only scanned $O(\sqrt{n})$ times. (Furthermore, repeatedly running the algorithm to binary search for R adds a logarithmic factor to the runtime.)

Algorithm for general graphs

We now give a high-level outline of our $\tilde{O}(m\sqrt{n})$ time 3-approximation algorithm for Min-Radius. This algorithm is much more delicate than our Min-Diameter algorithm, hence more of the details are deferred to the full description. We begin by running Dijkstra's algorithm

from a set W of $\tilde{O}(\sqrt{n})$ randomly sampled vertices to recursively partition the vertices into S_v and T_v as outlined in Section 1.2. This defines an initial DAG-like structure, however our analysis requires constructing a much more refined DAG-like structure.

Perhaps counter-intuitively, it makes sense to place vertices that are *far* from each other in the graph *close* to each other in the DAG-like structure. The reason for this is illuminated by the Min-Radius algorithm on DAGs, in which we find pairs of vertices u, v that are far from each other and apply the key observation that the min-center cannot be between u and v in the topological ordering. Intuitively, it is as if we collapse the interval between u and v in the DAG since we do not have to search within this interval for the min-center. An analogous key observation is true for general graphs: if there is a pair of vertices (u, v) with $d_{min}(u, v) > 2R$, then either $c \in S_u \cap S_v$ or $c \in T_u \cap T_v$. This is because if $c \in T_u \cap S_v$, then $d(u, c) \leq R$ and $d(c, v) \leq R$ so $d(u, v) \leq 2R$, a contradiction; the last case $c \in S_u \cap T_v$ is symmetric. In our algorithm for general graphs, we ensure that far vertices are near each other in the DAG-like structure by doing the following: we let the *far graph* G_{far} be an undirected graph on V with an edge between $u \in W$ and $v \in V$ if $d_{min}(u, v) > 2R$. All vertices in W that are in the same connected component in G_{far} will be grouped in the DAG-like structure. We let F_i be the set of vertices in W that are in the i^{th} connected component of G_{far}.

To construct the DAG-like structure, we show that precisely chosen groups of F_is can be merged to create *supercomponents*, which constitute a DAG-like structure in the following sense: there is an ordering of supercomponents such that for every pair of vertices $u, v \in W$ where the supercomponent containing u appears before that containing v, $d(u, v)$ is small and $d(v, u)$ is large. Specifically, we define the *close graph* H whose vertex set is the set of F_is. We add a directed edge between a pair of vertices in H if there exists a short path (length $\leq 5R$) between the corresponding F_is. Then we merge all F_is that appear in the same strongly connected component of H into a supercomponent. This contraction of strongly connected components of H results in a DAG, which defines the ordering of the supercomponents.

Now that we have arranged the vertices in W into a DAG-like structure, we would like to fit every vertex in the graph into this structure. Based on the precise way that we have defined the supercomponents, we can use an intricate argument to show *roughly* the following property: for every vertex v there exists an i such that for every vertex $u \in W$ in the first i supercomponents, $d(u, v)$ is small and for every vertex $u \in W$ in the remaining supercomponents, $d(v, u)$ is small.

After fitting every vertex into the refined DAG-like ordering, we can define each partial search graph to be an interval in the ordering that is large enough to contain several supercomponents. In the algorithm for DAGs, there were two important properties of the partial search graphs: (1) the min-eccentricity of the true min-center with respect to its partial search graph is at most R, and (2) if the min-eccentricity of a vertex with respect to its partial search graph is at most R then its min-eccentricity with respect to the full graph is at most $3R$. We show that due to the precise structure of the supercomponents, refinements of properties (1) and (2) are also true for general graphs.

Intuitively, property (1) is roughly true because for every pair of vertices $u, v \in W$ such that u's supercomponent appears before v's in the ordering, $d(v, u) > 5R$, since otherwise this pair of supercomponents would be in the same strongly connected component of H and would have been merged into a single supercomponent. This implies that paths of length at most R to or from the min-center cannot stray beyond its partial search graph. Intuitively, property (2) is roughly true because for every pair of vertices $u, v \in W$ such that u's supercomponents appears before v's in the ordering, $d(u, v) \leq 2R$ because otherwise, u

and v would be in the same component of G_{far} and thus be in the same supercomponent. Thus, like the argument for DAGs, for all u, all vertices that appear before u's partial search graph G_u have distance at most $2R$ to each supercomponent in G_u, and thus distance at most $3R$ to u. A symmetric argument holds for vertices after u in the ordering.

2.3 Min-Eccentricities

Our Min-Eccentricities algorithm is a modification of our Min-Radius algorithm. In our Min-Radius algorithm, we identify a vertex whose min-eccentricity is at most about $3R$, where R is the true min-radius. In our Min-Eccentricities algorithm, we show that with some extra bookkeeping, the algorithm can identify *all* vertices with min-eccentricity at most about 5ρ for any ρ. We run the algorithm repeatedly, increasing ρ by a factor of $(1+\delta)$ at each execution until we have estimated the min-eccentricity of every vertex.

The major modification of the Min-Radius algorithm here is that if one of the vertices that we run Dijkstra from has min-eccentricity at most 3ρ, we cannot stop running the algorithm, as we can in the Min-Radius algorithm. Instead, we use this vertex as a tool to find vertices with min-eccentricity at most 5ρ.

3 Preliminary Graph Partitioning

In this section we describe a graph partitioning procedure we use as a first step in our Min-Diameter, Min-Radius, and Min-Eccentricities algorithms. The goal of this partitioning is to define a DAG-like structure in general directed graphs.

▶ **Definition 5.** *Assign each vertex a unique ID from* $[n]$*. For each vertex* v*, let* $S_v = \{u \in V : d(u,v) < d(v,u) \vee [d(u,v) = d(v,u) \wedge ID(u) < ID(v)]\}$*. Let* $T_v = V \setminus (S_v \cup \{v\})$*.*

The runtime of our algorithms relies on whether the partition into S_v and T_v is *balanced*. Using the observation that if $u \in S_v$, then $v \in T_u$, the following lemma shows that for most vertices, the partition is indeed approximately balanced.

▶ **Lemma 6.** *For any* n*-vertex graph, there are* $> \frac{n}{2}$ *vertices* v *such that* $\frac{|S_v|}{8} \leq |T_v| \leq 8 \cdot |S_v|$*.*

More generally, for any $U \subseteq V$*, there are more than* $\frac{|U|}{2}$ *vertices* $v \in U$ *such that* $\frac{|S_v \cap U|}{8} \leq |T_v \cap U| \leq 8|S_v \cap U|$*.*

Lemma 6 is proved in the full version [16]. Next, we describe how we use Lemma 6 to recursively construct a balanced partition of the vertices into a given number of of sets.

▶ **Lemma 7.** *Given an* n*-vertex graph* $G = (V, E)$ *and a constant* $c > 0$*, in* $\tilde{O}(mn^{1-c})$ *time one can split* V *into disjoint sets* $W, V_1, V_2, \ldots, V_{q+1}$*, where* $q = |W| = n^{1-c}$*, such that with high probability:*

1. *for all* i*,* $|V_i| = \Theta(\frac{n}{q})$*;*
2. *for all* $i \neq j$*, there exists* $w \in W$ *such that either* $V_i \subseteq S_w$,$V_j \subseteq T_w$*, or* $V_i \subseteq T_w$,$V_j \subseteq S_w$*;*
3. *for all* $U \subseteq W$*, let* $V_U = \left(\bigcap_{w \in U} S_w\right) \bigcap \left(\bigcap_{w \in W \setminus U} T_w\right)$*, then* $V_U \subseteq V_i$ *for some* $i \in [q+1]$*.*

Proof. We begin with $W = \emptyset$ and we will iteratively populate W with vertices. We let $\mathcal{V}_0 = \{V\}$ and for all $i \in [q]$ when we add the i^{th} vertex to W, we will construct $\mathcal{V}_i$ from $\mathcal{V}_{i-1}$ by partitioning the largest set in $\mathcal{V}_{i-1}$ into two parts. After adding q vertices to W we will have constructed $\mathcal{V}_q = \{V_1 \ldots V_{q+1}\}$.

For all $i \in [q]$, let A_i, B_i be the largest and smallest sets in $\mathcal{V}_i$, respectively.

We describe how to construct W and $\mathcal{V}_q$ inductively. Suppose $|W| = r - 1$ and we have constructed $\mathcal{V}_{r-1}$. By Lemma 6, if we randomly sample $O(\log^2 n)$ vertices from A_{r-1}, with probability at least $1 - 2^{-\log^2 n} = 1 - n^{-\log n}$ we will sample a vertex w_r such that $A_S = A_{r-1} \cap S_{w_r}$ and $A_T = A_{r-1} \cap T_{w_r}$ differ by a factor of at most 8. We add w_r to W and let $\mathcal{V}_r = \mathcal{V}_{r-1} \cup \{A_S, A_T\} \setminus \{A_{r-1}\}$.

By union bound over the $q = n^{1-c}$ partitionings, with probability at least $1 - n^{1-c-\log n}$, every partitioning produces two sets that differ in size by a factor of at most 8.

We prove property 1 by induction on $|W| = r$. Specifically, we will show that for all $r \in [q]$, $|A_r| \leq 9|B_r|$. This implies that $|A_q| = O(|B_q|)$, and property 1 follows. Lemma 6 implies that $|A_1| \leq 9|B_1|$. Assume inductively that $|A_{r-1}| \leq 9|B_{r-1}|$. Since no subset grows in size, $|A_r| \leq |A_{r-1}|$ and $|B_r| \leq |B_{r-1}|$. If $|B_r| = |B_{r-1}|$, then $|A_r| \leq |A_{r-1}| \leq 9|B_{r-1}| = 9|B_r|$. Otherwise, $|B_r| < |B_{r-1}|$, which implies that B_r is one of the two sets obtained by partitioning A_{r-1}. Then by Lemma 6, $|A_{r-1}| \leq 9|B_r|$. Hence $|A_r| \leq |A_{r-1}| \leq 9|B_r|$, completing the induction.

Property 2 follows from the partitioning procedure: for any $i \neq j$, if for all $w \in W$, $V_i, V_j \subseteq S_w$ or $V_i, V_j \subseteq T_w$ then $V_i \cup V_j$ would never have been partitioned.

Property 3 also follows from the partitioning procedure: observe that for all $w \in W$ and all $U \subseteq W$, $V_U \subseteq S_w$ or $V_U \subseteq T_w$, so V_U is never partitioned and thus $V_U \subseteq V_i$ for some $i \in [q+1]$.

Since we sample $n^{1-c}\log^2 n$ vertices and for all v finding S_v, T_v takes $O(m)$ time, the runtime is $\tilde{O}(mn^{1-c})$. ◀

4 Min-Diameter Algorithm

Throughout this section, let D be the min-diameter, and let s^*, t^* the endpoints of the min-diameter. In this section we prove the time/accuracy trade-off theorem for Min-Diameter.

▶ **Theorem 8.** *For any integer $0 < \ell \leq O(\log n)$, there is an $\tilde{O}(mn^{1/(\ell+1)})$ time randomized algorithm that, given a directed weighted graph G with edge weights non-negative and polynomial in n, can output an estimate $\tilde{D}$ such that $D/(4\ell - 1) \leq \tilde{D} \leq D$ with high probability, where D is the min-diameter of G.*

We first prove a special case of Theorem 8 where $\ell = 1$, and the rest of the proof can be found in the full version [16].

4.1 An $\tilde{O}(m\sqrt{n})$ time 3-approximation

▶ **Theorem 9** (Theorem 8 with $\ell = 1$)**.** *There is an $\tilde{O}(m\sqrt{n})$ time randomized algorithm, that given a directed weighted graph $G = (V, E)$ with edge weights non-negative and polynomial in n, can output an estimate $\tilde{D}$ such that $D/3 \leq \tilde{D} \leq D$ with high probability, where D is the min-diameter of G.*

4.1.1 Algorithm Description

Applying Lemma 7 with $q = \sqrt{n}$ we obtain a partition of the vertices into $W, V_1, V_2, \ldots, V_{\sqrt{n}+1}$.

We perform Dijkstra's algorithm from every vertex in W and define $D' = \max_{w \in W} \varepsilon(w)$. We will later show that D' is a good approximation of the Min-Diameter when s^* and t^* are not in the same vertex set V_i.

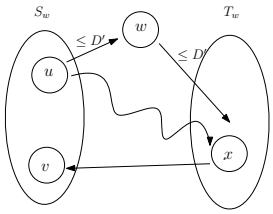


Figure 1 The case where $u, v \in S_w$ and the shortest path from u to v contains a node $x \in T_w \cup \{w\}$.

For every $i \in [\sqrt{n}+1]$, define $W_i^S = \{w \in W : V_i \subseteq S_w\}$, and $W_i^T = \{w \in W : V_i \subseteq T_w\}$. Then, for every i, we construct two graphs G_i^S and G_i^T. The first graph G_i^S contains all vertices of V_i and an additional node w_i^S. It has the following edges:

1. For every directed edge $(u, v) \in E$ such that $u, v \in V_i$, add this edge to G_i^S.
2. Add a directed edge from w_i^S to every $v \in V_i$, with weight $\max\left\{\min_{w \in W_i^S} d(w, v) - D', 0\right\}$, and a directed edge from every $v \in V_i$ to w_i^S with weight 0.

The second graph G_i^T is symmetric to G_i^S. It contains all vertices in V_i and an additional node w_i^T. It has the following edges:

1. For every directed edge $(u, v) \in E$ such that $u, v \in V_i$, add this edge to G_i^T.
2. Add a directed edge from every $v \in V_i$ to w_i^T, with weight $\max\left\{\min_{w \in W_i^T} d(v, w) - D', 0\right\}$, and add a directed edge from w_i^T to every $v \in V_i$ with weight 0.

For all i, we run an exact all-pairs shortest paths algorithm on G_i^S and G_i^T. This allows us to compute for all i and all $u, v \in V_i$ the quantity $\min\{d_{G_i^S}(u, v), d_{G_i^T}(u, v)\}$, which we denote by $d_i'(u, v)$.

We choose the larger between D' and $\max_{i \in [\sqrt{n}+1], u, v \in V_i} \min\{d_i'(u, v), d_i'(v, u)\}$ as our final estimate for the min-diameter.

4.1.2 Analysis

The following lemma will be used to show that D' is a good estimate for the min-diameter if s^* and t^* happen to fall into different sets V_i

▶ **Lemma 10.** *For all vertices v, if either $s^* \in S_v$, $t^* \in T_v$, or $t^* \in S_v$, $s^* \in T_v$, then $\varepsilon(v) \geq D/2$.*

Proof. We only consider the case when $s^* \in S_v$ and $t^* \in T_v$ as the other case is symmetric. By way of contradiction, assume that $\varepsilon(v) < D/2$, then we have $d_{\min}(s^*, v) < D/2$ and $d_{\min}(t^*, v) < D/2$. Since $s^* \in S_v$, $d(s^*, v) = d_{\min}(s^*, v) < D/2$; similarly, since $t^* \in T_v$, $d(v, t^*) = d_{min}(t^*, v) < D/2$. Therefore, by the triangle inequality, $d(s^*, t^*) < D$, a contradiction. ◀

The next two lemmas are used for the case where s^* and t^* fall into the same set V_i.

▶ **Lemma 11.** *For every i, and every pair of vertices $u, v \in V_i$, $d_i'(u, v) \leq d(u, v)$; that is, $\min\{d_{G_i^S}(u, v), d_{G_i^T}(u, v)\} \leq d(u, v)$.*

Proof. Take any shortest path in the original graph G from u to v. If this path does not leave V_i, then this path also exists in G_i^S and G_i^T, and thus the inequality is true.

It remains to prove for the case when the shortest u, v path in the original graph leaves V_i. Let $x \notin V_i$ be any vertex on a shortest u, v path. By Lemma 7, property 2, there exists $w \in W$ such that $x \in S_w \cup \{w\}$ and $V_i \subseteq T_w$, or $x \in T_w \cup \{w\}$ and $V_i \subseteq S_w$. We first assume $x \in T_w \cup \{w\}$ and $V_i \subseteq S_w$ as shown in Figure 1, and the other case is symmetric.

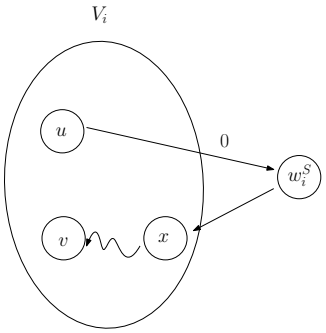


Figure 2 A shortest u, v path in G_i^S that contains w_i^S. The path goes from u, directly to w_i^S using a weight 0 edge, then directly to a vertex x, and finally reaches v.

Since x is on the shortest path from u to v, we have $d(u, v) \geq d(x, v)$. Also, we have $d(w, x) \leq D'$, by definition of D'. Therefore,

$$d(u, v) \geq d(x, v) \geq d(x, v) + (d(w, x) - D') \geq d(w, v) - D'. \tag{1}$$

Now consider the path $u \to w_i^S \to v$ in G_i^S. The first part $u \to w_i^S$ costs 0, because there is an edge from u to w_i^S with weight 0; the second part $w_i^S \to v$ costs at most $\max\{0, d(w, v) - D'\}$. If $d(w, v) < D'$, then $d'_i(u, v) \leq d_{G_i^S}(u, v) = 0 \leq d(u, v)$; otherwise, $d'_i(u, v) \leq d_{G_i^S}(u, v) \leq d(w, v) - D' \leq d(u, v)$, where the last step is Equation 1.

When $x \in S_w \cup \{w\}$, and $V_i \subseteq T_w$, we have a symmetric argument: $d(u, v) \geq d(u, x) \geq d(u, x) + (d(x, w) - D') \geq d(u, w) - D'$. Consider the path $u \to w_i^T \to v$ in G_i^T. The second part $w_i^T \to v$ costs 0, because there is an edge from w_i^T to v with weight 0; the first part $u \to w_i^T$ costs at most $\max\{0, d(u, w) - D'\}$. If $d(u, w) < D'$, then $d'_i(u, v) \leq d_{G_i^T}(u, v) = 0 \leq d(u, v)$; otherwise, $d'_i(u, v) \leq d_{G_i^T}(u, v) \leq d(u, w) - D' \leq d(u, v)$. ◀

▶ **Lemma 12.** *For every i, and every pair of vertices $u, v \in V_i$, $d'_i(u, v) \geq d(u, v) - 2D'$; that is, $d_{G_i^S}(u, v) \geq d(u, v) - 2D'$ and $d_{G_i^T}(u, v) \geq d(u, v) - 2D'$.*

Proof. We only provide full proof for $d_{G_i^S}(u, v) \geq d(u, v) - 2D'$. The inequality for G_i^T can be proved by a symmetrical argument. If the shortest path from u to v in G_i^S does not contain w_i^S, then this path also exists in the original graph G, and thus the inequality is true.

Otherwise, the shortest path from u to v in G_i^S contains w_i^S, as shown in Figure 2. All edges on the shortest path from w_i^S to v exist in the original graph G except for the first edge from w_i^S to some node x, since a shortest path cannot use the vertex w_i^S more than once. That is, $d_{G_i^S}(x, v) = d(x, v)$.

By the definition of w_i^S and the edges incident to it, there exists a $w \in W_i^S$ such that $d(w, x) \leq d_{G_i^S}(w_i^S, x) + D'$. Thus, we have

$$\begin{aligned}
d_{G_i^S}(u, v) &= d_{G_i^S}(u, w_i^S) + d_{G_i^S}(w_i^S, x) + d_{G_i^S}(x, v) \\
&= d_{G_i^S}(w_i^S, x) + d_{G_i^S}(x, v) && \text{since } d_{G_i^S}(u, w_i^S) = 0 \text{ by construction} \\
&= d_{G_i^S}(w_i^S, x) + d(x, v) && \text{from argument above} \\
&\geq d(w, x) - D' + d(x, v) && \text{by the definition of } w \\
&\geq d(w, v) - D' && \text{by the triangle inequality} \\
&\geq (d(w, v) - D') + (d(u, w) - D') && \text{since } d(u, w) \leq D' \text{ by definition} \\
&\geq d(u, v) - 2D' && \text{by the triangle inequality}
\end{aligned}$$

◀

We are now ready to prove our approximation ratio guarantee: $D/3 \leq \tilde{D} \leq D$. Clearly $D' \leq D$ because D' is the min-eccentricity of a vertex. By Lemma 11 $\max_{i,u\in V_i,v\in V_i} \min\{d'_i(u,v), d'_i(v,u)\} \leq \max_{i,u\in V_i,v\in V_i} d_{min}(u,v) \leq D$. Therefore, we never over estimate the Min-Diameter.

If $s^* \in W$ or $t^* \in W$, then since we run Dijkstra from all vertices in W we have $D' = D$. So assuming that $s^*, t^* \notin W$, we have two cases.

Case 1: s^* and t^* are not in the same vertex set V_i. By Lemma 7, property 2, there exists $w \in W$ such that one of s^* and t^* is in S_w and the other is in T_w, so by Lemma 10, $\varepsilon(w) \geq D/2$. Since $D' \geq \varepsilon(w)$, we have $D' \geq D/2$.

Case 2: s^* and t^* are in the same vertex set V_i for some i. By Lemma 12, $\min(d'_i(s^*,t^*), d'_i(t^*,s^*)) \geq d_{min}(s^*,t^*) - 2D' = D - 2D'$. Since $\max\{D-2D', D'\} \geq D/3$, we get a 3-approximation.

Runtime analysis

It takes $\tilde{O}(m\sqrt{n})$ time to perform the partitioning from Lemma 7 and to perform Dijkstra's algorithm from all $w \in W$ since $|W| = O(\sqrt{n})$.

For all i, the number of vertices in G_i^S is $|V_i| + 1 = O(\sqrt{n})$ with high probability by property 1 of Lemma 7 and the number of edges is $m_i + O(\sqrt{n})$ where m_i is the number of edges in the graph induced by V_i. Hence we can run an all-pairs shortest paths algorithm on G_i^S in time $\tilde{O}((m_i + \sqrt{n})\sqrt{n})$. Summing over all i gives us $\tilde{O}(m\sqrt{n})$. The same analysis also works for G_T^i.

References

1. Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. Subcubic Equivalences Between Graph Centrality Problems, APSP and Diameter. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1681–1697, 2015.
2. Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and Fixed Parameter Subquadratic Algorithms for Radius and Diameter in Sparse Graphs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 377–391, 2016.
3. D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast Estimation of Diameter and Shortest Paths (Without Matrix Multiplication). *SIAM J. Comput.*, 28(4):1167–1181, 1999.
4. Arturs Backurs, Liam Roditty, Gilad Segal, Virginia Vassilevska Williams, and Nicole Wein. Towards tight approximation bounds for graph diameter and eccentricities. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 267–280, 2018.
5. B. Ben-Moshe, B. K. Bhattacharya, Q. Shi, and A. Tamir. Efficient algorithms for center problems in cactus networks. *Theoretical Computer Science*, 378(3):237–252, 2007.
6. P. Berman and S. P. Kasiviswanathan. Faster Approximation of Distances in Graphs. In *Proc. WADS*, pages 541–552, 2007.
7. Michele Borassi, Pierluigi Crescenzi, Michel Habib, Walter A. Kosters, Andrea Marino, and Frank W. Takes. Fast diameter and radius BFS-based computation in (weakly connected) real-world graphs: With an application to the six degrees of separation games. *Theoretical Computer Science*, 2015. accepted.
8. Massimo Cairo, Roberto Grossi, and Romeo Rizzi. New Bounds for Approximating Extremal Distances in Undirected Graphs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 363–376, 2016.

9 T. M. Chan. All-pairs shortest paths for unweighted undirected graphs in $o(mn)$ time. *ACM Transactions on Algorithms*, 8(4):34, 2012.

10 Shiri Chechik, Daniel H. Larkin, Liam Roditty, Grant Schoenebeck, Robert Endre Tarjan, and Virginia Vassilevska Williams. Better Approximation Algorithms for the Graph Diameter. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1041–1052, 2014.

11 V. Chepoi, F. Dragan, and Y. Vaxès. Center and diameter problems in plane triangulations and quadrangulations. In *Proc. SODA*, pages 346–355, 2002.

12 V. Chepoi and F. F. Dragan. A Linear-Time Algorithm for Finding a Central Vertex of a Chordal Graph. In *ESA*, pages 159–170, 1994.

13 F. R. K. Chung. Diameters of graphs: Old problems and new results. *Congr. Numer.*, 60:295–317, 1987.

14 D.G. Corneil, F.F. Dragan, M. Habib, and C. Paul. Diameter determination on restricted graph families. *Discr. Appl. Math.*, 113:143–166, 2001.

15 L. Cowen and C. Wagner. Compact roundtrip routing for digraphs. In *SODA*, pages 885–886, 1999.

16 Mina Dalirrooyfard, Virginia Vassilevska Williams, Nikhil Vyas, Nicole Wein, Yinzhan Xu, and Yuancheng Yu. Approximation Algorithms for Min-Distance Problems, 2019. `arXiv:1904.11606`.

17 D. Dvir and G. Handler. The absolute center of a network. *Networks*, 43:109–118, 2004.

18 D. Eppstein. Subgraph isomorphism in planar graphs and related problems. *J. Graph Algorithms and Applications*, 3(3):1–27, 1999.

19 Silvio Frischknecht, Stephan Holzer, and Roger Wattenhofer. Networks cannot compute their diameter in sublinear time. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1150–1162. SIAM, 2012.

20 S.L. Hakimi. Optimum location of switching centers and absolute centers and medians of a graph. *Oper. Res.*, 12:450–459, 1964.

21 Seth Pettie. A faster all-pairs shortest path algorithm for real-weighted sparse graphs. In *International Colloquium on Automata, Languages, and Programming*, pages 85–97. Springer, 2002.

22 Seth Pettie and Vijaya Ramachandran. A Shortest Path Algorithm for Real-Weighted Undirected Graphs. *SIAM J. Comput.*, 34(6):1398–1431, 2005. `doi:10.1137/S0097539702419650`.

23 Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, STOC '13, pages 515–524, New York, NY, USA, 2013. ACM. `doi:10.1145/2488608.2488673`.

24 O. Weimann and R. Yuster. Approximating the Diameter of Planar Graphs in Near Linear Time. In *Proc. ICALP*, 2013.

25 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 664–673, 2014.

26 C. Wulff-Nilsen. Wiener index, diameter, and stretch factor of a weighted planar graph in subquadratic time. *Technical report, University of Copenhagen*, 2008.

27 Raphael Yuster. Computing the diameter polynomially faster than APSP. *arXiv preprint*, 2010. `arXiv:1011.6181`.

Tight Approximation Algorithms for Bichromatic Graph Diameter and Related Problems

Mina Dalirrooyfard
MIT, Cambridge, MA, USA
minad@mit.edu

Virginia Vassilevska Williams
MIT, Cambridge, MA, USA
virgi@mit.edu

Nikhil Vyas
MIT, Cambridge, MA, USA
nikhilv@mit.edu

Nicole Wein
MIT, Cambridge, MA, USA
nwein@mit.edu

Abstract

Some of the most fundamental and well-studied graph parameters are the Diameter (the largest shortest paths distance) and Radius (the smallest distance for which a "center" node can reach all other nodes). The natural and important ST-variant considers two subsets S and T of the vertex set and lets the ST-diameter be the maximum distance between a node in S and a node in T, and the ST-radius be the minimum distance for a node of S to reach all nodes of T. The *bichromatic* variant is the special case in which S and T partition the vertex set.

In this paper we present a comprehensive study of the *approximability* of ST and Bichromatic Diameter, Radius, and Eccentricities, and variants, in graphs with and without directions and weights. We give the first nontrivial approximation algorithms for most of these problems, including time/accuracy trade-off upper and lower bounds. We show that nearly *all* of our obtained bounds are tight under the Strong Exponential Time Hypothesis (SETH), or the related Hitting Set Hypothesis.

For instance, for Bichromatic Diameter in undirected weighted graphs with m edges, we present an $\tilde{O}(m^{3/2})$ time [1] 5/3-approximation algorithm, and show that under SETH, neither the running time, nor the approximation factor can be significantly improved while keeping the other unchanged.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Approximation algorithms analysis; Theory of computation → Problems, reductions and completeness; Theory of computation → Shortest paths

Keywords and phrases approximation algorithms, fine-grained complexity, diameter, radius, eccentricities

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.47

Category Track A: Algorithms, Complexity and Games

Related Version `https://arxiv.org/abs/1904.11601`

Funding *Mina Dalirrooyfard*: Supported by an Akamai Presidential Fellowship and NSF Grant CCF-6936484.
Virginia Vassilevska Williams: Supported by an NSF CAREER Award, NSF Grants CCF-1417238, CCF-1528078 and CCF-1514339, a BSF Grant BSF:2012338 and a Sloan Research Fellowship.
Nikhil Vyas: Supported by an Akamai Presidential Fellowship and NSF Grant CCF-1552651.
Nicole Wein: Supported by an NSF Graduate Fellowship and NSF Grant CCF-1514339.

Acknowledgements The authors would like to thank Arturs Backurs for discussions during the early stages of this work.

[1] $\tilde{O}$ notation hides polylogarithmic factors.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 47; pp. 47:1–47:15

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

1 Introduction

A fundamental and very well studied problem in algorithms is the Diameter of a graph, where the output is the largest (shortest path) distance over all pairs of vertices. Over the years many different algorithms have been developed for the problem, both in theory (e.g. [3, 21, 24, 8, 4]) and in practice (e.g. [10, 25, 20]).

A very natural variant is the so called ST-Diameter problem [4]: given a graph and two subsets S and T of its vertex set, determine the largest distance between a vertex of S and a vertex of T. In the Subset version of ST-Diameter, we have $S = T$. Bichromatic Diameter is the version of ST-Diameter for which S and T partition the vertex set. Besides Diameter, the Radius (the smallest distance for which a "center" node can reach all other nodes) and Eccentricities (the largest distance out of every vertex) problems are also very well studied, and analogous ST, Subset, and Bichromatic versions are easy to define.

All of these parameters are simple to compute by computing all pairwise distances in the graph, i.e. by solving All-Pairs Shortest Paths (APSP). In sparse n-node graphs, where the number of edges m is $\tilde{O}(n)$, APSP still needs $\Omega(n^2)$ time, as this is the size of the output, whereas it is apriori unclear whether this much time is needed for computing the Diameter, Radius and Eccentricities or their ST and bichromatic variants, as the output is small.

A related extremely well-studied problem in computational geometry is Bichromatic Diameter on point sets (commonly known as Bichromatic Farthest Pair), where one seeks to determine the farthest pair of points in a given set of points in space (see e.g. [29, 13, 28, 2, 17]). Another related problem is the Subset version of spanners (e.g. [19, 11]), as well as the ST version of spanners (e.g. [9, 18]). Furthermore, the ST, Subset, and Bichromatic versions of many problems have been of great interest; for instance Steiner Tree, Subset TSP, and a number of problems in computational geometry such as Bichromatic Matching (e.g. [16]) and Bichromatic Line Segment Intersection (e.g. [7]).

There are several known approximation algorithms for the standard version of Diameter, most of which have been developed in the last 6 years. Trivially, running Dijkstra's algorithm from an arbitrary vertex gives a simple $\tilde{O}(m)$ time 2-approximation algorithm for directed and weighted graphs. Non-trivial algorithms achieve an improved approximation factor with an increased runtime: Building on Aingworth et al. [3], Roditty and Vassilevska W. [24] showed for instance that an "almost" 1.5 approximation for Diameter can be computed in $\tilde{O}(m\sqrt{n})$ time in m-edge n-vertex directed weighted graphs – the approximation factor is 1.5 if the Diameter is divisible by 3, and there is a slight additive error otherwise. Chechik et al. [8] gave a true 1.5 approximation at the expense of increasing the runtime to $\tilde{O}(mn^{2/3})$, and Cairo, Grossi and Rizzi [5] generalized the approach giving an $\tilde{O}(mn^{1/(k+1)})$ time, "almost" $2 - 1/2^k$ approximation algorithm for all $k \geq 1$ which works only in undirected graphs.

In STOC'18, Backurs et al. [4] gave the first non-trivial approximation algorithms for ST-Diameter: an $\tilde{O}(m^{3/2})$ time 2-approximation and an $\tilde{O}(m)$ time 3-approximation. They also showed that these algorithms cannot be improved significantly, unless the Strong Exponential Time Hypothesis (SETH) fails. Backurs et al. did not provide algorithms for ST-Eccentricities or ST-Radius, and they did not study the natural Subset and Bichromatic versions. They also only focused on undirected graphs.

We study the following natural and fundamental questions:

How well can ST-Eccentricities and ST-Radius be approximated? Are any interesting approximation algorithms possible for directed graphs for any of the ST-variants? Does the approximability of the problems change when one turns to the Subset versions in which $S = T$, or the Bichromatic versions in which S and T are required to partition the vertex set?

1.1 Our Results

We present a comprehensive study of the approximability of the ST, Subset and Bichromatic variants of the Diameter, Radius and Eccentricities problems in graphs, both with and without directions and weights. We obtain the first non-trivial approximation algorithms for most of these problems, including time/accuracy trade-off upper and lower bounds. We show that nearly *all* of our approximation algorithms are tight under SETH (or under the related Hitting Set Hypothesis for Radius). Additionally, we study a parameterized version of these problems.

Our results are summarized in Tables 1-4.

Table 1 Bichromatic undirected results. All of our parameterized algorithms and near-linear time algorithms, except for directed Subset Radius and Eccentricities, are deterministic. The rest are randomized and work with high probability[2]. Our lower bounds for Diameter and Eccentricities are under SETH and our lower bounds for Radius are under the Hitting Set (HS) Hypothesis, defined later. All of our lower bounds hold even for unweighted graphs. The trade-off lower bounds in terms of k hold for any integer $k \geq 2$. δ is any constant > 0. B and B' are parameters defined in our parameterized algorithms. The lower bound constructions for the parameterized algorithms have $|B| = \tilde{O}(1)$.
* Multiplicative approximation factor is tight, but not runtime.

	Upper Bounds			Lower Bounds	
Problem	Runtime	Approx.	Comments	Runtime	Approx.
Diameter	$O(m + n \log n)$	almost 2	unweighted, **tight**	$m^{1+o(1)}$	$2 - \delta$
	$\tilde{O}(m\sqrt{n})$	almost 5/3	unweighted, nearly **tight**	$m^{\frac{k}{k-1}-o(1)}$	$2 - \frac{1}{2k-1} - \delta$
	$\tilde{O}(m^{3/2})$	5/3	weighted, **tight**	"	"
	$O(m\|B\|)$	almost 3/2	unweighted, **tight***	$m^{2-o(1)}$	$3/2 - \delta$
Radius	$O(m + n \log n)$	almost 2	unweighted		
	$\tilde{O}(m\sqrt{n})$	almost 5/3	unweighted, nearly **tight***	$m^{2-o(1)}$	$5/3 - \delta$
	$\tilde{O}(m^{3/2})$	5/3	weighted, **tight***	"	"
	$O(m\|B\|)$	almost 3/2	unweighted, **tight***	$m^{2-o(1)}$	$3/2 - \delta$
Eccentricities	$O(m + n \log n)$	3	weighted, **tight**	$m^{1+o(1)}$	$3 - \delta$
	$\tilde{O}(m\sqrt{n})$	almost 2	unweighted, nearly **tight**	$m^{\frac{k}{k-1}-o(1)}$	$3 - 2/k - \delta$
	$\tilde{O}(m^{3/2})$	2	weighted, **tight**	"	"
	$O(m\|B\|)$	almost 5/3	unweighted, **tight***	$m^{2-o(1)}$	$5/3 - \delta$

Table 2 Bichromatic directed results. See caption of Table 1.

	Upper Bounds			Lower Bounds	
Problem	Runtime	Approx.	Comments	Runtime	Approx.
Diameter	$\tilde{O}(m^{3/2})$	2	weighted, **tight***	$m^{2-o(1)}$	$2 - \delta$
	$O(m\|B'\|)$	almost 3/2	unweighted, **tight***	$m^{2-o(1)}$	$3/2 - \delta$
Radius	N/A	N/A	weighted, **tight**	$m^{2-o(1)}$	any finite
Eccentricities	N/A	N/A	weighted, **tight**	$m^{2-o(1)}$	any finite

All our algorithms in m-edge, n-node graphs, run in $\tilde{O}(m^{3/2})$ time or in $\tilde{O}(m\sqrt{n})$ time when a small additive error is allowed. For sparse graphs the $m^{3/2}$ runtime beats the fastest

[2] *with high probability* means with probability at least $1 - 1/n^c$ for all constants c.

Table 3 ST undirected results. See caption of Table 1.

	Upper Bounds			Lower Bounds	
Problem	Runtime	Approx.	Comments	Runtime	Approx.
Diameter[4]	$O(m + n \log n)$	3	weighted, **tight**	$m^{1+o(1)}$	$3 - \delta$
	$\tilde{O}(m\sqrt{n})$	almost 2	unweighted, nearly **tight**	$m^{\frac{k}{k-1}-o(1)}$	$3 - 2/k - \delta$
	$\tilde{O}(m^{3/2})$	2	weighted, **tight**	"	"
Radius	$O(m + n \log n)$	3	weighted		
	$\tilde{O}(m\sqrt{n})$	almost 2	unweighted, nearly **tight***	$m^{2-o(1)}$	$2 - \delta$
	$\tilde{O}(m^{3/2})$	2	weighted, **tight***	"	"
Eccentricities	$O(m + n \log n)$	3	weighted, **tight**	$m^{1+o(1)}$	$3 - \delta$ [4]
	$\tilde{O}(m\sqrt{n})$	almost 2	unweighted, nearly **tight**	$m^{\frac{k}{k-1}-o(1)}$	$3 - 2/k - \epsilon$ [4]
	$\tilde{O}(m^{3/2})$	2	weighted, **tight**	"	"

Table 4 Subset results. See caption of Table 1.

	Upper Bounds			Lower Bounds	
Problem	Runtime	Approx.	Comments	Runtime	Approx.
Diameter	$\tilde{O}(m)$	2	weighted, directed, **tight**	$m^{2-o(1)}$	$2 - \delta$
Radius	$\tilde{O}(m)$	2	weighted, undirected, **tight**	$m^{2-o(1)}$	$2 - \delta$
	$\tilde{O}(m/\delta)$	$2 + \delta$	weighted, directed, **tight** up to an additive δ	"	"
Eccentricities	$\tilde{O}(m/\delta)$	$2 + \delta$	weighted, directed, **tight** up to an additive δ	$m^{2-o(1)}$	$2 - \delta$

APSP algorithms [6, 23, 22] as they run in $\tilde{O}(mn)$ time. The $m\sqrt{n}$ time of the algorithms that allow small additive error beat the APSP algorithms for every graph sparsity.

Bichromatic Diameter and Radius

Our first contribution is an algorithm with the same running time as the 2-approximation ST-Diameter algorithm of [4], achieving a better, 5/3 approximation for Bichromatic Diameter. In other words, when S and T partition the vertex set of the graph, ST-Diameter can be approximated much better! Moreover, we show that under SETH, neither the runtime nor the approximation factor of our algorithm can be improved. The result is summarized in Theorem 1 below, and proven in Theorems 11 and 12.

▶ **Theorem 1.** *There is a randomized $\tilde{O}(m^{3/2})$ time algorithm, that given an undirected graph $G = (V, E)$ with nonnegative integer edge weights and $S \subseteq V, T = V \setminus S$, can output an estimate D' such that $3D_{ST}/5 \leq D' \leq D_{ST}$ with high probability, where D_{ST} is the ST-Diameter of G.*

Moreover, if there is an $O(m^{3/2-\varepsilon})$ time 5/3-approximation algorithm for some $\varepsilon > 0$, or if there is an $O(m^{2-\varepsilon})$ time $(5/3 - \varepsilon)$-approximation algorithm for the problem, then SETH is false.

We also obtain an $\tilde{O}(m\sqrt{n})$ time algorithm that achieves an "almost" 5/3-approximation: the guarantee for unweighted graphs is $3D_{ST}/5 - 6/5 \leq D' \leq D_{ST}$. We also obtain a near-linear time algorithm for weighted graphs that returns an estimate D' with $D_{ST}/2 - W/2 \leq D' \leq D_{ST}$ where W is the minimum weight of a $S \times T$ edge. Using our general theorem 12, we get that this result is also essentially tight, as a $(2 - \varepsilon)$-approximation for $\varepsilon > 0$ running in near-linear time would refute SETH.

To obtain our improvements for Bichromatic Diameter over the known ST-Diameter algorithms, we crucially exploit the basic fact that as S, T partition V any path that starts

from a vertex $s \in S$ and ends in a vertex $t \in T$ must cross a (u, v) edge such that $u \in S, v \in T$. While this fact is clear, it not at all obvious how one might try to exploit it.

We explain our technique in more detail for the bichromatic diameter problem, and similar ideas are used for our algorithms for the other problems. Let $s^* \in S$ and $t^* \in T$ be end-points of an ST-Diameter path. Similarly to prior Diameter algorithms, our goal is to run Dijkstra's algorithm from some $s \in S$ which is close to s^*, and hence far from t^*, or from some $t \in T$ which is close to t^* and hence far from s^* (by the triangle inequality). Our 5/3-approximation algorithms are a delicate combination of two themes: (1) randomly sample nodes in S and nodes in T – similarly to prior works, the sampling works well if there are many nodes of S that are close to s^*, or if there are many nodes of T that are close to t^*. If (1) is not good enough, in theme (2) we show that we can find a node $w \in S$ close to t^* for which we can "catch" an $S \times T$ edge (s, t) on the shortest $w \to t^*$ path, such that t is close to t^*. Theme (2) is our new contribution. Because of theme (2), our algorithms are more complicated than the ST-Diameter algorithms, but run in asymptotically the same time, and achieve a better approximation guarantee. In order to better separate the ideas in our algorithms, we explain them in several steps, where Theme (1) can be seen in the first steps and Theme (2) appears towards the last steps.

Following a similar approach to our Bichromatic Diameter algorithms, we develop similar algorithms for Bichromatic Radius. First, we give a simple near-linear time almost 2-approximation algorithm, and then we adapt the 5/3-approximation for Bichromatic Diameter to also give a 5/3-approximation for Bichromatic Radius. Moreover, we show that any better approximation factor requires essentially quadratic time, under the Hitting Set (HS) Hypothesis of [1] (see also [14]).

▶ **Theorem 2.** *There is a randomized $\tilde{O}(m^{3/2})$ time algorithm, that given an undirected graph $G = (V, E)$ with nonnegative integer edge weights and $S \subseteq V, T = V \setminus S$, can output an estimate R' such that $R_{ST} \leq R' \leq 5R_{ST}/3$ with high probability, where R_{ST} is the ST-Radius of G. Moreover, if there is a $5/3 - \varepsilon$ approximation algorithm running in $O(m^{2-\delta})$ time for any $\varepsilon, \delta > 0$, then the HS Hypothesis is false.*

Similarly to the Bichromatic Diameter algorithm, if one is satisfied with a slight additive error, one can improve the runtime to $\tilde{O}(m\sqrt{n})$.

ST-Eccentricities and ST-Radius

Prior work only considered ST-Diameter but did not consider the more general ST- Eccentricities problem in which one wants to approximate for every $s \in S$, $\varepsilon_{ST}(s) := \max_{t \in T} d(s, t)$.

Here we show that one can achieve exactly the same approximation factors for ST-Eccentricities as for ST-Diameter. Since any conditional lower bound for ST-Diameter also applies for the ST-Eccentricities problem, the algorithms we obtain are conditionally optimal, similarly to the ST-Diameter algorithms in [4]. Interestingly, we show that the same conditional lower bounds apply for Bichromatic Eccentricities (see the full version [12]), and therefore our ST-Eccentricities algorithms are optimal even for the Bichromatic case.

▶ **Theorem 3.** *There is a randomized $\tilde{O}(m^{3/2})$ time algorithm, that given an undirected graph $G = (V, E)$ with nonnegative integer edge weights and $S, T \subseteq V$, can output for every $s \in S$, an estimate $\varepsilon'(s)$ such that $\varepsilon_{ST}(s)/2 \leq \varepsilon'(s) \leq \varepsilon_{ST}(s)$ with high probability. Moreover, if there is a $2 - \varepsilon$ approximation algorithm running in $O(m^{2-\delta})$ time for any $\varepsilon, \delta > 0$ or a 2-approximation algorithm running in $O(m^{3/2-\varepsilon})$ time for $\varepsilon > 0$, even for the Bichromatic case when $T = V \setminus S$, then SETH is false.*

Again, as before, one can improve the runtime to $\tilde{O}(m\sqrt{n})$ with a slight additive error, and there is a simple near-linear time 3-approximation algorithm which is tight under SETH, similar to the one in [4] for ST-Diameter. A simple argument shows that these algorithms imply algorithms with the same running time and approximation factor for ST-Radius.

Bichromatic and ST Problems in Directed Graphs

Using simple reductions we first show that there can be no $O(m^{2-\varepsilon})$ time (for $\varepsilon > 0$) algorithms that achieve any finite approximation for ST-Diameter or ST-Eccentricities (under SETH), or ST-Radius (under HS). Interestingly, the same holds for Bichromatic Eccentricities (under SETH, see the full version [12]) and Bichromatic Radius (under HS, see [12]), but not Bichromatic Diameter! Surprisingly, unlike those two problems, Bichromatic Diameter does admit a finite, in fact 2-approximation algorithm running in subquadratic time, and this algorithm is conditionally optimal:

▶ **Theorem 4.** *There is a randomized $\tilde{O}(m^{3/2})$ time algorithm, that given a directed graph $G = (V, E)$ with nonnegative integer edge weights and $S \subseteq V, T = V \setminus S$, can output an estimate D' such that $D_{ST}/2 \leq D' \leq D_{ST}$ with high probability, where D_{ST} is the ST-Diameter of G.*

Moreover, if there is an $O(m^{2-\varepsilon})$ time $2-\delta$-approximation algorithm for the problem for some $\varepsilon, \delta > 0$, then SETH is false.

The previously known techniques for approximating Diameter in directed graphs fail here. The main issue is that the prior techniques were general enough that they also gave algorithms for Eccentricities and Radius as a byproduct. In the Bichromatic case, however, there is a genuine difference between Diameter and Radius, as we noted above, and new techniques are needed. Here again it turns out that combining theme (2) with a delicate argument is sufficient to get conditionally tight algorithms under SETH.

Subset Versions

Recall that Subset Diameter, Radius, and Eccentricities are the versions of the corresponding ST problems with the constraint that $S = T$. Interestingly, Subset Diameter, Radius, and Eccentricities all exhibit the same sharp threshold behavior. For all three problems, there are near-linear time algorithms that achieve a 2 (or almost 2) approximation, as well as conditional lower bounds that show that there is no $2-\delta$ approximation in $m^{2-o(1)}$ time.

Parameterized Algorithms

We consider the Bichromatic Diameter, Radius, and Eccentricities problems parameterized by the size of the *boundary* between the S and T sets. If S' is the set of vertices in S that have a neighbor in T, and T' is the set of vertices in T that have a neighbor in S, then the boundary B is whichever of S' or T' is smaller in size. Our lower bound constructions already have small boundary so they rule out algorithms even for graphs with small boundary. However, interestingly we obtain near-linear time algorithms for graphs with small boundary that achieve *better* multiplicative approximation factors than the optimal non-parameterized algorithms. This is not a contradiction because our parameterized algorithms have a constant additive error, while the apparently contradictory lower bounds do not tolerate additive error.

2 Preliminaries

Given a graph $G = (V, E)$ (directed or undirected, weighted or unweighted), let $d(u, v)$ denote the distance from $u \in V$ to $v \in V$. For a subset $X \subseteq V$ and $v \in V$, define $d(v, X) := \min_{x \in X} d(v, x)$. Similarly $d(X, v) := \min_{x \in X} d(x, v)$.

Unless otherwise stated, m denotes the number of edges and n the number of vertices of the underlying graph. Without loss of generality, we can assume that all undirected graphs are connected, and all directed graphs are weakly connected, so that $m \geq n - 1$.

The *Eccentricity* $\varepsilon(v)$ of a vertex $v \in V$ is $\max_{u \in V} d(v, u)$. The *Diameter* $D(G)$ of G is $\max_{v \in V} \varepsilon(v)$, and the *Radius* $R(G)$ of G is $\min_{v \in V} \varepsilon(v)$.

Given $S, T \subseteq V$, we define analogous parameters as follows. The ST-Eccentricity $\varepsilon_{ST}(v)$ of $v \in S$ is $\max_{u \in T} d(v, u)$. The ST-Diameter $D_{ST}(G)$ is $\max_{v \in S} \varepsilon_{ST}(v)$, and the ST-Radius $R_{ST}(G)$ is $\min_{v \in S} \varepsilon_{ST}(v)$.

The above parameters are called *Bichromatic* Eccentricities, Diameter, and Radius if S and T form a partition of V, i.e. $T = V \setminus S$.

The above parameters are called *Subset* Eccentricities, Diameter, and Radius if $S = T$ and are notated with subscript S instead of ST.

2.1 Preliminaries for algorithms

▶ **Lemma 5.** *Let $G = (V, E)$ be a (possibly directed and weighted graph) and let $W \subseteq V$. Let $g \geq \Omega(\ln n)$ be an integer. Let $S \subseteq W$ be a random subset of $c(|W|/g) \ln n$ vertices for some constant $c > 1$. For every $v \in V$, let $W(v)$ be the set of vertices $x \in W$ for which $d(v, x) < d(v, S)$. Then with probability at least $1 - 1/n^{c-1}$, for every $v \in V$, $|W(v)| \leq g$, and moreover, if one takes the closest g vertices of W to v, they will contain $W(v)$.*

Proof. For each $v \in V$, imagine sorting the nodes $x \in W$ according to $d(v, x)$. Define Q_v to be the first g nodes in this sorted order - those are the nodes of W closest to v (in the $v \to x$ direction).

We pick S randomly by selecting each vertex of W with probability $(c \ln n)/g$. The probability that a particular $q \in Q_v$ is not in S is $1 - (c \ln n)/g$, and the probability that no $q \in Q_v$ is in S is $(1 - (c \ln n)/g)^g \leq 1/n^c$. By a union bound, with probability at least $1 - 1/n^{c-1}$, for every $v \in V$, we have that $Q_v \cap S \neq \emptyset$.

Now, for each particular v, say that $w(v)$ is a node in $Q_v \cap S$. Since all nodes $x \in W$ with $d(v, x) < d(v, w(v))$ must be in Q_v, and since $d(v, w(v)) \geq d(v, S)$, we must have that $W(v) \subseteq Q_v$. Hence, with probability at least $1 - 1/n^{c-1}$, for every $v \in V$, $|W(v)| \leq g$ and $W(v) \subseteq Q_v$. ◀

▶ **Lemma 6.** *Let $G = (V, E)$ be a (possibly directed and weighted) graph. Let $M, W \subseteq V$ and let $S \subseteq W$ be a random subset of $c(n/g) \ln n$ vertices for some large enough constant c and some integer $g \geq 1$.*

Then, for any $D > 0$ and for any $w \in M$ with $d(w, S) > D$, if one takes the closest g vertices of W to w, they will contain all nodes of W at distance $< D$ from w, with high probability.

Proof. Let Q be the closest g vertices of W to w. By Lemma 5, with high probability Q contains all nodes of W at distance $< d(w, S)$ from w, and hence Q contains all nodes of W at distance $< D$ from w, with high probability. ◀

We sometimes sample edges instead of vertices, so analogous lemmas to Lemmas 5 and 6 hold when the sample is from a set of edges. Here is the analogue of Lemma 6. The other lemma is similar.

▶ **Lemma 7.** *Let $G = (V, E)$ be a (possibly directed and weighted graph) and let $M, W \subseteq V$. Let $E' \subseteq E$ be a random subset of $c(|E|/g) \ln n$ edges for some large enough constant c and some integer $g \geq 1$. Let Q be the endpoints of edges in E' that are in W.*

Then, for any $D > 0$, and for any w with $d(w, S) > D$, if one takes the closest g edges of E' to w wrt the distance from their W endpoints, they will contain all edges of E' whose W endpoints are at distance $< D$ from w, with high probability.

2.2 Preliminaries for lower bounds

The Strong Exponential Time Hypothesis (SETH) asserts that on a Word-RAM with $O(\log n)$ bit words, there is no $(2-\varepsilon)^n$ time (possibly randomized) algorithm for some constant $\varepsilon > 0$ that can determine whether a given CNF-Formula with n variables and $O(n)$ clauses is satisfiable. (This version of SETH is equivalent to the original formulation by Impagliazzo, Paturi and Zane [15].) By a result of Williams [27], the following Orthogonal Vectors (OV) Problem requires $n^{2-o(1)}\text{poly}(d)$ time (on a word-RAM with $O(\log n)$ bit words), unless SETH fails: given two sets $U, V \subseteq \{0,1\}^d$ with $|U| = |V| = n$ and $d = \omega(\log n)$, determine whether there are $u \in U, v \in V$ with $u \cdot v = 0$.

Given an arbitrary instance of OV with $d = \tilde{O}(1)$ (while respecting $d = \omega(\log n)$, e.g. $d = \Theta(\log^2 n)$), consider the following graph representation, which we call the *OV-graph*: the vertex set consists of a node for every $u \in U$, for every $v \in V$ and for every coordinate $c \in [d] = C$, and there is an edge $(x \in U \cup V, c \in C)$ if and only if $x[c] = 1$. OV is then equivalent to the question of whether there exist $u \in U, v \in V$ such that $d(u, v) > 2$. In fact, it is equivalent to distinguishing whether for every $u \in U, v \in V$, $d(u, v) = 2$ (no OV-solution), or there is some $u \in U, v \in V$ such that $d(u, v) \geq 4$ (OV-solution). In other words, if we set $S = U, T = V$, the ST-Diameter of the OV-graph is 2 if and only if there is no OV-solution and at least 4 otherwise. Because the OV graph has $m = \tilde{O}(n)$, under SETH, any $(2-\delta)$-approximation algorithm for ST-Diameter requires $m^{2-o(1)}$.

A related problem to OV is the Hitting Set (HS) problem [1, 14, 26]: given two sets $U, V \subseteq \{0,1\}^d$ with $|U| = |V| = n$ and $d = \omega(\log n)$, determine whether there is $u \in U$ such that for all $v \in V$, $u \cdot v \neq 0$. A common hypothesis is that (on the word-RAM) HS requires $n^{2-o(1)}$ time.

If we form the OV-graph on the HS instance input, then the HS problem becomes equivalent to determining whether there is some $u \in U$ such that for all $v \in V$, $d(u, v) \leq 2$. In other words, if we set $S = U, T = V$, the ST-Radius of the OV-graph is 2 if and only if there is a HS-solution and at least 4 otherwise. Thus, under the HS hypothesis, any $(2-\delta)$-approximation algorithm for ST-Radius requires $m^{2-o(1)}$.

Additionally for our constructions we assume that if there is a HS solution u' then for all $c \in C$, $d(u', c) \leq 3$. This is because for every coordinate index i there must be $v \in V$ with $v[i] = 1$ as otherwise we can just delete the i^{th} bit from all vectors.

Let $k \geq 2$ be an integer. Then, a generalization of the OV problem is k-OV: given k sets $U_1, \ldots, U_k \subseteq \{0,1\}^d$, are there $u_1 \in U_1, \ldots, u_k \in U_k$ so that $\sum_{c=1}^{d} \prod_{i=1}^{k} u_i[c] = 0$? It is known that, under SETH, when $d = \omega(\log n)$, there is no $n^{k-o(1)}$ time algorithm for k-OV (in the word RAM model) [27].

Similar to the OV-graph, Backurs et al. [4] define a graph for k-OV which we will refer to as the k-OV-graph. We do not explicitly define the k-OV-graph here; instead we list its properties in the following theorem.

▶ **Theorem 8** ([4]). *Let $k \geq 2$. Given a k-OV instance consisting of sets $W_0, W_1, \ldots, W_{k-1} \subseteq \{0,1\}^d$, each of size n, we can in $O(kn^{k-1}d^{k-1})$ time construct an unweighted, undirected graph with $O(n^{k-1} + kn^{k-2}d^{k-1})$ vertices and $O(kn^{k-1}d^{k-1})$ edges that satisfies the following properties.*

1. *The graph consists of $k+1$ layers of vertices $L_0, L_1, L_2, \ldots, L_k$. The number of nodes in the sets is $|L_0| = |L_k| = n^{k-1}$ and $|L_1|, |L_2|, \ldots, |L_{k-1}| \leq n^{k-2}d^{k-1}$.*
2. *L_0 consists of all tuples $(a_0, a_1, \ldots, a_{k-2})$ where for each i, $a_i \in W_i$. Similarly, L_k consists of all tuples $(b_1, b_2, \ldots, b_{k-1})$ where for each i, $b_i \in W_i$.*
3. *If the k-OV instance has no solution, then $d(u,v) = k$ for all $u \in L_0$ and $v \in L_k$.*
4. *If the k-OV instance has a solution $a_0, a_1, \ldots, a_{k-1}$ where for each i, $a_i \in W_i$ then if $\alpha = (a_0, \ldots a_{k-2}) \in L_0$ and $\beta = (a_1, \ldots, a_{k-1}) \in L_k$, then $d(\alpha, \beta) \geq 3k-2$.*
5. *For all i from 1 to $k-1$, for all $v \in L_i$ there exists a vertex in L_{i-1} adjacent to v and a vertex in L_{i+1} adjacent to v.*

2.3 Organization

In Section 3 we present our algorithms and conditional lower bound for undirected bichromatic diameter. We defer the rest of our algorithms and conditional lower bounds to the full version [12].

3 Algorithms and Lower Bound for Undirected Bichromatic Diameter

We begin with a simple near-linear time algorithm.

▶ **Proposition 9.** *There is an $O(m + n \log n)$ time algorithm, that given an undirected graph $G = (V, E)$ and $S \subseteq V, T = V \setminus S$, can output an estimate D' such that $D_{ST}(G)/2 - W/2 \leq D' \leq D_{ST}$, where W is the minimum weight of an edge in $S \times T$.*

Proof. Let (s,t) be a minimum weight edge of G with $s \in S$ and $t \in T$. Run Dijkstra's algorithm from s and from t. Let $D' = \max\{\max_{t' \in T} d(s,t'), \max_{s' \in S} d(s',t)\}$. Let $s^* \in S, t^* \in T$ be endpoints of an ST-Diameter path, i.e. $d(s^*, t^*) = D_{ST}$. Then, suppose that $\max_{t' \in T} d(s,t') < D_{ST}/2 - W/2$. In particular, $d(s,t^*) < D_{ST}/2 - W/2$, and hence $d(s,s^*) > D_{ST}/2 + W/2$ by the triangle inequality. Also by the triangle inequality,

$$D_{ST}/2 + W/2 < d(s,t) + d(t,s^*) \leq w(s,t) + \max_{s' \in S} d(s',t).$$

Hence, $D' > D_{ST}/2 - W/2$, where W is the minimum weight of an edge in $S \times T$. ◀

Now we turn to our 5/3-approximation algorithms. Our first theorem is for unweighted graphs. Later on, we modify the algorithm in this theorem to obtain an algorithm for weighted graphs as well, and at the same time remove the small additive error that appears in the theorem below.

▶ **Theorem 10.** *There is an $\tilde{O}(m\sqrt{n})$ time algorithm, that given an unweighted undirected graph $G = (V, E)$ and $S \subseteq V, T = V \setminus S$, can output an estimate D' such that $3D_{ST}(G)/5 \leq D' \leq D_{ST}(G)$ if $D_{ST}(G)$ is divisible by 5, and otherwise $3D_{ST}(G)/5 - 6/5 \leq D' \leq D_{ST}(G)$.*

Proof. Let $D = D_{ST}(G)$ and let us assume that D is divisible by 5. If D is not divisible by 5, the estimate we return will have a small additive error. For clarity of presentation, we omit the analysis of the case where D is not divisible by 5. However, we include such analyses in our proofs for Bichromatic Radius and ST-Eccentricities (see ARXIV) and the analysis for Diameter is analogous.

Suppose the (bichromatic) ST-Diameter endpoints are $s^* \in S$ and $t^* \in T$ and that the ST-Diameter is D. The algorithm does not know D, but we will use it in the analysis.

(Algorithm Step 1): The algorithm first samples $Z \subseteq S$ of size $c\sqrt{n}\ln n$ uniformly at random. For every $z \in Z$, run BFS, and let $D_1 = \max_{z\in Z, t\in T} d(z,t)$.

(Analysis Step 1): If for some $s' \in Z$ we have that $d(s^*, s') \leq 2D/5$, then $D_1 \geq d(s', t^*) \geq D - d(s^*, s') \geq 3D/5$.

(Algorithm Step 2): Now, sample a set X from T of size $C\sqrt{n}\ln n$ uniformly at random for large enough constant C. For every $t \in X$, run BFS and find the closest node $s(t)$ of S to t. Run BFS from every $s(t)$. Let $D_2 = \max_{t\in X, t'\in T} d(s(t), t')$.

(Analysis Step 2): If s^* is at distance $\leq D/5$ from some node t of X, then $d(s^*, s(t)) \leq 2D/5$ (since $s(t)$ is closer to t than s^*), and so $D_2 \geq d(s(t), t^*) \geq 3D/5$.

If neither D_1, nor D_2 are good approximations, it must be that $d(s^*, X) > D/5$ and $d(s^*, Z) > 2D/5$. Consider the nodes M of S that are at distance $> 2D/5$ from Z, then the node $w \in M$ that is furthest from X among all nodes of M. If neither D_1, nor D_2 was a good approximation, $s^* \in M$ and since $d(s^*, X) > D/5$, we must have that $d(w, X) > D/5$ (and also $d(w, Z) > 2D/5$). In the next step we will look for such a w.

(Algorithm Step 3): For each $s \in S$ define D_s to be the biggest integer which satisfies $d(s, X) > D_s/5$ and $d(s, Z) > 2D_s/5$. Let $w = \arg\max D_s$ and $D' = \max D_s$.

(Analysis Step 3): By Lemma 6 we have that whp, the number of nodes of T at distance $\leq D'/5$ from w and the number of nodes of S at distance $\leq 2D'/5$ from w are both $\leq \sqrt{n}$. Also if neither D_1, nor D_2 are good approximations, it must be that $d(s^*, X) > D/5$ and $d(s^*, Z) > 2D/5$ and hence $D' \geq D$.

(Algorithm Step 4): Run BFS from w. Take all nodes of S at distance $\leq 2D'/5$ from w, call these S_w, and run BFS from them. Whp, $|S_w| \leq \sqrt{n}$, so that this BFS run takes $O(m\sqrt{n})$ time. Let $D_3 := \max_{s\in S_w, t\in T} d(s,t)$.

For every $s \in S_w$, let $t(s)$ be the closest node of T to s (breaking ties arbitrarily). Run BFS from each $t(s)$. Let $D_4 := \max_{s\in S_w, s'\in S} d(s', t(s))$.

(Analysis Step 4): If $D_3 \geq 3D/5$ or $D_4 \geq 3D/5$, we are done, so let us assume that $D_3, D_4 < 3D/5$. Since $D_3 < 3D/5$, and since $D_3 \geq d(w, t^*)$, it must be that $d(w, t^*) < 3D/5$. Let P_{wt^*} be the shortest w to t^* path. Consider the node b on P_{wt^*} for which $d(w, b) = 2D/5$. If $b \in S$, then since $D' \geq D$, $b \in S_w$ and hence we ran BFS from $t(b)$. But since $d(b, t^*) = d(w, t^*) - 2D/5 < D/5$, and $d(b, t(b)) \leq d(b, t^*)$ we have that $d(t(b), t^*) \leq 2D/5$ and hence $D_4 \geq d(s^*, t(b)) \geq D - d(t(b), t^*) \geq 3D/5$. Thus, if $D_4 < 3D/5$, it must be that $b \in T$.

(Algorithm Step 5): Take all nodes of T at distance $\leq D'/5$ from w, call these T_w and run BFS from them. Since $d(w, X) > D'/5$, whp $|T_w| \leq \sqrt{n}$, so this step runs in $O(m\sqrt{n})$ time. Let $D_5 = \max_{t\in T_w, s\in S} d(t, s)$.

(Analysis Step 5): If $D_5 \geq 3D/5$, we would be done, so assume that $D_5 < 3D/5$. Let a be the node on the shortest w to t^* path P_{wt^*} with $d(w, a) = D/5$. Suppose that $a \in T$. Since $D' \geq D$, $a \in T_w$ and we ran BFS from it. However, also $d(a, t^*) = d(w, t^*) - d(w, a) < 3D/5 - D/5 = 2D/5$, and hence $D_5 \geq d(a, s^*) \geq d(t^*, s^*) - d(t^*, a) \geq D - 2D/5 = 3D/5$. Since $D_5 < 3D/5$, it must be that $a \in S$.

Now, since $a \in S$ and $b \in T$, somewhere on the a to b shortest path P_{ab}, there must be an edge (s', t') with $s' \in S, t' \in T$. Since s' is before b, $d(w, s') \leq 2D/5 \leq 2D'/5$, and hence $s' \in S_w$. Thus we ran BFS from $t(s')$. Since s' has an edge to $t' \in T$, $d(s', t(s')) \leq d(s', t') = 1$.

Also, since $d(w,s') \geq d(w,a) = D/5$ and $d(w,t^*) \leq 3D/5 - 1$, $d(s',t^*) \leq 2D/5 - 1$. Thus,

$$\begin{aligned} D_4 \geq d(t(s'),s^*) &\geq d(s^*,t^*) - d(t(s'),t^*) \\ &\geq D - d(t(s'),s') - d(s',t^*) \\ &\geq D - 1 - 2D/5 + 1 = 3D/5. \end{aligned}$$

Hence if we set $D'' = \max\{D_1, D_2, D_3, D_4, D_5\}$, we get that $3D/5 \leq D'' \leq D$. ◀

We now modify the algorithm for unweighted graphs, both making the algorithm work for weighted graphs and removing the additive error, at the expense of increasing the runtime to $\tilde{O}(m^{3/2})$.

▶ **Theorem 11.** *There is an $\tilde{O}(m^{3/2})$ time algorithm, that given an undirected graph $G = (V,E)$ with nonnegative integer edge weights and $S \subseteq V, T = V \setminus S$, can output an estimate D' such that $3D_{ST}(G)/5 \leq D' \leq D_{ST}$.*

Proof. Suppose as before the (bichromatic) ST-Diameter endpoints are $s^* \in S$ and $t^* \in T$ and that the ST-Diameter is D.

(Algorithm Modified Step 1): The algorithm here samples $E' \subseteq E$ of size $c\sqrt{m}\ln n$ uniformly at random, for large enough c. Let Z be the endpoints of edges in E' that are in S. For every $z \in Z$, run Dijkstra's algorithm, and let $D_1 = \max_{z \in Z, t \in T} d(z,t)$.

(Analysis Step 1): If for some $s' \in Z$ we have that $d(s^*,s') \leq 2D/5$, then $D_1 \geq d(s',t^*) \geq D - d(s^*,s') \geq 3D/5$. Let us then assume that $d(s^*,Z) > 2D/5$.

(Algorithm Modified Step 2): Let X be the endpoints of edges in E' that are in T. For every $t \in X$, run Dijkstra's algorithm and find the closest node $s(t)$ of S to t. Run Dijkstra's algorithm from every $s(t)$. Let $D_2 = \max_{t \in X, t' \in T} d(s(t),t')$.

(Analysis Step 2): If s^* is at distance $\leq D/5$ from some node t of X, then $d(s^*,s(t)) \leq 2D/5$ (since $s(t)$ is closer to t than s^*), and so $D_2 \geq d(s(t),t^*) \geq 3D/5$. Let us then assume that $d(s^*,X) > D/5$.

As before, if we consider the nodes M of S that are at distance $> 2D/5$ from Z, then the node $w \in M$ that is furthest from X among all nodes of M, would have both $d(w,Z) > 2D/5$ and $d(w,X) > D/5$, as s^* is in M and satisfies $d(s^*,X) > D/5$. We will find a node w with these properties in the next step.

(Algorithm Unmodified Step 3): Perform exactly the same Step 3 as before, finding the largest integer D' such that there is some node $w \in S$ with $d(w,Z) > 2D'/5$ and $d(w,X) > D'/5$.

(Analysis Step 3): Let $w \in S$ be the node we found such that $d(w,X) > D'/5, d(w,Z) > 2D'/5$. By Lemma 7 we have that whp, the number of edges (s,g) where $s \in S, g \in V$ and $d(w,s) \leq 2D'/5$ and the number of edges (t,g') where $t \in T, g' \in V$ and $d(w,t) \leq D'/5$ is at most $\sqrt{m}$. Also, if $D_1, D_2 < 3D/5$, then $D' \geq D$, so that we also have that the number of edges (s,b) where $s \in S$ and $d(w,s) \leq 2D/5$ and the number of edges (t,b') where $t \in T$ and $d(w,t) \leq D/5$ is at most $\sqrt{m}$, whp.

(Algorithm Modified Step 4): Run Dijkstra's algorithm from w. Take all edges incident to nodes of S at dist $\leq 2D'/5$ from w. Call these edges E_S and their endpoints S_w. Run Dijkstra's algorithm from both of their end points. Whp, $|E_S| \leq \sqrt{m}$ and so $|S_w| \leq 2\sqrt{m}$, so that this Dijkstra run takes $\tilde{O}(m^{3/2})$ time. Let $D_3 := \max_{t \in S_w \cap T, s \in S} d(s,t)$.

For every $s \in S_w \cap S$, determine a closest node $t(s) \in T$ to s, and run Dijkstra's algorithm from $t(s)$ as well. This search also takes $O(m^{3/2})$ time. Let $D_4 := \max_{s \in S_w \cap S, s' \in S} d(s', t(s))$.

(Analysis Step 4): If $d(w,t^*) \geq 3D/5$, or $D_3 \geq 3D/5$ or $D_4 \geq 3D/5$, we are done, so let us assume that $d(w,t^*), D_3, D_4 < 3D/5$.
Now consider the node b on the shortest w to t^* path P_{wt^*} for which $d(w,b) \leq 2D/5$, but such that the node b' after it on P_{wt^*} has $d(w,b') > 2D/5$.
Suppose that $b \in S$. Then since $D' \geq D$, we have $d(w,b) \leq 2D'/5$ and hence $(b,b') \in E_S$. Let us consider $d(b',t^*) = d(w,t^*) - d(b',w)$. Since $d(w,t^*) < 3D/5$ and $d(b',w) > 2D/5$, $d(b',t^*) < D/5$. If $b' \in T$, then since we ran Dijkstra's algorithm from b', we got $D_3 \geq D - D/5 = 4D/5$. If $b' \in S$, then we ran Dijkstra's algorithm from $t(b')$ and $d(t(b'),t^*) \leq d(t(b'),b') + d(b',t^*) \leq 2d(b',t^*) < 2D/5$, and hence $D_4 \geq d(t(b),s^*) \geq D - 2D/5 = 3D/5$. Thus if neither $d(w,t^*)$, D_3, nor D_4 are good approximations, then $b \in T$.

(Algorithm Modified Step 5): Take all edges incident to nodes of T at dist $\leq D'/5$ from w. Call these edges E_T and their endpoints that are in T, T_w. Run Dijkstra's algorithm from all nodes in T_w.
Since $d(w,X) > D'/5$, whp $|T_w| \leq 2\sqrt{m}$, so this step runs in $O(m^{3/2})$ time. Let $D_5 = \max_{t \in T_w, s \in S} d(t,s)$.

(Analysis Step 5): If $D_5 \geq 3D/5$, we would be done, so assume that $D_5 < 3D/5$. Let a be the node on P_{wt^*} with $d(w,a) \leq D/5$ but so that the node a' after a on P_{wt^*} has $d(w,a') > D/5$. Suppose that $a' \in T$. Since $D' \geq D$, $(a,a') \in E_T$, $a' \in T_w$ and we ran Dijkstra's algorithm from a'. However, also $d(a',t^*) = d(w,t^*) - d(w,a') < 3D/5 - D/5 = 2D/5$, and hence $D_5 \geq d(a,s^*) \geq d(t^*,s^*) - d(t^*,a') \geq D - 2D/5 = 3D/5$. Since $D_5 < 3D/5$, it must be that $a' \in S$.

Now, since $a' \in S$ and $b \in T$, somewhere on the a' to b shortest path P_{ab}, there must be an edge (s',t') with $s' \in S, t' \in T$. However, since s' is before b, we have that $d(w,s') \leq d(w,b) \leq 2D/5 \leq 2D'/5$. Thus, $(s',t') \in E_S$ and we ran Dijkstra's algorithm from t'. However, $d(t',t^*) = d(w,t^*) - d(w,t') \leq d(w,t^*) - d(w,a') < 3D/5 - D/5 = 2D/5$, and hence $D_3 \geq d(t',s^*) \geq d(s^*,t^*) - d(t',t^*) > 3D/5$.

Hence if we set $D'' = \max\{d(w,t^*), D_1, D_2, D_3, D_4, D_5\}$, we get that $3D/5 \leq D'' \leq D$. ◀

Conditional Lower Bound

The following theorem implies that our algorithms for undirected Bichromatic Diameter from Theorem 11 and Proposition 9 are tight under SETH.

▶ **Theorem 12.** *Under SETH, for every $k \geq 2$, every algorithm that can distinguish between Bichromatic Diameter $2k-1$ and $4k-3$ in undirected unweighted graphs requires $m^{1+1/(k-1)-o(1)}$ time.*

In particular setting $k = 2$ and 3 in Theorem 12 implies that our $m^{3/2}$ time 5/3-approximation algorithm from Theorem 11 is tight in approximation factor and runtime, respectively. Furthermore, setting k to be arbitrarily large implies that our $\tilde{O}(m)$ time almost 2-approximation algorithm from Proposition 9 is tight under SETH.

Theorem 12 follows from the following lemma.

▶ **Lemma 13.** *Let $k \geq 2$ be any integer. Given a k-OV instance, we can in $O(kn^{k-1}d^{k-1})$ time construct an unweighted, undirected graph with $O(kn^{k-1} + kn^{k-2}d^{k-1})$ vertices and $O(kn^{k-1}d^{k-1})$ edges that satisfies the following two properties.*

1. *If the k-OV instance has no solution, then for* all *pairs of vertices $u \in S$ and $v \in T$ we have $d(u,v) \leq 2k-1$.*

2. *If the k-OV instance has a solution, then there exists a pair of vertices $u \in S$ and $v \in T$ such that $d(u, v) \geq 4k - 3$.*

Proof.

Construction of the graph. We begin with the k-OV-graph from Theorem 8. Additionally, we add $k - 1$ new layers of vertices $L_{k+1}, \ldots, L_{2k-1}$, where each new layer contains n^{k-1} vertices and is connected to the previous layer by a matching. That is, each new layer contains one vertex for every tuple $(a_1, \ldots, a_{k-1})$ where $a_i \in W_i$ for all i, and each $(a_1, \ldots, a_{k-1}) \in L_j$ is connected to its counterpart $(a_1, \ldots, a_{k-1}) \in L_{j-1}$ by an edge, for all j.

We let $S = L_0$ and we let T contain the rest of the vertices in the graph.

Correctness of the construction.

Case 1: The k-OV instance has no solution. By property 3 of Theorem 8 for all $u \in S$ and $v \in L_k$, $d(u, v) = k$. Then, since $L_k, \ldots, L_{2k-1}$ form a series of matchings, for all $u \in S$ and $v \in L_{k+1} \cup \cdots \cup L_{2k-1}$, $d(u, v) \leq 2k - 1$. Furthermore, property 5 of Theorem 8 implies that for all $u \in S$ and $v \in L_1 \cup \cdots \cup L_{k-1}$, $d(u, v) \leq 2k - 1$. Thus, we have shown that for all $u \in S$ and $v \in T$ we have $d(u, v) \leq 2k - 1$.

Case 2: The k-OV instance has a solution. Let $(a_0, a_1, \ldots, a_{k-1})$ be a solution to the k-OV instance where $a_i \in W_i$ for all i. We claim that $d((a_0, \ldots, a_{k-2}) \in S, (a_1, \ldots, a_{k-1}) \in L_{2k-1})) \geq 4k - 3$. Since $L_k, \ldots, L_{2k-1}$ form a series of matchings, every path from $(a_0, \ldots, a_{k-2}) \in S$ to $(a_1, \ldots, a_{k-1}) \in L_{2k-1}$ contains the vertex $(a_1, \ldots, a_{k-1}) \in L_k$. By property 4 of Theorem 8, $d((a_0, \ldots, a_{k-2}) \in S, (a_1, \ldots, a_{k-1}) \in L_k) \geq 3k - 2$. Thus, $d((a_0, \ldots, a_{k-2}) \in S, (a_1, \ldots, a_{k-1}) \in L_{2k-1})) \geq 4k - 3$. ◀

References

1. Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and Fixed Parameter Subquadratic Algorithms for Radius and Diameter in Sparse Graphs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 377–391, 2016.
2. Pankaj K. Agarwal, Herbert Edelsbrunner, Otfried Schwarzkopf, and Emo Welzl. Euclidean Minimum Spanning Trees and Bichromatic Closest Pairs. *Discrete Comput. Geom.*, 6(1):407–422, December 1991.
3. D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast Estimation of Diameter and Shortest Paths (Without Matrix Multiplication). *SIAM Journal on Computing*, 28(4):1167–1181, 1999.
4. Arturs Backurs, Liam Roditty, Gilad Segal, Virginia Vassilevska Williams, and Nicole Wein. Towards Tight Approximation Bounds for Graph Diameter and Eccentricities. In *Proceedings of STOC'18*, page to appear, 2018.
5. Massimo Cairo, Roberto Grossi, and Romeo Rizzi. New Bounds for Approximating Extremal Distances in Undirected Graphs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 363–376, 2016.
6. T. M. Chan. More Algorithms for All-Pairs Shortest Paths in Weighted Graphs. In *Proc. STOC*, pages 590–598, 2007.
7. Bernard Chazelle, Herbert Edelsbrunner, Leonidas J Guibas, and Micha Sharir. Algorithms for bichromatic line-segment problems and polyhedral terrains. *Algorithmica*, 11(2):116–132, 1994.
8. Shiri Chechik, Daniel H. Larkin, Liam Roditty, Grant Schoenebeck, Robert Endre Tarjan, and Virginia Vassilevska Williams. Better Approximation Algorithms for the Graph Diameter. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1041–1052, 2014.
9. Don Coppersmith and Michael Elkin. Sparse sourcewise and pairwise distance preservers. *SIAM Journal on Discrete Mathematics*, 20(2):463–501, 2006.

10 Pierluigi Crescenzi, Roberto Grossi, Leonardo Lanzi, and Andrea Marino. On Computing the Diameter of Real-World Directed (Weighted) Graphs. In Ralf Klasing, editor, *Experimental Algorithms: 11th International Symposium, SEA 2012, Bordeaux, France, June 7-9, 2012. Proceedings*, pages 99–110, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

11 Marek Cygan, Fabrizio Grandoni, and Telikepalli Kavitha. On pairwise spanners. *arXiv preprint*, 2013. `arXiv:1301.1999`.

12 Mina Dalirrooyfard, Virginia Vassilevska Williams, Nikhil Vyas, and Nicole Wein. Tight Approximation Algorithms for Bichromatic Graph Diameter and Related Problems, 2019. `arXiv:1904.11601`.

13 Adrian Dumitrescu and Sumanta Guha. Extreme Distances in Multicolored Point Sets. *J. Graph Algorithms and Applications*, 8(1):27–38, 2004.

14 Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and R. Ryan Williams. Completeness for First-Order Properties on Sparse Structures with Algorithmic Applications. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2162–2181, 2017.

15 R. Impagliazzo and R. Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

16 Piotr Indyk. A near linear time constant factor approximation for Euclidean bichromatic matching (cost). In *SODA*, volume 7, pages 39–42, 2007.

17 Naoki Katoh and Kazuo Iwano. Finding K Farthest Pairs and K Closest/Farthest Bichromatic Pairs for Points in the Plane. In *Proceedings of the Eighth Annual Symposium on Computational Geometry*, SCG '92, pages 320–329, 1992.

18 Telikepalli Kavitha. New pairwise spanners. *Theory of Computing Systems*, 61(4):1011–1036, 2017.

19 Philip N Klein. A subset spanner for planar graphs, with application to subset TSP. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 749–756. ACM, 2006.

20 Clémence Magnien, Matthieu Latapy, and Michel Habib. Fast Computation of Empirically Tight Bounds for the Diameter of Massive Graphs. *J. Exp. Algorithmics*, 13:10:1.10–10:1.9, February 2009.

21 David Peleg, Liam Roditty, and Elad Tal. Distributed Algorithms for Network Diameter and Girth. In *Automata, Languages, and Programming: 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, pages 660–672, 2012.

22 S. Pettie. A new approach to all-pairs shortest paths on real-weighted graphs. *Theor. Comput. Sci.*, 312(1):47–74, 2004.

23 Seth Pettie and Vijaya Ramachandran. A Shortest Path Algorithm for Real-Weighted Undirected Graphs. *SIAM J. Comput.*, 34(6):1398–1431, 2005.

24 Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, STOC '13, pages 515–524, New York, NY, USA, 2013. ACM. `doi:10.1145/2488608.2488673`.

25 Frank W. Takes and Walter A. Kosters. Determining the Diameter of Small World Networks. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 1191–1196, 2011.

26 Virginia Vassilevska Williams. Hardness of Easy Problems: Basing Hardness on Popular Conjectures such as the Strong Exponential Time Hypothesis (Invited Talk). In *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, pages 17–29, 2015.

27 R. Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2–3):357–365, 2005.

28 Ryan Williams. On the Difference Between Closest, Furthest, and Orthogonal Pairs: Nearly-linear vs Barely-subquadratic Complexity. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 1207–1215, 2018.

29 A. Yao. On Constructing Minimum Spanning Trees in k-Dimensional Spaces and Related Problems. *SIAM Journal on Computing*, 11(4):721–736, 1982.

Faster Algorithms for All Pairs Non-Decreasing Paths Problem

Ran Duan
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China
duanran@mail.tsinghua.edu.cn

Ce Jin
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China
jinc16@mails.tsinghua.edu.cn

Hongxun Wu
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China
wuhx18@mails.tsinghua.edu.cn

Abstract

In this paper, we present an improved algorithm for the All Pairs Non-decreasing Paths (APNP) problem on weighted simple digraphs, which has running time $\tilde{O}(n^{\frac{3+\omega}{2}}) = \tilde{O}(n^{2.686})$. Here n is the number of vertices, and $\omega < 2.373$ is the exponent of time complexity of fast matrix multiplication [Williams 2012, Le Gall 2014]. This matches the current best upper bound for $(\max, \min)$-matrix product [Duan, Pettie 2009] which is reducible to APNP. Thus, further improvement for APNP will imply a faster algorithm for $(\max, \min)$-matrix product. The previous best upper bound for APNP on weighted digraphs was $\tilde{O}(n^{\frac{1}{2}(3+\frac{3-\omega}{\omega+1}+\omega)}) = \tilde{O}(n^{2.78})$ [Duan, Gu, Zhang 2018]. We also show an $\tilde{O}(n^2)$ time algorithm for APNP in undirected simple graphs which also reaches optimal within logarithmic factors.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases graph optimization, matrix multiplication, non-decreasing paths

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.48

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1904.10701`.

1 Introduction

Given a directed or undirected graph $G = (V, E)$ with arbitrary real edge weights, a *non-decreasing path* is a path on which edge weights form a non-decreasing sequence [11]. We define the weight of a non-decreasing path to be the weight of its last edge, which we want to minimize. The motivation of this definition comes from train scheduling [13]. Suppose each train station is mapped to a vertex of a directed graph, and a train from station v_1 to station v_2 scheduled at time w is mapped to a directed edge (v_1, v_2) with weight w. If we neglect the time trains spent on their way, a trip from s to t is just a non-decreasing path from s to t in the constructed graph, and the earliest time arriving at t equals the minimum weight of such non-decreasing path. If we consider the train starts from v_1 at time w_1 and arrives at v_2 at time w_2, we can add a vertex u and two edges $(v_1, u), (u, v_2)$, then gives edge weights w_1, w_2 on them, respectively.

The *Single Source Non-decreasing Paths* (SSNP) problem, first studied by Minty [11] in 1958, is to find the minimum weight non-decreasing path from a given source s to all $t \in V$. The first linear time algorithm for SSNP problem in RAM model was given by Williams [13]. She also gave an $O(m \log \log n)$ time algorithm in the standard addition-comparison model. (m is the number of edges.)

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 48; pp. 48:1–48:13


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

The *All Pairs Non-decreasing Paths* (APNP) problem is the all-pairs version of the above problem, asking for the minimum weight non-decreasing path between *all* pairs of vertices in the graph. Williams [13] gave the first truly sub-cubic time algorithm of APNP. The original time complexity of the algorithm was $\tilde{O}(n^{(15+\omega)/6})$, obtained by using an $\tilde{O}(n^{2+\omega/3})$-time $(\min, \leq)$-matrix product[1] algorithm [12] as a subroutine. Here $\omega < 2.373$ is the exponent of time complexity of fast matrix multiplication [2, 14, 8]. Recently, Duan et al. [5] obtained a faster algorithm for APNP in $\tilde{O}(n^{\frac{1}{2}(3+\frac{3-\omega}{\omega+1}+\omega)}) = \tilde{O}(n^{2.78})$, by using a simple balancing technique introduced in [6]. We also adapt this technique in our algorithm.

Computing APNP is at least as hard as the *All Pair Bottleneck Paths* (APBP) problem [13], which asks for the maximum bottleneck path between every pair of vertices, where the bottleneck of a path is defined as the minimum weight (capacity) among all edges on this path [12, 15, 6, 9]. APBP is shown to have the same complexity as the $(\max, \min)$-matrix product ($C_{i,j} = \max_k\{\min(A_{i,k}, B_{k,j})\}$) [1, 12]. The current fastest algorithm for $(\max, \min)$-matrix product runs in $\tilde{O}(n^{(3+\omega)/2}) = \tilde{O}(n^{2.686})$ time [6]. Our algorithm for APNP matches this running time, so any further improvement on APNP will imply a faster algorithm for APBP and $(\max, \min)$-matrix product as well.

The *vertex-weighted* APNP problem on directed graphs, a restricted version of APNP, is computationally equivalent to the problem of *Maximum Witness for Boolean Matrix Multiplication* (MWBMM) [13]. An algorithm of $O(n^{2+\mu})$ time for MWBMM was given by Czumaj et al. [3], where μ satisfies the equation $\omega(1, \mu, 1) = 1 + 2\mu$ and $\omega(1, \mu, 1)$ is the exponent of multiplying an $n \times n^{\mu}$ matrix with an $n^{\mu} \times n$ matrix. Currently, the best bounds on rectangular matrix multiplication by Le Gall and Urrutia [7] imply that $\mu < 0.5286$.

1.1 Our results

In this paper we describe a faster algorithm for directed APNP problem running in $\tilde{O}(n^{(3+\omega)/2})$ time, which reaches optimal if the algorithm for APBP cannot be improved.

▶ **Theorem 1.** *The all pairs non-decreasing paths (APNP) problem on directed simple graphs can be solved in $\tilde{O}(n^{(3+\omega)/2})$ time.*

As in Dijkstra search [4] we can maintain a priority queue of current non-decreasing paths we have found, then the minimum unvisited one is the optimal paths between its endpoints. Every time we visit an optimal path, we "relax" all edges following that path. In [5] they partition the edge set into some parts by increasing order. For low-degree vertices in one part, trivially relax all of its outgoing edges, while for high-degree ones, use matrix multiplication to accelerate. Our algorithm adapt this idea, but we recursively divide the edges to $O(\log n)$ levels. In order the optimize the running time, a careful analysis of matrix multiplication is needed, and we need new ideas to use fast matrix multiplication to "predetermine" some of the useless edges from high-degree vertices.

We also give a $\tilde{O}(n^2)$ time algorithm for undirected APNP using the dynamic sequence data structure [10], which also reaches optimal within logarithmic factors.

▶ **Theorem 2.** *The all pairs non-decreasing paths (APNP) problem on undirected simple graphs can be solved in $\tilde{O}(n^2)$ time.*

[1] The $(\min, \leq)$-product of two matrices A, B is defined as $C_{i,j} = \min_k\{B_{k,j} : A_{i,k} \leq B_{k,j}\}$.

1.2 Organization of this paper

In Section 2, we introduce some terminologies used throughout this paper, and discuss how to recursively divide the edges. Then Section 3 and 4 will discuss our APNP algorithm for directed graphs in detail, where Section 3 explains the main techniques used, and Section 4 describes our whole algorithm and its analysis using procedures in Section 3 as subroutines. Due to page limit, the algorithm for undirected graphs will be discussed in the full version of this paper[2].

2 Preliminaries

2.1 Basic definitions

APNP problem

Let $G = (V, E)$ be a directed simple graph with edge weight $w(e)$ for each edge $e \in E$. We denote $n = |V|$ and $m = |E| = O(n^2)$.

A path is a sequence of edges $e_1, e_2, \ldots, e_l$. A non-decreasing path is a path satisfying $\forall 1 \leq i \leq l-1$, $w(e_i) \leq w(e_{i+1})$, and the weight of this non-decreasing path is defined to be $w(e_l)$, the weight of the last edge. All pairs non-decreasing paths problem asks to determine the minimum weight non-decreasing path between every pair of vertices. Let $OPT(i, j)$ denote the optimal (minimum) non-decreasing path between i and j. In contrast, during the algorithm, we use $d(i, j)$ to denote the current minimum non-decreasing path that our algorithm has found so far. In our algorithm, instead of explicitly maintaining the paths, we only need to maintain the weights of the paths $d(i, j)$ and $OPT(i, j)$ (and their last edges if we want to retrieve the paths), so we will also use $d(i, j)$ and $OPT(i, j)$ to denote the weights of corresponding paths $d(i, j)$ and $OPT(i, j)$, respectively. (Remember that $w(j, k)$ denotes the weight of the edge (j, k).)

Notation for String and Subgraph

Without loss of generality[3], we assume all edges have distinct edge weights ranged from 0 to $2^b - 1$, where $b = \lceil \log_2 |E| \rceil \leq 2 \log_2 n + O(1)$, so $2^b = O(n^2)$. Then every edge weight $w(e)$ corresponds to a b-bit 0-1 string $[w(e)]$ which is its binary representation, with the rightmost bit being the lowest bit.

To distinguish between values and strings, string s is written as $[s]$. $LCP([w_1], [w_2])$ (which is also a binary string) is the longest common prefix of $[w_1]$ and $[w_2]$. For example $LCP([100], [101]) = [10]$, and $LCP([000], [100]) = [\,]$(empty string). $|[w]|$ denotes the length of the string $[w]$, and $[x][y]$ denotes the concatenation of two strings $[x]$ and $[y]$. $[x] < [y]$ means that the lexicographical order of $[x]$ is smaller. For example, $[0111] < [101] < [1010]$.

We define $E_{[x]} = \{e \in E \mid [w(e)] \text{ has prefix } [x]\}$ to be the set of all edges whose weight has prefix $[x]$ (also call the edges have prefix $[x]$), and similarly the subgraph $G_{[x]} = (V, E_{[x]})$. For convenience, an edge set or a subgraph with subscript $[x]$ means all of the edges in it have prefix $[x]$.

[2] See `https://arxiv.org/abs/1904.10701`.
[3] See the full version of this paper for the proof.

Rectangular Matrix Multiplication

We use $M(m,n,p)$ to denote the asymptotic time complexity of multiplying an $m \times n$ matrix with an $n \times p$ matrix. We denote $M(n,n,n) = O(n^\omega)$, where $\omega < 2.373$ [2, 14, 8]. In this paper, rectangular matrix multiplications are straightforwardly reduced to square matrix multiplications. So we will only use the following fact. (Here $\min(x_1, \cdots, x_k)$ means the minimum number in $\{x_1, \cdots, x_k\}$.)

▶ **Lemma 3.** $M(m,n,p) = O\left(\frac{mnp}{\min(m,n,p)^{3-\omega}}\right)$.

Proof. The rectangular matrices are decomposed into $\min(m,n,p) \times \min(m,n,p)$ square matrices, then standard fast square matrix multiplication is applied. ◀

2.2 Naïve Algorithm

Let us first take a look at the naïve algorithm for APNP, a simple Dijkstra-type search [4]. Initially set $d(i,j) = w(i,j)$ for all edges $(i,j) \in E$, and $d(i,j) = +\infty$ otherwise. Each time we visit the minimum unvisited $d(i,j)$ and enumerate every out-going edge (j,k) of vertex j. If $d(i,j)$ and (j,k) form a nondecreasing path, then we update $d(i,k) \leftarrow \min(d(i,k), w(j,k))$. (See Algorithm 1.) We refer to this update step as *relaxing* edge (j,k) w.r.t. $d(i,j)$. By the greedy nature of Dijkstra search, when we visit $d(i,j)$, $d(i,j) = OPT(i,j)$. Namely, it is optimal.

Algorithm 1 Relaxing edges naïvely.

```
1: while there exists unvisited d(i,j) do
2:     visit the minimum unvisited d(i,j)
3:     for every (j,k) such that w(j,k) > d(i,j) do
4:         perform update d(i,k) ← min(d(i,k), w(j,k))
```

For clarity, our usages of symbols i, j, k stick to the following convention: $d(i,k)$ refers to the path being updated by the concatenation of path $d(i,j)$ and edge (j,k).

The naïve algorithm takes $O(n^3)$ time when edge weights are integers from 0 to $2^b - 1$, since a bucket heap is enough to maintain unvisited $d(i,j)$.

2.3 Classifying edges according to degrees

In order to avoid relaxing all edges when visiting $d(i,j)$, our algorithm will classify the edges based on the degrees of their two endpoints, and partition the edges whose both endpoints have high degrees into two sets by the order of their weights, then recursively deal with these two sets. We relax different types of edges by different approaches. First, we define "high-degree" and "low-degree" vertices in a subgraph.

High degree and low degree

In this paper, sometimes we use a subset of edges $E' \subseteq E$ to denote the subgraph $G' = (V', E')$ where V' is the set of vertices associated with E', namely, vertices in E' refers to vertices in V'. In a subgraph $G' = (V', E')$ of G, a vertex has high outdegree if its outdegree is larger than n^{1-t}, and otherwise, it has low outdegree. Here t is a parameter to be determined later (we will choose $t = \frac{3-\omega}{2}$). Similarly, a vertex has high indegree if its indegree is larger than n^{1-t}, and otherwise, it has low indegree. In our algorithm, edges (j,k) in a subgraph are divided into three types based on the outdegree of j and indegree of k:

- Low edge: if j has low outdegree
- High-high edge: if j has high outdegree and k has high indegree
- High-low edge: if j has high outdegree and k has low indegree

Divide the edges

In this binary partition procedure, starting from the entire edge set $E'_{[\,]} = E$, each time we divide the set of high-high edges $H_{[x]}$ in $E'_{[x]}$ into two parts: $H'_{[x][0]}$ and $H'_{[x][1]}$, based on its next bit after prefix $[x]$, then recursively partition the edge sets $H'_{[x][0]}$ and $H'_{[x][1]}$. Here we use $L_{[x]}$ and $\Gamma_{[x]}$ to denote the low edges and high-low edges with prefix $[x]$ obtained in the algorithm.

Algorithm 2 Divide the edges.

procedure DIVIDE($E'_{[x]}$)
  Consider the indegrees and outdegrees of vertices in the graph $G'_{[x]} = (V, E'_{[x]})$;
  Let $L_{[x]}$ be the set of edges from low outdegree vertices;
  Let $H_{[x]}$ be the set of edges from high outdegree vertices to high indegree vertices;
  Let $\Gamma_{[x]}$ be the set of edges from high outdegree vertices to low indegree vertices;
  Let $H'_{[x][0]} = \{(j,k) \in H_{[x]}$ s.t. $[w(j,k)]$ has prefix $[x][0]\}$ and $H'_{[x][1]} = \{(j,k) \in H_{[x]}$ s.t. $[w(j,k)]$ has prefix $[x][1]\}$.
  DIVIDE($H'_{[x][0]}$)
  DIVIDE($H'_{[x][1]}$)
end procedure

At the beginning we call DIVIDE(E). Since only the edges in $H_{[x]}$ go into the next recursion, every edge can only appear in one of $L_{[x]}$ or $\Gamma_{[x]}$ but can appear in many $H_{[x]}$.

2.4 Outline of our algorithm

In our algorithm, we run a Dijkstra-type procedure. When visiting $d(i,j)$, which is guaranteed to be optimal, we relax all the edges (j,k) in $L_{[x]}$ and $\Gamma_{[x]}$ w.r.t. $d(i,j)$ for all $[x]$ which is a prefix of $[d(i,j)]$. The outdegree of j is small in $L_{[x]}$ but not in $\Gamma_{[x]}$, so to relax $L_{[x]}$, we relax all edges from j as the naïve algorithm. But to relax $\Gamma_{[x]}$, a preprocessing procedure of the edges in $\Gamma_{[x]}$ is needed in order to save time. For edges in $H_{[x]}$, instead of immediately relaxing them, we wait until all optimal paths with prefix $[x][0]$ are visited, then perform a $(\min, \leq)$-matrix product of these paths with the adjacency matrix of $H'_{[x][1]}$ to relax all edges in it. Details of these methods will be given in Section 3.

3 Basic techniques

In this section, we explain the method we use for relaxing edges. Suppose we are trying to relax edges (j,k) w.r.t. $d(i,j)$, it will be based on whether (j,k) is in $L_{[x]}$, $H_{[x]}$, or $\Gamma_{[x]}$.

As explained before, $L_{[x]}$ is easiest. Since j has low outdegree, like the naïve algorithm, we simply relax all of its outgoing edges, which takes $O(n^{1-t})$ time. $H_{[x]}$ and $\Gamma_{[x]}$ are handled by different methods, though they both use the "row/column balancing" technique of matrix proposed in [6]. Instead of considering it as splitting rows/columns on matrix, we describe the balancing technique as splitting vertices so that every vertex has low outdegree/indegree. In the following subsections we describe how to handle $H_{[x]}$ and $\Gamma_{[x]}$.

Table 1 Balancing the indegrees of vertices k.

Balancing the indegrees of vertices
(S1) For every k, sort all the in-coming edges of k in increasing order and obtain the sorted list L_k.
(S2) Split k into vertices $k'_1, k'_2, \cdots k'_p$, and divide L_k into segments each of size n^{1-t} (while the last segment is possibly incomplete). The edges in the r-th segment is assigned to vertex k'_r. Namely, edge (j,k) in the r-th segment now becomes edge (j,k'_r).
(S3) Let $[L(k'_r), R(k'_r)]$ denote the weight range of in-coming edges of k'_r. Namely $L(k'_r)$ is the minimum weight of in-coming edges of k'_r, and $R(k'_r)$ is the maximum. Obviously, $L(k'_1) \le R(k'_1) < L(k'_2) \le R(k'_2) < \cdots < L(k'_p) \le R(k'_p)$.

3.1 Balancing

A graph $G = (V, E)$ contains at most $\frac{|E|}{n^{1-t}}$ vertices with high indegrees or outdegrees. If there are $\Theta(\frac{|E|}{n^{1-t}})$ number of vertices with high degrees, we would expect an average degree of $O(n^{1-t})$. However, the degrees of some vertices may be far greater than n^{1-t}. To balance the indegree (outdegree) of each vertex, we split every vertex into several vertices each of indegree (outdegree) n^{1-t} and one vertex of indegree (outdegree) $\le n^{1-t}$. The number of new vertices with indegree (outdegree) exactly n^{1-t} is bounded by $O\left(\frac{|E|}{n^{1-t}}\right)$, and every original high indegree (outdegree) vertex corresponds to at most one new vertex with indegree (outdegree) $< n^{1-t}$, thus we have at most $O\left(\frac{|E|}{n^{1-t}}\right)$ many new vertices each with indegree (outdegree) $\le n^{1-t}$.

In our algorithm, for edge set $\{(j,k)\}$, we use this technique to either balance the outdegrees of vertices j or the indegrees of vertices k. Here we demonstrate this technique for balancing indegrees of k as an example in Table 1. Denote the set of edges after balancing to be $\bar{E}$, then the graph corresponding to $\bar{E}$ is actually a bipartite graph with edges between vertices j and k'_r. The procedure for balancing outdegrees of j is symmetric.

3.2 High-high edges

The technique for high-high edges solves the following problem: Given a set P of optimal paths of prefix $[x][0]$ and a set $H'_{[x][1]}$ of high-high edges, the problem asks to relax all edges in $H'_{[x][1]}$ w.r.t. paths in P, namely, extend paths in P by a single edge in $H'_{[x][1]}$.

This problem is equivalent with a length-two nondecreasing path problem. As discussed in Section 1, this can be solved by fast $(\min, \le)$-product implied in [6]. But the rectangular version is not covered by [6], so we fully describe the algorithm and its analysis (in graphs).

We have two extra guarantees when we use this procedure in our main algorithm:

- P is the set of optimal paths $OPT(i,j)$ such that $[OPT(i,j)]$ has prefix $[x][0]$, so any path $d(i,j)$ in P can form a nondecreasing path with any edge (j,k) in $H'_{[x][1]}$.
- All $d(i,j)$ satisfying $[d(i,j)] < [x][1]$ are already visited by Dijkstra search. So we can tell whether $[OPT(i,j)] < [x][1]$ or not.

Suppose there are $n_{[x][1]}$ many $[OPT(i,j)]$ which have $[x][1]$ as a prefix. In our algorithm the time complexity will depend on $n_{[x][1]}$.

The first step is to apply the balancing technique in Section 3.1 to indegrees of vertices k in $H'_{[x][1]}$, and let the edge set after balancing be $\bar{H'}_{[x][1]}$. Here each high indegree vertex

k is split into vertices $\{k'_r\}$. We define the following two matrices: (j and k'_r only include vertices having out-going edges and in-coming edges in $\bar{H}'_{[x][1]}$, respectively.)

$$A_{i,j} = \begin{cases} 1 & d(i,j) \in P \\ 0 & \text{otherwise} \end{cases} \qquad B_{j,k'_r} = \begin{cases} 1 & (j,k'_r) \in \bar{H}'_{[x][1]} \\ 0 & \text{otherwise} \end{cases}$$

Then we multiply them with rectangular matrix multiplication. Let $C = AB$, then,

$$C_{i,k'_r} \begin{cases} > 0 & \text{if there is a nondecreasing path from } i \text{ to } k'_r \\ = 0 & \text{otherwise} \end{cases}$$

For every pair (i,k) such that $[OPT(i,k)] \geq [x][1]$ ($d(i,j) \notin P$) and k is an in-coming vertex in $H'[x][1]$, we find the minimum r with $C_{i,k'_r} > 0$ and relax all n^{1-t} incoming edges (j,k'_r) of k'_r w.r.t. $d(i,j)$ if $d(i,j) \in P$. We can skip other $r' > r$ because $OPT(i,k) \leq R(k'_r) < L(k'_{r'})$. Namely, we only have to relax $O(n^{1-t})$ many edges for each (i,k), which is the benefit of balancing. We attribute this $O(n^{1-t})$ cost of relaxations to the $OPT(i,k)$ with prefix $[x][1]$ found in these relaxations, which must exist if $C_{i,k'_r} > 0$.

The procedure above consists of three parts: finding minimum r for each (i,k), relaxation and matrix multiplication. The time cost for the first part is at most enumerating all nonzero elements of C, so it is dominated by the matrix multiplication part.

▶ **Lemma 4.** *The relaxation part takes $O\left(n_{[x][1]} n^{1-t}\right)$ time in total.*

Proof. The cost of relaxation is attributed to each $OPT(i,k)$ with prefix $[x][1]$. Since there are $n_{[x][1]}$ many $OPT(i,k)$ with prefix $[x][1]$, and each corresponds to $O(n^{1-t})$ relaxations, it costs $O(n_{[x][1]} n^{1-t})$ time in total. ◀

▶ **Lemma 5.** *In the matrix multiplication part, A is an $n \times O\left(\min\left(n, \frac{2^{b-|[x]|}}{n^{1-t}}\right)\right)$ matrix, and B is a $O\left(\min\left(n, \frac{2^{b-|[x]|}}{n^{1-t}}\right)\right) \times O\left(\frac{2^{b-|[x]|}}{n^{1-t}}\right)$ matrix, so the time complexity for matrix multiplication is $M\left(n, \min\left(n, \frac{2^{b-|[x]|}}{n^{1-t}}\right), \frac{2^{b-|[x]|}}{n^{1-t}}\right)$.*

Proof. There are at most $\min\left(n, \frac{|H'_{[x][1]}|}{n^{1-t}}\right)$ many j because vertices j have high outdegrees. After balancing, there are at most $O\left(\frac{|H'_{[x][1]}|}{n^{1-t}}\right)$ many k'_r by discussion in Section 3.1. Since each edge in $H'_{[x][1]}$ has prefix $[x][1]$, $|H'_{[x][1]}| \leq 2^{b-|[x]|-1} = O\left(2^{b-|[x]|}\right)$. Plug in the size of $H'_{[x][1]}$ gives the desired bound. ◀

3.3 High-low edges

Now we consider the relaxation of the high-low edges in $\Gamma_{[x]}$ when visiting paths with the same prefix $[x]$. To preprocess high-low edges, we run an initialization step when all optimal paths less than $[x]$ have been visited. As before, we denote the number of $[OPT(i,j)]$ with prefix $[x]$ by $n_{[x]}$.

Since the outdegrees of vertices j are high, we cannot relax edges one by one. But we still want to utilize the property that the indegrees of k are low. As in the last subsection, we denote the set of optimal paths we have found to be P, namely when we visit $d(i,j)$, we add $d(i,j)$ to P. By the nature of Dijkstra search, such $d(i,j)$ is always visited in increasing order. At the initialization step, P is the set of all optimal paths less than $[x]$. During the procedure, optimal paths with prefix $[x]$ are added to P.

We also maintain a dynamic set Q which is initially empty. When we relax an edge (j,k) w.r.t. $d(i,j)$, we put $d(i,k)$ into Q only if $[d(i,k)] \geq [x]$, that is, $d(i,k)$ was not visited at initialization. So Q contains new nondecreasing path but not guaranteed to be optimal.

Initialization by Matrix Multiplication

The first step is to apply the balancing technique to the outdegrees of vertices j in $\Gamma_{[x]}$ such that every vertex j in $\Gamma_{[x]}$ is split into a sequence of vertices $\{j'_r\}$. Suppose the edge set after balancing is $\bar{\Gamma}_{[x]}$. Then, we define the following two matrices: (j'_r and k only include vertices having out-going edges and in-coming edges in $\bar{\Gamma}_{[x]}$, respectively.)

$$A_{i,k} = \begin{cases} 1 & d(i,k) \notin P \cup Q \\ 0 & \text{otherwise} \end{cases} \qquad B_{k,j'_r} = \begin{cases} 1 & (j'_r, k) \in \bar{\Gamma}_{[x]} \\ 0 & \text{otherwise} \end{cases}$$

We compute $C = AB$. Basically speaking, the matrix C can indicate whether we need to relax edges (j'_r, k) from j'_r when visiting $d(i,j)$. Note that when we run initialization, Q is empty, so $d(i,k) \notin Q$ is trivially true. But we will add paths to Q and dynamically update the matrix multiplication later.

We make the following observation about C_{i,j'_r}.

▶ **Observation 1.** *If $d(i,j) < L(j'_r)$ and $C_{i,j'_r} > 0$, there is at least one edge (j'_r, k) such that $d(i,k) \notin P \cup Q$ and $[x]$ is a prefix of $[OPT(i,k)]$.*
Conversely, if $C_{i,j'_r} = 0$, there is no such an edge (j'_r, k).

Proof. Since $C_{i,j'_r} > 0$, at least one vertex k satisfy the following:
- $A_{i,k} > 0$: $d(i,k) \notin P$, so $[OPT(i,k)] \geq [x][0 \cdots 0]$. Also $d(i,k) \notin Q$.
- $B_{k,j'_r} > 0$: $(j'_r, k) \in \bar{\Gamma}_{[x]}$.

Since $d(i,j) < L(j'_r)$, $d(i,j)$ and the original edge of (j'_r, k) form a non-decreasing path. So $[OPT(i,k)] \leq [x][1 \cdots 1]$.

Conversely, if $C_{i,j'_r} = 0$, for every k, at least one of these happens:
- $A_{i,k} = 0$: $d(i,k) \in P \cup Q$
- $B_{k,j'_r} = 0$: edge (j'_r, k) does not exist in $\bar{\Gamma}_{[x]}$. ◀

Update matrix multiplication

When a new path $d(i,k)$ is added to P or Q, the matrix A and product C need to be updated. Adding a new path to P or Q only changes one entry of $A_{i,k}$, so we utilize the low indegree of k. There are at most $O(n^{1-t})$ many j'_r such that $B_{k,j'_r} \neq 0$, since $\Gamma_{[x]}$ contains high-low edges only. So the update of C when changing one element $A_{i,k}$ takes only $O(n^{1-t})$ time by enumerating all nonzero B_{k,j'_r}. This cost can be attributed to each $[OPT(i,k)]$ with prefix $[x]$, since every such path can only be added to $P \cup Q$ once. However, adding $d(i,k)$ to Q does not mean we have found the optimal path $OPT(i,k)$, as it can still be updated. How to deal with this will be discussed later.

Relaxation when visiting $d(i,j)$

When we visit $d(i,j)$, for each split vertex j'_r of j, there are three cases:
1. $d(i,j) > R(j'_r)$: $d(i,j)$ cannot form a non-decreasing path with any out-going edge of j'_r, so we skip j'_r.
2. $d(i,j) \in [L(j'_r), R(j'_r)]$: We relax all out-going edges of j'_r larger than $d(i,j)$.
3. $d(i,j) < L(j'_r)$: Only when $C_{i,j'_r} > 0$, we relax all out-going edges of j'_r one by one.

By Observation 1, when $C_{i,j'_r} = 0$, for each edge (j'_r, k), either $d(i,k) \in P$ or $d(i,k) \in Q$. If $d(i,k) \in P$, it needs no more update. If $d(i,k) \in Q$, roughly speaking, since k has indegree less than n^{1-t}, the updates for $d(i,k)$ can be done "in advance" when it is added to Q. The details will be clear later. This is why we can skip j'_r when $C_{i,j'_r} = 0$.

Since the degree of j'_r is bounded by n^{1-t}, the relaxation takes $O(n^{1-t})$ time for each j'_r. For the second case, it only happens once for each $d(i,j)$ because $[L(j'_r), R(j'_r)]$ are disjoint for different j'_r, so the $O(n^{1-t})$ cost is attributed to $d(i,j)$. For the third case, by Observation 1, there is at least one edge (j'_r, k) such that $OPT(i,k) \notin P \cup Q$ with prefix $[x]$, so the $O(n^{1-t})$ time is attributed to $d(i,k)$. (If there is more than one such k, choose an arbitrary one.) Then $d(i,k)$ is added to Q, and the cost of updating A and C is also bounded by n^{1-t}, dominated by the cost of relaxation.

Because the first path $d(i,k)$ we found for each (i,k) is not necessarily the optimal one, we discuss how to handle all future updates of $d(i,k)$ "in advance" after adding it to Q. We enumerate every in-coming edge $(j'', k) \in \Gamma_{[x]}$ of k. If $d(i,j'')$ is not in P, we add (i,k) to a *waiting list* for (i,j''), denoted by $W(i,j'')$. When $d(i,j)$ is visited in the future, we can go through its waiting list $W(i,j)$ and update $d(i,k)$ for all pair (i,k) in the list. There are only n^{1-t} in-coming edges for k, so the waiting list construction cost is also $O(n^{1-t})$ for every $d(i,k)$.

In conclusion, we follow the procedure in Algorithm 3 when visiting $d(i,j)$ with prefix $[x]$.

Algorithm 3 High-low relaxation when visiting $d(i,j)$.

```
1:  Add d(i,j) to P and update A and C = AB.
2:  for (i,k) in the waiting list W(i,j) do
3:      Relax (j,k) w.r.t. d(i,j) if w(j,k) > d(i,j)
4:  for every j'_r satisfying d(i,j) ∈ [L(j'_r), R(k'_r)] or (d(i,j) < L(j'_r) and C_{i,j'_r} > 0) do
5:      for every outgoing edge (j'_r, k) of j'_r larger than d(i,j) do
6:          if d(i,k) ∉ P ∪ Q then
7:              Relax (the original edge of) (j'_r, k) w.r.t. d(i,j)
8:              Add d(i,k) to Q and update A and C = AB
9:              for incoming edge (j'', k) of k do
10:                 Add (i,k) to the waiting list W(i,j'') if d(i,j'') ∉ P
```

Complexity

This procedure is divided into matrix multiplication part (initialization) and relaxation part (Algorithm 3) as well.

▶ **Lemma 6.** *The relaxation part takes $O\left(n_{[x]}n^{1-t}\right)$ time.*

Proof. From discussion above, the $O(n^{1-t})$ cost of each relaxation is either attributed to optimal $d(i,j)$ with prefix $[x]$ or $d(i,k) \in Q$. For each $d(i,k) \in Q$, $OPT(i,k)$ is of course larger than $[x][0\cdots0]$, and then relaxed by an edge $\leq [x][1\cdots1]$, so the size of Q is also bounded by $n_{[x]}$. Since each $d(i,j)$ can only be added to P and Q once, respectively, the total time is $O\left(n_{[x]}n^{1-t}\right)$.

The total size of all waiting lists $W(i,j)$ is bounded by $O\left(n_{[x]}n^{1-t}\right)$ as well, because each time when an $d(i,k)$ is added to Q, we enumerate $\leq n^{1-t}$ many incoming edges of k, and add (i,k) to waiting list at Line 10. Every waiting list can only be relaxed once, thus, the relaxation of waiting list edges in Line 3 needs $O\left(n_{[x]}n^{1-t}\right)$ in total. ◀

The following lemma is crucial. Although edges (j,k) in $\Gamma_{[x]}$ are high-low edges, the number of k is not directly bounded, but remind that in our binary partition of edges, only high-high edges of previous level can be in this set, thus in fact the number of k *cannot* be asymptotically larger than the number of j'_r.

▶ **Lemma 7.** *In the matrix multiplication part, A is an $n \times O\left(\min\left(n, \frac{2^{b-|[x]|}}{n^{1-t}}\right)\right)$ matrix, and B is an $O\left(\min\left(n, \frac{2^{b-|[x]|}}{n^{1-t}}\right)\right) \times O\left(\frac{2^{b-|[x]|}}{n^{1-t}}\right)$ matrix, so the time complexity of matrix multiplication is $M\left(n, \min\left(n, \frac{2^{b-|[x]|}}{n^{1-t}}\right), \frac{2^{b-|[x]|}}{n^{1-t}}\right)$.*

Proof. Since $|\Gamma_{[x]}| \leq 2^{b-|[x]|}$ (because each edge in it has prefix $[x]$), after balancing, there are at most $O\left(\frac{2^{b-|[x]|}}{n^{1-t}}\right)$ many j'_r.

If $[x] \neq []$, suppose $[x] = [x'][0/1]$, namely $[x']$ is the prefix of $[x]$ which is one bit shorter. If $(j,k) \in \Gamma_{[x]}$, by Algorithm 2, $(j,k) \in H_{[x']}$. Since k has high indegree in $H_{[x']}$, the number of such k is bounded by $O\left(\min\left(n, \frac{|H_{[x']}|}{n^{1-t}}\right)\right)$. Also $|H_{[x']}| = O\left(2^{b-|[x]|}\right)$. Plug it in gives the $O\left(\min\left(n, \frac{2^{b-|[x]|}}{n^{1-t}}\right)\right)$ bound for the number of k. If $[x] = []$, of course the number of k is bounded by n. ◀

4 Main algorithm for directed graphs and analysis

4.1 Main algorithm

Just like in the naïve algorithm, we use a bucket to maintain all $d(i,j)$ we have found, and the minimal *unvisited* $d(i,j)$ is guaranteed to be optimal. Our algorithm enumerates the value x from 0 to $2^b - 1$ and visit $d(i,j)$ if $d(i,j) = x$. We carefully combine techniques introduced in the previous section into this framework.

Recall that in Algorithm 2 of Section 2.3 we define $L_{[x]}$ to be the set of low edges (from low outdegree vertices) and $\Gamma_{[x]}$ to be the set of high-low edges in $H'_{[x]}$, which are high-high edges in higher level, then divide the edge set $H_{[x]}$ of high-high edges in $H'_{[x]}$ to $H'_{[x][0]}$ and $H'_{[x][1]}$ and recursively deal with them. Our main algorithm is presented in Algorithm 4.

Algorithm 4 Main algorithm.

```
d(i,j) = w(i,j) for all edges (i,j) ∈ E, and d(i,j) = +∞ otherwise
for x from 0 to 2^b − 1 do
    for all prefix [y] of [x] do
        if [x] = [y][000···0] then
            Run high-low edge initialization for edges in Γ_[y]
        if [x] = [y][100···0] then
            P_[y][0] = {d(i,j) | [y][0] is a prefix of [d(i,j)]}
            Append high-high edges in H'_[y][1] to paths in P_[y][0]
    for d(i,j) = x do
        Mark d(i,j) as visited, add d(i,j) to P.
        for all prefix [y] of [d(i,j)] do
            Relax edges (j,k) ∈ L_[y]
            Relax edges (j,k) ∈ Γ_[y] by Algorithm 3
```

When visiting an optimal path $d(i,j)$, we need to relax all edges (j,k) which are larger than $d(i,j)$.

▶ **Observation 2.** *For $[d(i,j)] = [x]$, every edge (j,k) larger than $d(i,j)$ must be one of the following three cases: (so $[y]$ is a prefix of both $[x]$ and $[w(j,k)]$.)*

- *$(j,k) \in L_{[y]}$ for some prefix $[y]$ of $[x]$*
- *$(j,k) \in \Gamma_{[y]}$ for some prefix $[y]$ of $[x]$*
- *$(j,k) \in H'_{[y][1]}$ where $[y][0]$ is a prefix of $[x]$*

Proof. Consider the longest common prefix $[y] = LCP([x], [w(j,k)])$. If (j,k) is not in the $L_{[y']}$ or $\Gamma_{[y']}$ for any prefix $[y']$ of $[y]$, it must be in $H_{[y]}$. Since $[y]$ is the longest common prefix and $w(j,k)$ is larger than $d(i,j)$, $[d(i,j)]$ has prefix $[y][0]$ and $[w(j,k)]$ has prefix $[y][1]$, thus (j,k) is in $H'_{[y][1]}$. ◀

Thus, we can simply relax the edges of the first type, and use the method in Section 3.2 to relax the edges in $H'_{[y][1]}$ when all of the optimal paths with prefix $[y][0]$ have been visited. The method for high-low edges $\Gamma_{[y]}$ is like a dynamic data structure: we initialize it when $[x] = [y][0 \cdots 0]$, and update it when relaxing an edge in $\Gamma_{[y]}$.

High-high edges

For high-high edges, when $[x] = [y][100 \cdots 0]$, before those $d(i,j) = x$ are visited, we append edges in $H'_{[y][1]}$ to paths in $P_{[y][0]} = \{d(i,j) \mid [y][0] \text{ is a prefix of } [d(i,j)]\}$ using the technique introduced in Section 3.2. See Line 8, Algorithm 4.

In Section 3.2, we have two guarantees. Now we check them one by one:

- Because each edge in $H'_{[y][1]}$ has prefix $[y][1]$, and each path in $P_{[y][0]}$ has prefix $[y][0]$, the maximum weight in $P_{[y][0]}$ is smaller than the minimum weight of $H'_{[y][1]}$. At the time of $[x] = [y][100 \cdots 0]$, all paths in $P_{[y][0]}$ are optimal.
- Since $[x] = [y][100 \cdots 0]$, all $[d(i,j)] < [y][1]$ are visited, and none of $[d(i,j)] \geq [y][1]$ are visited yet.

High-low edges

We initialize for $\Gamma_{[y]}$ when $[x] = [y][000 \cdots 0]$ before we visit those $d(i,j) = x$. See Line 5 of Algorithm 4. All $[d(i,j)] < [y]$ are visited, and none of $[d(i,j)] \geq [y]$ are visited. Once a $d(i,j)$ within the range $[y][000 \cdots 0] \sim [y][111 \cdots 1]$ is visited, we use the approach in Algorithm 3.

4.2 Correctness

We now prove the correctness of our algorithm. Suppose the last edge of $OPT(i,k)$ is (j,k). Then $d(i,k)$ is correctly computed before it is visited if and only if the following two conditions holds:

- If $i \neq j$, $d(i,j)$ is correctly computed before it is visited.
- After $d(i,j)$ is visited, before we visit $d(i,k)$, $d(i,k)$ is updated by relaxing the edge (j,k) w.r.t. $d(i,j)$.

We prove the second condition holds for every $d(i,j)$, and the first one simply follows from induction.

Suppose $[z] = LCP([OPT(i,j)], [OPT(i,k)]) = LCP([OPT(i,j)], [w(j,k)])$. By Observation 2 the last edge (j,k) must be in one of the three cases, so we check them one by one.

▶ **Lemma 8.** *If (j,k) is in $L_{[y]}$ for some prefix $[y]$ of $[z]$, and $d(i,j)$ is correctly computed before visited, then $d(i,k)$ is also correctly computed before visited.*

Proof. Because $[y]$ is also a prefix of $[OPT(i,j)]$, at Line 12, when we visit $d(i,j)$, $d(i,k)$ is updated by relaxing (j,k). Because $OPT(i,j) < OPT(i,k)$, $d(i,k)$ is not visited yet. ◀

▶ **Lemma 9.** *Suppose (j,k) is in $H'_{[y][1]}$ where $[y][0]$ is a prefix of $[OPT(i,j)]$. If $d(i,j)$ is correctly computed before visited, $d(i,k)$ is also correctly computed before visited.*

Proof. We can see $[y] = [z]$ from the proof of Observation 2. At Line 8, when $[x] = [y][100\cdots0]$, $d(i,k)$ is updated by $d(i,j)$ and (j,k). $d(i,j)$ is already visited before because it has prefix $[y][0]$. $d(i,k)$ will be visited later because it has prefix $[y][1]$. ◀

▶ **Lemma 10.** *If (j,k) is in $\Gamma_{[y]}$ for some prefix $[y]$ of $[z]$, and $d(i,j)$ is correctly computed before visited, then $d(i,k)$ is also correctly computed before visited.*

Proof. Since $[y]$ is a prefix of both $[OPT(i,j)]$ and $[w(j,k)]$, the initialization for $\Gamma_{[y]}$ is done at Line 5 when $[x] = [y][000\cdots0]$. After that, $d(i,k)$ is updated when we visit $d(i,j)$ at Line 13. $d(i,k)$ is not visited yet because $OPT(i,k) > OPT(i,j)$. ◀

▶ **Lemma 11.** *All $d(i,j)$ are correctly computed before visited.*

Proof. This follows from a simple induction. In the base case, for all length 1 optimal paths $OPT(i,j)$, they are obviously correctly computed in Line 1. Then if all length $l-1$ paths $OPT(i,j)$ are correctly computed before visited, by Lemma 8, 9, 10, all length l paths $OPT(i,j)$ are also correctly computed before visited. ◀

4.3 Running time

▶ **Lemma 12.** *The relaxation for low edges ($L_{[x]}$) takes $\tilde{O}\left(n^{3-t}\right)$ time in total. (Line 12)*

Proof. At Line 12, we only enumerate $O(n^{1-t})$ many edges because j has low outdegree in $L_{[y]}$. Since there are only $b = O(\log n)$ many prefix $[y]$ for each $[d(i,j)]$, each $d(i,j)$ takes $\tilde{O}(n^{1-t})$ time. So in total, these updates take $\tilde{O}(n^{3-t})$ time for all $O(n^2)$ many $d(i,j)$. ◀

▶ **Lemma 13.** *The relaxation for high-high edges and high-low edges besides matrix multiplication takes $\tilde{O}\left(n^{3-t}\right)$ time in total.*

Proof. By Lemma 4 and Lemma 6, for each $[y]$, the complexity for relaxation is bounded by $(n_{[y]} + n_{[y][1]})n^{1-t} = O\left(n_{[y]}n^{1-t}\right)$, where $n_{[y]}$ stands for the number of optimal paths with prefix $[y]$. Since an optimal path can be counted in $O(\log n)$ many $n_{[y]}$, the total time is therefore $\tilde{O}(n^{3-t})$. ◀

▶ **Lemma 14.** *The matrix multiplication parts for high-high edge updates and the initialization of high-low edge updates take $\tilde{O}\left(n^{t+\omega}\right)$ time in total.*

Proof. By Lemma 5 and Lemma 7, the complexity for matrix multiplication is at most $M\left(n, \min\left(\frac{2^{b-|[y]|}}{n^{1-t}}, n\right), \frac{2^{b-|[y]|}}{n^{1-t}}\right)$ for each $[y]$. We fix the length of $[y]$, denoted by $l = |[y]|$, then consider the two cases:

- $2^l < n^t$: There are at most 2^l many such $[y]$, and each takes

$$M\left(n, n, \frac{2^{b-l}}{n^{1-t}}\right) = O\left(\frac{n^2 \cdot \frac{2^{b-l}}{n^{1-t}}}{n^{3-\omega}}\right) = O\left(n^{t+\omega} \cdot 2^{-l}\right)$$

  This follows from both Lemma 3 and the fact that $2^b = |E| = O(n^2)$. For each l, the time complexity is exactly $O(n^{t+\omega})$. So the total complexity is $\tilde{O}(n^{t+\omega})$ since $l = O(\log_2(n))$.
- $2^l \geq n^t$: There are at most 2^l many such $[y]$. Each takes

$$\begin{aligned} M\left(n, \frac{2^{b-l}}{n^{1-t}}, \frac{2^{b-l}}{n^{1-t}}\right) &= O\left(n \cdot \left(\frac{2^{b-l}}{n^{1-t}}\right)^{2-(3-\omega)}\right) \\ &= O\left(n^{(t+1)(\omega-1)+1} \cdot 2^{-l(\omega-1)}\right) \end{aligned}$$

So for all $[y]$ of length l, it takes $O\left(n^{(t+1)(\omega-1)+1} \cdot 2^{-l(\omega-2)}\right)$ time. The term $2^{-l(\omega-2)}$ is maximized when l is minimized, so $2^{-l(\omega-2)} \leq n^{-t(\omega-2)}$, and the total time for all lengths of $[y]$ is

$$\tilde{O}\left(n^{(t+1)(\omega-1)+1-t(\omega-2)}\right) = \tilde{O}\left(n^{t+\omega}\right)$$ ◀

▶ **Theorem 15.** *The All Pair Non-decreasing Paths (APNP) problem on directed simple graphs can be solved in* $\tilde{O}\left(n^{\frac{3+\omega}{2}}\right)$ *time. The optimal path of length* l *between any two vertices can also be explicitly found in* $O(l)$ *time if we slightly modify the algorithm.*

Proof. We choose $t = \frac{3-\omega}{2}$. The running time of this algorithm follows from previous lemmas. Since all optimal paths $OPT(i,k)$ are obtained by relaxation of edges (j,k), we can store the last edge (j,k) for each $OPT(i,k)$, so retrieving the optimal path can be done in $O(l)$ time. ◀

References

1. Alfred V. Aho and John E. Hopcroft. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1974.
2. D. Coppersmith and S. Winograd. Matrix Multiplication via Arithmetic Progressions. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 1–6, New York, NY, USA, 1987. ACM. doi:10.1145/28395.28396.
3. Artur Czumaj, Mirosław Kowaluk, and Andrzej Lingas. Faster algorithms for finding lowest common ancestors in directed acyclic graphs. *Theoretical Computer Science*, 380(1-2):37–46, 2007.
4. Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.
5. Ran Duan, Yong Gu, and Le Zhang. Improved Time Bounds for All Pairs Non-decreasing Paths in General Digraphs. In *45th International Colloquium on Automata, Languages, and Programming*, pages 44:1–44:14, 2018.
6. Ran Duan and Seth Pettie. Fast algorithms for $(\max, \min)$-matrix multiplication and bottleneck shortest paths. In *Proceedings of the 20th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 384–391. SIAM, 2009.
7. François Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the Coppersmith-Winograd tensor. In *Proceedings of the 29th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1029–1046. SIAM, 2018.
8. François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, pages 296–303. ACM, 2014.
9. François Le Gall and Harumichi Nishimura. Quantum algorithms for matrix products over semirings. In *Scandinavian Workshop on Algorithm Theory*, pages 331–343. Springer, 2014.
10. Kurt Mehlhorn, Rajamani Sundar, and Christian Uhrig. Maintaining dynamic sequences under equality tests in polylogarithmic time. *Algorithmica*, 17(2):183–198, 1997.
11. George J. Minty. A Variant on the Shortest-Route Problem. *Operations Research*, 6(6):882–883, 1958.
12. Virginia Vassilevska, Ryan Williams, and Raphael Yuster. All Pairs Bottleneck Paths and Max-min Matrix Products in Truly Subcubic Time. *Theory of Computing*, 5(9):173–189, 2009.
13. Virginia Vassilevska Williams. Nondecreasing Paths in a Weighted Graph or: How to Optimally Read a Train Schedule. *ACM Trans. Algorithms*, 6(4):70:1–70:24, September 2010.
14. Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith–Winograd. In *Proceedings of the 44th annual ACM Symposium on Theory of Computing*, pages 887–898. ACM, 2012.
15. Uri Zwick. All Pairs Shortest Paths Using Bridging Sets and Rectangular Matrix Multiplication. *J. ACM*, 49(3):289–317, May 2002.

Faster Approximation Algorithms for Computing Shortest Cycles on Weighted Graphs

Guillaume Ducoffe
National Institute for Research and Development in Informatics, Romania
The Research Institute of the University of Bucharest ICUB, Romania
University of Bucharest, Romania
guillaume.ducoffe@ici.ro

Abstract

Given an n-vertex m-edge graph G with non-negative edge-weights, a *shortest cycle* of G is one minimizing the sum of the weights on its edges. The *girth* of G is the weight of such a shortest cycle. We obtain several new approximation algorithms for computing the girth of weighted graphs:

- For any graph G with polynomially bounded integer weights, we present a deterministic algorithm that computes, in $\tilde{\mathcal{O}}(n^{5/3} + m)$-time[1], a cycle of weight at most twice the girth of G. This matches both the approximation factor and – almost – the running time of the best known subquadratic-time approximation algorithm for the girth of *unweighted* graphs.
- Then, we turn our algorithm into a deterministic $(2+\varepsilon)$-approximation for graphs with arbitrary non-negative edge-weights, at the price of a slightly worse running-time in $\tilde{\mathcal{O}}(n^{5/3}\text{polylog}(1/\varepsilon) + m)$. For that, we introduce a generic method in order to obtain a polynomial-factor approximation of the girth in subquadratic time, that may be of independent interest.
- Finally, if we assume that the adjacency lists are sorted then we can get rid off the dependency in the number m of edges. Namely, we can transform our algorithms into an $\tilde{\mathcal{O}}(n^{5/3})$-time *randomized* 4-approximation for graphs with non-negative edge-weights. This can be derandomized, thereby leading to an $\tilde{\mathcal{O}}(n^{5/3})$-time deterministic 4-approximation for graphs with polynomially bounded integer weights, and an $\tilde{\mathcal{O}}(n^{5/3}\text{polylog}(1/\varepsilon))$-time deterministic $(4+\varepsilon)$-approximation for graphs with non-negative edge-weights.

To the best of our knowledge, these are the first known *subquadratic-time* approximation algorithms for computing the girth of weighted graphs.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Graph algorithms analysis; Theory of computation → Approximation algorithms analysis

Keywords and phrases girth, weighted graphs, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.49

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1810.10229`.

Funding This work was supported by a grant of Romanian Ministry of Research and Innovation CCCDI-UEFISCDI. project no. 17PCCDI/2018.

1 Introduction

The exciting program of "Hardness in P" aims at proving (under plausible complexity theoretic conjectures) the *exact* time-complexity of fundamental, polynomial-time solvable problems in computer science. In this paper, we consider the Girth problem on edge-weighted undirected graphs, for which almost all what is known in terms of finer-grained complexity

[1] The $\tilde{\mathcal{O}}(\cdot)$ notation suppresses polylogarithmic factors.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 49; pp. 49:1–49:13

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

only holds for dense graphs ($m = \Omega(n^2)$). We recall that the girth of a given graph G is the minimum weight of a cycle in G – with the weight of a cycle being defined as the sum of the weights on its edges (see Sec. 2 for any undefined terminology in this introduction). For dense graphs this parameter can be computed in time $\mathcal{O}(n^3)$, and Vassilevska Williams and Williams [19] proved a bunch of combinatorial subcubic equivalences between Girth and other path and matrix problems. In particular, for every $\varepsilon > 0$, there cannot exist any *combinatorial* $(4/3 - \varepsilon)$-approximation algorithm for Girth that runs in truly subcubic time unless there exists a truly subcubic combinatorial algorithm for multiplying two boolean matrices. Roditty and Tov completed this above hardness result with an $\tilde{\mathcal{O}}(n^2/\varepsilon)$-time $(4/3 + \varepsilon)$-approximation algorithm [15], thereby essentially completing the picture of what can be done combinatorially in subcubic time. However, the story does not end here for at least two reasons. A first simple but important observation is that as the graphs considered get sparser, the complexity for computing their girth falls down to $\mathcal{O}(n^2)$. In fact, when the edge-weights are integers bounded by some constant M, there is a non-combinatorial algorithm for computing the girth of *any* n-vertex graph G in time $\tilde{\mathcal{O}}(Mn^\omega)$ where ω stands for the the exponent of square matrix multiplication over a ring [16]. It is widely believed that $\omega = 2$ [13], and if true, that would imply we can compute the *exact* value of the girth in quasi *quadratic* time – at least when edge-weights are bounded. So far, all the *approximation* algorithms for Girth on weighted graphs run in $\tilde{\Omega}(n^2)$-time [10, 15]. This leads us to the following, natural research question:

Does there exist a subquadratic approximation algorithm for Girth *on weighted graphs?*

In this paper, we answer to this above question in the affirmative.

1.1 Our contributions

We present new approximation algorithms for the girth of graphs with non-negative real edge-weights. These are the first algorithms to break the quadratic barrier for this problem – at the price of a slightly worse approximation factor compared to the state of the art [15] – see Sec. 1.2 on the Related work for more details. Our first result is obtained for graphs with bounded integer edge-weights.

▶ **Theorem 1.** *For every $G = (V, E, w)$ with edge-weights in $\{1, \ldots, M\}$, we can compute a deterministic 2-approximation for* Girth *in time $\tilde{\mathcal{O}}(n^{5/3}\text{polylog}M + m)$.*

Our starting point for Theorem 1 is a previous 2-approximation algorithm from Lingas and Lundell [10], that runs in *quadratic* time. Specifically, these two authors introduced an $\tilde{O}(n \log M)$-time procedure that takes as entry a specified vertex of the graph and needs to be applied to *every* vertex in order to obtain the desired 2-approximation of the girth. Inspired by the techniques used for approximate distance oracles [18] we informally modify their algorithm as follows. We only apply their procedure to the vertices in a *random* subset S: where each vertex is present with equal probability $n^{-1/3}$ (we can derandomize our approach by using known techniques from the litterature [14]). Furthermore, for the vertices not in S, we rather apply a modified version of their procedure that is restricted to a small subgraph – induced by some ball of expected size $\mathcal{O}(n^{1/3})$. A careful analysis shows this is a 2-approximation. The reason why this above approach works is that, when we run the procedure of Lingas and Lundell at some arbitrary vertex s, it will always detect a short cycle if there is one passing *close* to s (but not necessarily passing through s itself). This nice property has been noticed and exploited for related algorithms on *unweighted*

graphs [10][2]. However, we think we are the first to prove such a property in the weighted case. We note that one of the two algorithms proposed by Roditty and Tov in [15] also satisfies such a property. We did not find a way to exploit their algorithm in order to improve our approximation factor.

Our approach for graphs with bounded integer edge-weights (see Sec. 3.2) is the cornerstone of all our other results in the paper. We considerably refine this approach so that it also applies to graphs with *arbitrary* non-negative edge-weights.

▶ **Theorem 2.** *For every $\varepsilon > 0$ and $G = (V, E, w)$ with non-negative edge-weights, we can compute a deterministic $(2+\varepsilon)$-approximation for* Girth *in time $\tilde{\mathcal{O}}(n^{5/3}\text{polylog}1/\varepsilon + m)$.*

We note that in [15], Roditty and Tov introduced a nice technique – that we partly reuse in this paper – in order to transpose their results for bounded integer-weights to arbitrary weights. However, we face several new difficulties, not encountered in [15], due to the need to perform all the intermediate operations in *subquadratic* time. As a side contribution of this work, we present an intricate modification of our approach for graphs with bounded integer edge-weights that we use in order to approximate the girth of graphs with arbitrary non-negative edge-weights up to a polynomial-factor. This subquadratic-time routine could be useful to anyone improving our result for the graphs with integer-weights in order to generalize their results to the graphs with non-negative real weights.

Our algorithms are subquadratic in the size of the graph, but they may be quadratic in its order n if there are $m = \Theta(n^2)$ edges. By a folklore application of Moore bounds, any unweighted graph with $\mathcal{O}(n^{1+\frac{1}{\ell}})$ edges has girth at most 2ℓ, and so, we can always output a constant upper-bound on the girth of moderately dense graphs. It implies that the dependency on m can always be removed in the running-time of approximation algorithms for the girth of *unweighted* graphs. However, in the full version of this paper, we prove by using elementary arguments that any approximation algorithm for the girth on weighted graphs must run in $\Omega(m)$-time. We study what happens if we have *sorted* adjacency lists[3].

▶ **Theorem 3.** *Let $G = (V, E, w)$ have sorted adjacency lists.*

1. *If all edge-weights are in $\{1, \ldots, M\}$ then, we can compute a deterministic 4-approximation for* Girth *in time $\tilde{\mathcal{O}}(n^{5/3}\text{polylog}M)$.*
2. *If all edge-weights are non-negative then, we can compute a* randomized *4-approximation for* Girth *in time $\tilde{\mathcal{O}}(n^{5/3})$. For every $\varepsilon > 0$, we can also compute a deterministic $(4+\varepsilon)$-approximation for* Girth *in time $\tilde{\mathcal{O}}(n^{5/3}\text{polylog}1/\varepsilon)$.*

We observe that even assuming sorted adjacency lists, it is not clear whether the algorithm of Theorem 1 can be implemented to run in time $\tilde{\mathcal{O}}(n^{5/3}\text{polylog}M)$. Indeed, this algorithm requires to build several induced subgraphs in time roughly proportional to their size, that requires a different preprocessing on the adjacency lists. We prove that we do not need to construct these induced subgraphs *entirely* in order to derive a constant-factor approximation of the girth. Similarly, for graphs with non-negative edge-weights we cannot use our polynomial-factor approximation algorithm for the girth directly, as it needs to enumerate all edges in the graph. We overcome this difficulty through the help of a classical density result for the C_4-free *unweighted* graphs [4].

[2] There is a subtle difference between our approach for weighted graphs and the one formerly applied to unweighted graphs. Indeed, we need to consider all edges in the *subgraphs* that are induced by some small balls in the graph, that might include some large-weight edges not on any shortest-path in G. For unweighted graphs [10, 16], they mostly consider edges on some shortest-paths in G between a pre-defined vertex and the other vertices in the ball.

[3] Throughout this paper, we call an adjacency list *sorted* if it is sorted by edge-weight, and *ordered* if it is sorted by neighbour index. See also Sec. 2.

1.2 Related work

Approximation algorithms for the girth. Itai and Rodeh were the first to study the Girth problem for *unweighted* graphs [9]. Among other results, they showed how to compute an additive +1-approximation of the girth in time $\mathcal{O}(n^2)$. This was later completed by Lingas and Lundell [10], who proposed a randomized quasi 2-approximation algorithm for this problem that runs in time $\mathcal{O}(n^{3/2}\sqrt{\log n})$. In [17], Roditty and Vassilevska Williams presented the first deterministic approximation algorithm for the girth of unweighted graphs. Specifically, they obtained a 2-approximation algorithm in time $\tilde{\mathcal{O}}(n^{5/3})$, and they conjectured that there does not exist any subquadratic $(2-\varepsilon)$-approximation for Girth. We obtain the same approximation factor for weighted graphs, and we almost match their running time up to polylog factors and to an additional term in $\tilde{O}(m)$. It would be interesting to know whether in our case, this dependency on m can be removed while preserving the approximation factor 2. Very recently, new subquadratic-time approximation algorithms were proposed for Girth in unweighted graphs (see [7]). It is open whether one can achieve a constant-factor approximation for the girth in, say, $\tilde{\mathcal{O}}(n^{1+o(1)})$-time.

Much less is known about the girth of weighted graphs. The first known subcubic approximation was the one of Lingas and Lundell [10], that only applies to graphs with bounded integer edge-weights. Their work somewhat generalizes the algorithm of Itai and Rodeh for unweighted graphs. The approximation factor was later improved to 4/3 by Roditty and Tov, still for the graphs with bounded integer weights, and to $4/3+\varepsilon$ for the graphs with *arbitrary weights* [15]. Our algorithms in this paper are faster than these two previous algorithms, but they use the latter as a routine to be applied on several subgraphs of sublinear size. Therefore, the approximation factors that we obtain cannot outperform those obtained in [10, 15].

More recently, a breakthrough logarithmic approximation of the girth of *directed* weighted graphs was obtained in [11].

Approximate distances. Finally, approximation algorithms for the girth are tightly related to the computation of approximate distances in weighted graphs. In a seminal paper [18], Thorup and Zwick showed that we can compute in expected time $\mathcal{O}(mn^{1/k})$ an approximate distance oracle: that can answer any distance query in time $\mathcal{O}(k)$ with a multiplicative stretch at most $2k-1$. This has been improved in several follow-ups [2, 5, 12, 14, 20]. However, the construction of most oracles already takes (super)quadratic time for moderately dense graphs. A key observation is that we do not need to construct these oracles *entirely* if we just want to approximate the girth. This allows us to avoid a great deal of distance computations, and so, to lower the running time.

1.3 Organization of the paper

We start gathering in Section 2 some known results from the literature that we will use for our algorithm. Then, in Section 3, we give some new insights on the algorithm of Lingas and Lundell [10] before presenting our main result (Theorem 1). Our algorithm is generalized to graphs with arbitrary weights in Section 4. Finally, we remove the dependency on the number of edges in the time complexity of our algorithms in Section 5. We conclude this paper with some open perspectives (Section 6). Due to space restrictions, some of the proofs are omitted. Full proofs can be found in our technical report [8].

2 Preliminaries

We refer to [3] for any undefined graph terminology. Graphs in this study are finite, simple (hence, without any loop nor multiple edges), connected and edge-weighted. Specifically, we denote a weighted graph by a triple $G = (V, E, w)$ where $w : E \to \mathbb{R}^+$ is the edge-weight function of G. The weight of a subgraph $H \subseteq G$, denoted $w(H) := \sum_{e \in E(H)} w_e$, is the sum of the weights on its edges. The girth of G is the minimum weight of a cycle in G. The distance $dist_G(u, v)$ between any two vertices $u, v \in V$ is the minimum weight of an uv-path in G. By extension, for every $v \in V$ and $S \subseteq V$ we define $dist_G(v, S) := \min_{u \in S} dist_G(u, v)$. – We will sometimes omit the subscript if no ambiguity on the graph G can occur. – For any $v \in V$ and $r \geq 0$, we also define the ball $B_G(v, r) := \{u \in V \mid dist_G(u, v) \leq r\}$. Finally, an r-nearest set for v is any r-set $\mathcal{N}_r(v)$ such that, for any $x \in \mathcal{N}_r(v)$ and $y \notin \mathcal{N}_r(v)$, we have $dist_G(v, x) \leq dist_G(v, y)$.

For every $v \in V$, let $N_G(v) = \{u \in V \mid uv \in E\}$ be the (open) neighbourhood of vertex v and let $d_v = |N_G(v)|$ be its degree. Let $Q_v = \{vu \mid u \in N_G(v)\}$ be totally ordered. We call it a *sorted* adjacency list if edges incident to v are ordered by increasing weight, *i.e.*, $Q_v = (vu_1, vu_2, \ldots, vu_{d_v})$ and $w_{vu_i} \leq w_{vu_{i+1}}$ for every $i < d_v$. However, we call it an *ordered* adjacency list if, given some fixed total ordering $\prec$ over V the neighbours of v are ordered according to $\prec$ (*i.e.*, $u_i \prec u_{i+1}$ for every $i < d_v$). Throughout the rest of the paper we will assume that each vertex has access to two copies of its adjacency list: one being sorted and the other being ordered. The latter can always be ensured up to an $\tilde{O}(m)$-time preprocessing.

2.1 The Hitting Set method

We gather many well-known facts in the literature, that can be found, *e.g.*, in [14, 18, 1, 6]. All these facts are combined in order to prove the following useful result for our algorithms:

▶ **Proposition 4.** *For any graph $G = (V, E, w)$ with* sorted *adjacency lists, in $\tilde{O}(n^{5/3})$-time we can compute a set $S \subseteq V$, and the open balls $B_S(v) := \{u \in V \mid dist(v, u) < dist(v, S)\}$ for every $v \in V$, such that the following two properties hold true:*

1. $|S| = \tilde{\mathcal{O}}(n^{2/3})$;
2. *and for every $v \in V$ we have $|B_S(v)| = \mathcal{O}(n^{1/3})$.*

It is well-known that a set S as requested by Proposition 4 can be constructed *randomly* as follows: every vertex in V is added in S with equal probability $n^{-1/3}$ [18]. This construction was derandomized in [14, 18, 1, 6]. In what follows we will not only need the balls $B_S(v)$ for every vertex v, but also the subgraphs these balls induce in G. Next, we observe that all these subgraphs can be obtained almost for free. Namely:

▶ **Lemma 5** (folklore). *For every $G = (V, E, w)$ and $U \subseteq V$ we can compute the subgraph $G[U]$ induced by U in time $\tilde{\mathcal{O}}(|U|^2)$ (assuming* ordered *adjacency lists).*

3 Case of graphs with bounded integer weights

This section is devoted to the proof of Theorem 1. We start presenting some new properties of a previous approximation algorithm for the girth of weighted graphs (Section 3.1) as we will need to use them in our own algorithm. Then, we prove our main result for graphs with bounded integer weights in Section 3.2.

3.1 Reporting a close short cycle

We propose a deeper analysis of an existing approximation algorithm for GIRTH on weighted graphs [10]. Roughly, this algorithm applies a same procedure to every vertex of the graph. In order to derive the approximation factor of their algorithm, the authors in [10] were considering a run that takes as entry some vertex *on a shortest cycle*. This is in contrast with the classical algorithm from Itai and Rodeh on unweighted graphs [9], that also offers provable guarantees on the length of the output assuming there is a shortest cycle passing *close* to the source (but not necessarily passing by this vertex); see [10, Lemma 2]. We revisit the analysis of the algorithm in [10] for weighted graphs, and we prove that this algorithm also satisfies such a "closeness property".

The HBD-algorithm from [10]. Given $G = (V, E, w)$, $s \in V$ and $t \geq 0$, the algorithm $HBD(G, s, t)$ is a relaxed version of Dijkstra's single-source shortest-path algorithm. We are only interested in computing the ball of radius t around s, and so, we stop if there are no more unvisited vertices at a distance $\leq t$ from s. Furthermore, whenever we visit a vertex $u \in B_G(s, t)$, we only relax edges $e = \{u, v\}$ such that $dist(s, u) + w_e \leq t$. Then, a cycle is detected if we already inferred that $dist(s, v) \leq t$ (*i.e.*, using another neighbour of v than u). Overall, the algorithm stops as soon as it encounters a cycle, or all the vertices in $B_G(s, t)$ were visited. Assuming sorted adjacency lists, this algorithm runs in $\tilde{\mathcal{O}}(n)$-time [10].

(a) `HBD`(G, s, t).

```
1: for all v ∈ V do
2:     d(v) ← ∞; π(v) ← NIL
3: d(s) ← 0; Q ← {s}
4: while Q ≠ ∅ do
5:     u ← Extract-min(Q)
6:     Controlled-Relax(u, t)
```

(b) `Controlled-Relax`(u, t).

```
1: Q_u ← sorted adj. list
2: uv ← Extract-min(Q_u)
3: while d(u) + w_uv ≤ t do
4:     RelaxOrStop(u, v)
5:     uv ← Extract-min(Q_u)
```

(c) `RelaxOrStop`(u, v).

```
1: if d(v) ≠ ∞ then
2:     return a cycle and stop
3: else
4:     d(v) ← d(u) + w_uv
5:     Q ← Q ∪ {v}
```

▶ **Lemma 6** ([10]). *If HBD(G, s, t) detects a cycle, then its weight is $\leq 2t$.*

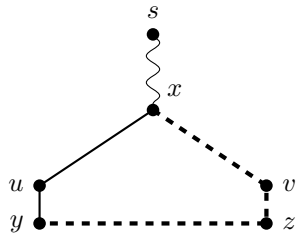
We now complete the analysis of the `HBD`-algorithm in order to derive a generalization of [10, Lemma 2] to weighted graphs. Assuming no cycle has been detected, we first gain more insights on the structure of the ball of radius t centered at s.

▶ **Lemma 7.** *If HBD(G, s, t) does not detect a cycle then, for any $v \in B_G(s, t)$, there exists a unique sv-path of weight $\leq t$.*

Based on Lemma 7, we state some bounds on the weight of the cycle detected using `HBD`. In particular, Corollary 9 will play a key role in the analysis of our algorithms.

▶ **Corollary 8.** *Given $G = (V, E, w)$, let $s \in V$ and let C be a cycle. The minimum t_0 such that HBD(G, s, t_0) detects a cycle satisfies $t_0 \leq dist(s, C) + w(C)$.*

▶ **Corollary 9.** *Given $G = (V, E, w)$, let $s \in V$ and let C be a cycle. Assume the existence of a vertex $x \in V(C)$ such that $\max_{v \in V(C)} dist_C(x, v) \geq dist_G(s, x) > 0$. Then, the minimum t_0 such that HBD(G, s, t_0) detects a cycle satisfies $t_0 \leq w(C)$.*



Proof. Let $B_x = \{v \in V(C) \mid dist_C(x,v) < dist_G(x,s)\}$. Since we assume that we have $\max_{v \in V(C)} dist_C(x,v) \geq dist_G(x,s)$, $B_x \neq V(C)$. Hence there exist $uy, vz \in E(C)$ distinct such that $u, v \in B_x$ but $y, z \notin B_x$. W.l.o.g. $dist_C(x,y) \leq dist_C(x,z)$. We can bipartition $E(C)$ in two edge-disjoint xy-paths P_1 and P_2, with P_1 being the xy-subpath passing by vz (and so, P_2 is the other xy-subpath passing by uy). Note that it implies $dist_C(x,y) = w(P_2) \leq w(C)/2$. Then, by Lemma 7 we have $t_0 \leq dist_G(s,x) + \max\{w(P_1), w(P_2)\} = dist_G(s,x) + w(P_1) \leq dist_C(x,y) + w(P_1) = w(P_2) + w(P_1) = w(C)$. ◀

3.2 Subquadratic-time approximation

Proof of Theorem 1. We analyse the following `Subquadratic-Approx` algorithm:

(a) `Approx-Girth`(G, s, M).

1: Find the minimum $t \in [3; M \cdot |V(G)|]$ such that: `HBD`(G, s, t) detects a cycle.
2: Let C_s be the shortest cycle we so computed.
3: **return** C_s.

(b) `Subquadratic-Approx`(G, M).

1: Let S and $(B_S(v))_{v \in V}$ be as in Prop. 4.
2: **for all** $s \in S$ **do**
3: $\quad C_s \leftarrow$ `Approx-Girth`(G, s, M)
4:
5: **for all** $v \notin S$ **do**
6: $\quad$ Let G'_v be induced by $B_S(v)$.
7: $\quad C_v \leftarrow$ `Approx-Girth`(G'_v, v, M)
8:
9: **return** a shortest cycle in $\{C_v \mid v \in V\}$.

The algorithm starts precomputing a set $S \subseteq V$ and the open balls $(B_S(v))_{v \in V}$ as described in Proposition 4. This takes time $\tilde{\mathcal{O}}(n^{5/3})$, plus an additional preprocessing time in $\tilde{O}(m)$ for sorting the adjacency lists. Then, we process the vertices in S and those in $V \setminus S$ separately: For every $s \in S$, we compute the smallest $t_s \in [3; Mn]$ such that `HBD`(G, s, t_s) detects a cycle by using a dichotomic search (procedure `Approx-Girth`(G, s, M)). We store the cycle C_s outputted by `HBD`(G, s, t_s). Since each test we perform during the dichotomic search consists in a call to the `HBD`-algorithm, this takes time $\tilde{\mathcal{O}}(n \log M)$ per vertex in S, and so, $\tilde{\mathcal{O}}(n|S| \log M) = \tilde{\mathcal{O}}(n^{5/3} \log M)$ in total. We now consider the vertices $v \in V \setminus S$ sequentially. Let G'_v be the subgraph of G induced by the open ball $B_S(v)$. By Lemma 5, this subgraph can be computed in time $\tilde{\mathcal{O}}(|B_S(v)|^2) = \tilde{\mathcal{O}}(n^{2/3})$ – assuming a preprocessing of the graph in time $\mathcal{O}(m)$ for ordering the adjacency lists. We apply the same procedure as for the vertices in S but, we restrict ourselves to the ball $B_S(v)$. That is, we call `Approx-Girth`(G'_v, v, M), and we denote by C_v the cycle outputted by this algorithm. Since we restrict ourselves to a subgraph of order $\mathcal{O}(n^{1/3})$, this takes total time $\tilde{\mathcal{O}}(n \cdot (n^{2/3} + n^{1/3} \log M)) = \tilde{\mathcal{O}}(n^{5/3} \log M)$.

Let $C \in \{C_v \mid v \in V\}$ be of minimum weight. We claim that $w(C)$ is a 2-approximation of the girth of G, that will end proving the theorem. In order to prove this claim, we apply the following case analysis to some arbitrary shortest cycle C_0 of G. If $V(C_0) \cap S \neq \emptyset$ then, let C_S be a shortest cycle among $\{C_s \mid s \in S\}$. We prove as a subclaim that $w(C_S)$ is at most twice the weight of a shortest cycle intersecting S. In order to prove this subclaim,

it suffices to prove that for every $s \in S$, we compute a cycle C_s of weight no more than twice the weight of a shortest cycle passing by s. By Corollary 8, if t_s is the smallest t such that $HBD(G, s, t)$ detects a cycle then, a shortest cycle C passing by s must have weight $\geq t_s$. Furthermore, by Lemma 6 we get $w(C_s) \leq 2t_s$, thereby proving the subclaim. Thus, $w(C_S) \leq 2w(C_0)$ if $V(C_0) \cap S \neq \emptyset$. From now on we assume $V(C_0) \cap S = \emptyset$. Let $v \in V(C_0)$ be arbitrary. There are two subcases:

- If $V(C_0) \subseteq B_S(v)$ then, C_0 is also a cycle in G'_v. Moreover by Corollary 8 applied for $dist(v, C_0) = 0$, the smallest t_v such that $\texttt{HBD}(G'_v, v, t_v)$ detects a cycle satisfies $t_v \leq w(C_0)$. By Lemma 6, $w(C) \leq w(C_v) \leq 2w(C_0)$.
- Otherwise $V(C_0) \not\subseteq B_S(v)$. This implies that we have: $\max_{u \in V(C_0)} dist_{C_0}(u, v) \geq \max_{u \in V(C_0)} dist_G(u, v) \geq dist_G(v, S) > 0$. Furthermore, let $s \in S$ minimize $dist_G(s, v)$. Then, by Corollary 9, the smallest t_s such that $\texttt{HBD}(G, s, t_s)$ detects a cycle satisfies $t_s \leq w(C_0)$. As a result, by Lemma 6 $w(C) \leq w(C_s) \leq 2w(C_0)$.

Summarizing, $w(C) \leq 2w(C_0)$ in all the cases. ◀

4 Generalization to unbounded weights

This section is devoted to the proof of Theorem 2. We divide it into two parts. In Section 4.1 we present a *polynomial-factor* approximation of the girth in subquadratic time. This part is new compared to [15] and the techniques used are interesting in their own right. Then, based on a clever technique from [15], we end up refining this rough estimate of the girth until we obtain a constant-factor approximation (Section 4.2).

Throughout this section, we will use the main result of Roditty and Tov as a subroutine:

▶ **Theorem 10** ([15]). *For every $G = (V, E, w)$ with arbitrary non-negative edge-weights, we can compute a $(4/3 + \varepsilon)$-approximation for* Girth *in time $\tilde{\mathcal{O}}(n^2/\varepsilon)$.*

4.1 A polynomial-factor approximation

For simplicity, we first reduce the general case of graphs with non-negative weights to the subcase of graphs with positive weights. We omit the proof as it quite similar, but simpler, to the one of Proposition 12 (presented next).

▶ **Lemma 11.** *Assume there exists an $T(n, m)$-time α-approximation algorithm for* Girth *for graphs with positive edge-weights, where $T(n, m) = \Omega(m)$. Then, there also exists an $\mathcal{O}(T(n, m))$-time α-approximation algorithm for* Girth *for graphs with non-negative edge-weights.*

We now obtain an approximation of the girth that only depends on the order of the graph. We stress that for weighted graphs, this is already a non-trivial task.

▶ **Proposition 12.** *For every $G = (V, E, w)$ with arbitrary positive edge-weights, we can compute an $\tilde{\mathcal{O}}(n^{2/3})$-approximation for* Girth *in time $\tilde{\mathcal{O}}(n^{5/3} + m)$.*

Proof. Let S be as in Proposition 4. We show a significantly more elaborate method that uses S in order to approximate the girth. We divide this method into five main steps.

Step 1: check the small balls. For every $v \notin S$, let G'_v be the subgraph induced by the open ball $B_S(v)$. As before, we first estimate the girth of G'_v. Since this subgraph has order $\mathcal{O}(n^{1/3})$, by Theorem 10 we can compute a constant-factor approximation for its girth in time $\tilde{\mathcal{O}}(n^{2/3})$ (say, a 2-approximation). Overall, this step takes total time $\tilde{\mathcal{O}}(n^{5/3})$. Furthermore, after completing this step the following property (also used in Theorem 1) becomes true:

▷ Claim 13. Let C_v be a shortest cycle passing through v. If $w(C_v) < 2 \cdot dist(v, S)$ then, we computed a cycle of weight $\leq 2w(C_v)$.

Step 2: partitioning into (shortest path) subtrees. Intuitively, what we try to do next is to approximate the weight of a shortest cycle passing close to S. The difference with Theorem 1 is that we cannot use directly the algorithm of Roditty and Tov for that. Indeed, their algorithm has some global steps (*e.g.*, the approximate computation of the girth of some sparse spanner) that we currently do not know how to do in subquadratic time. So, we need to find some new techniques. Specifically, we partition the vertex-set V into shortest-path subtrees $(T_s)_{s \in S}$ such that, for every $s \in S$ and $v \in V(T_s)$ we have $dist(v, s) = dist(v, S)$. As noted, *e.g.*, in [18], a simple way to do that is to add a dummy vertex $x_S \notin V$, edges sx_S for every $s \in S$ with weight 0, then to compute a shortest-path tree rooted at x_S in time $\tilde{\mathcal{O}}(m)$. See Fig. 3 for an example. In what follows, we show how to use this tree structure in order to compute short cycles.

Figure 3 An example of Step 2. The two vertices in S are drawn as rectangles.

Step 3: finding short cycles in a subtree. Let $s \in S$ be fixed. Informally, we try to estimate the weight of a shortest cycle in $V(T_s)$. Note that every such a cycle has an edge that is not contained in T_s. So, we consider all the edges $e = uv$ such that $u, v \in V(T_s)$ but $e \notin E(T_s)$. Adding this edge in T_s closes a cycle. Let $C_{e,s}$ be an (unknown) shortest cycle passing by e and contained in $V(T_s)$. We output $dist(s, u) + w_e + dist(v, s)$ as a rough estimate of $w(C_{e,s})$. Indeed, the latter is a straightforward upper-bound on $w(C_{e,s})$, and this bound is reached if $s \in \{u, v\}$. Overall, this step takes total time $\mathcal{O}(m)$.

▷ Claim 14. Let C^*_s be a shortest cycle contained in $V(T_s)$. After Steps 1-3, we computed a cycle of weight $\leq 2w(C^*_s)$.

Step 4: finding short cycles in two subtrees. We now want to estimate the weight of a shortest cycle in $V(T_s) \cup V(T_{s'})$, for some distinct $s, s' \in S$. We only need to consider the case where this cycle must contain two edges e, e' with an end in $V(T_s)$ and the other end in $V(T_{s'})$ (all other cases have already been considered at Step 3).

1. We scan all the edges $e = uv \in E$ such that u and v are not in a same subtree. Let $s_u, s_v \in S$ such that $u \in V(T_{s_u}), v \in V(T_{s_v})$. We set $\ell(e) = dist(s_u, u) + w_e + dist(v, s_v)$.
2. Group all these above edges with their two ends in the same two subtrees. It takes time $\mathcal{O}(m + |S|) = \mathcal{O}(m + n^{2/3})$ by using, say, a linear-time sorting algorithm.
3. Finally, for every distinct $s, s' \in S$, let $E(s, s')$ contain all the edges with one end in T_s and the other end in $T_{s'}$. If $|E(s, s')| \geq 2$ then, we pick e, e' minimizing $\ell(\cdot)$ and we output $\ell(e) + \ell(e')$. Overall, since the sets $E(s, s')$ partition the edges of G, this last phase also takes time $\mathcal{O}(m)$.

▷ Claim 15. Let $s, s' \in S$ be distinct and let $C^*_{s,s'}$ be a shortest cycle contained in $V(T_s) \cup V(T_{s'})$. After Steps 1-4, we computed a cycle of weight $\leq 3w(C^*_{s,s'})$.

Step 5: the general case. We end up defining a weighted graph $H_S = (S, E_S, w^S)$, where $E_S = \{ss' \mid E(s,s') \neq \emptyset\}$, and for every $ss' \in E_S$:

$$w^S_{ss'} = \min_{e \in E(s,s')} \ell(e) = \min_{uv \in E(s,s')} dist(s,u) + w_{uv} + dist(v,s').$$

Roughly, $w^S_{ss'}$ is the smallest weight of an ss'-path with one edge in $E(s,s')$. We can construct H_S simply by scanning all the sets $E(s,s')$ (computed during Step 4). Overall, since the sets $E(s,s')$ partition the edges of G, this takes total time $\mathcal{O}(m+|S|) = \mathcal{O}(m+n^{2/3})$. Furthermore, by Theorem 10 we can compute a constant-factor approximation of the girth of H_S in time $\tilde{\mathcal{O}}(|S|^2) = \tilde{\mathcal{O}}(n^{4/3})$. The graph H_S is *not* a subgraph of G. However, given a cycle C_H for H_S, we can compute a cycle C^*_H of G as follows. For every $s \in V(C_H)$ let $s', s'' \in V(C_H)$ be its two neighbours. By construction, there exist $e = uv \in E(s',s)$ and $e' = xy \in E(s,s'')$ such that the edges ss' and ss'' in H_S have weights $dist(s',u) + w_e + dist(v,s)$ and $dist(s,x) + w_{e'} + dist(y,s'')$, respectively. – We may assume the edges e, e' to be stored in H_S so that s', s'' will choose the same common edge with s. – Then, we replace s by the vx-path in T_s. It is important to notice that, by construction, we have $w(C^*_H) \leq w(C_H)$. In particular, we can apply this above transformation to the (approximately shortest) cycle of H_S that has been outputted by the algorithm of Roditty and Tov (Theorem 10).

Overall, let $C_{\min}$ be a shortest cycle computed by the algorithm above (*i.e.*, after Steps 1-5). In order to finish the proof, we need to show that $w(C_{\min})$ is an $\tilde{\mathcal{O}}(n^{2/3})$-approximation of the girth of G. By Claims 14 and 15, this is the case if there exists a shortest cycle intersecting at most two subtrees $T_s, s \in S$. From now on assume that any shortest cycle C_0 of G intersects at least three subtrees T_s. Write $C_0 = (v_0, v_1, \ldots, v_{p-1}, v_0)$ and assume w.l.o.g. v_0, v_{p-1} are not contained into the same subtree T_s. We partition the v_i's into the maximal subpaths $P_0, P_1, \ldots, P_{q-1},\ q \leq p$ that are contained into the vertex-set of a same subtree T_s (in particular, $v_0 \in V(P_0)$ and $v_{p-1} \in V(P_{q-1})$). Furthermore for every $j \in \{0,1,\ldots,q-1\}$ let $s_j \in S$ be such $V(P_j) \subseteq V(T_{s_j})$, and let i_j be the largest index such that $v_{i_j} \in V(P_j)$. For instance, $i_{q-1} = p-1$ by construction. Since $P_0 = P_q$ and $q \geq 3$ by the hypothesis, there exist distinct indices j_1, j_2 such that $s_{j_1} = s_{j_2+1}$ and for every $j \in \{j_1, j_1+1, \ldots, j_2\}$ the s_j's are pairwise different (indices are taken modulo q). Then, two cases may arise:

- Case $j_2 = j_1 + 1$. We have: $e_{j_1} := v_{i_{j_1}} v_{i_{j_1}+1},\ e_{j_2} := v_{i_{j_2}+1} v_{i_{j_2}} \in E(s_{j_1}, s_{j_2})$. Furthermore, C_0 goes by $v_{i_{j_1}}$ (by $v_{i_{j_1}+1}, v_{i_{j_2}+1}, v_{i_{j_2}}$, respectively), and so, by Claim 13, either we computed a short cycle of weight $\leq 2w(C_0)$ during Step 1, or we have $w(C_0) \geq 2 \cdot \max\{dist(s_{j_1}, v_{i_{j_1}}), dist(s_{j_1}, v_{i_{j_2}+1}), dist(s_{j_2}, v_{i_{j_1}+1}), dist(s_{j_2}, v_{i_{j_2}})\}$. In the latter case, there exists a cycle of weight: $\leq dist(s_{j_1}, v_{i_{j_1}}) + w_{e_{j_1}} + dist(s_{j_2}, v_{i_{j_1}+1}) + dist(s_{j_2}, v_{i_{j_2}}) + w_{e_{j_2}} + dist(s_{j_1}, v_{i_{j_2}+1}) \leq 3w(C_0)$ that is fully contained in $V(T_{s_{j_1}}) \cup V(T_{s_{j_2}})$. By Claim 15, we so computed a cycle of weight $\leq 9w(C_0)$ at Step 4.

- From now on let us assume $j_2 \neq j_1 + 1$. For every j we have $e_j := v_{i_j} v_{i_j+1} \in E(s_j, s_{j+1})$, and so, the edge $s_j s_{j+1} \in E_S$ has weight no more than $dist(s_j, v_{i_j}) + w_{e_j} + dist(v_{i_j+1}, s_{j+1})$ in H_S (indices are taken modulo q for the s_j's and modulo p for the v_i's). Furthermore, C_0 goes by v_{i_j} (by v_{i_j+1}, respectively), and so, by Claim 13, either we computed a short cycle of weight $\leq 2w(C_0)$ during Step 1, or we have $w(C_0) \geq 2 \cdot \max\{dist(s_j, v_{i_j}), dist(s_{j+1}, v_{i_j+1}\}$ for every j. In the latter case,

$(s_{j_1}, s_{j_1+1}, \ldots, s_{j_2}, s_{j_2+1} = s_{j_1})$ is a cycle in H_S of weight:

$$\leq w(C_0) + \sum_{j=j_1}^{j_2} (dist(v_{i_{j-1}+1}, s_j) + dist(s_j, v_{i_j}))$$
$$\leq w(C_0)(1 + (j_2 - j_1 + 1))$$
$$\leq w(C_0)|S| = \tilde{\mathcal{O}}(n^{2/3} \cdot w(C_0)).$$

Then, let C_H be a cycle of H_S such that $w(C_H) = \tilde{\mathcal{O}}(n^{2/3} \cdot w(C_0))$ (obtained by applying the algorithm of Roditty and Tov to H_S). As explained above, we can derive from C_H a cycle C^*_H of G such that $w(C^*_H) \leq w(C_H) = \tilde{\mathcal{O}}(n^{2/3} \cdot w(C_0))$.

Summarizing, we obtain an $\tilde{\mathcal{O}}(n^{2/3})$-approximation of the girth by outputting a shortest cycle computed during Steps 1,3,4,5. ◀

4.2 Improving the approximation factor

Sketch Proof of Theorem 2. We may assume that all weights are positive by Lemma 11. Let g^* be the $\tilde{\mathcal{O}}(n^{2/3})$-approximation that we computed by using Proposition 12. There exists some (known) constant c such that the girth of G is somewhere between $g^*/(cn^{2/3}\log n)$ and g^*. Then, let $i_{\min}, i_{\max}$ be the smallest nonnegative integers such that $g^*/(cn^{2/3}\log n) \leq (1+\varepsilon/2)^{i_{\min}}$ and in the same way $g^* \leq (1+\varepsilon/2)^{i_{\max}}$. We have that:

$$i_{\max} - i_{\min} = \mathcal{O}\left(\log_{1+\varepsilon/2}\left(\frac{g^*}{g^*/(cn^{2/3}\log n)}\right)\right) = \mathcal{O}(\log n/\log(1+\varepsilon/2)) = \mathcal{O}(\log n/\varepsilon).$$

Let S be as in Proposition 4. For every $v \in V \setminus S$, we compute a 2-approximation of a shortest cycle in G'_v: the subgraph of G induced by the ball $B_S(v)$[4]. By Theorem 10, it can be done in time $\tilde{\mathcal{O}}(n^{2/3})$ for each v, and so, this takes total time $\tilde{\mathcal{O}}(n^{5/3})$. Then, let $T = \{(1+\varepsilon/2)^i \mid i_{\min} \leq i \leq i_{\max}\}$. For every $s \in S$, we compute the smallest $t \in T$ such that $HBD(G, s, t)$ detects a cycle (if any). It can be done in time $\tilde{\mathcal{O}}(|S|n\log|T|) = \tilde{\mathcal{O}}(n^{5/3}\log 1/\varepsilon)$ by using a dichotomic search. Finally, let $g_{\min}$ be the value computed by the above algorithm (with a corresponding cycle). In order to conclude we prove, with a similar case analysis as for Theorem 1, that the girth of G is at least $g_{\min}/(2+\varepsilon)$. ◀

5 A subquadratic algorithm for dense graphs

A drawback of the algorithms in Theorems 1 and 2 is that their time complexity also depends on the number m of edges. It implies that for dense graphs with $m = \Theta(n^2)$ edges we do not achieve any improvement on the running time compared to [10, 15]. The main result of this section is that assuming sorted adjacency lists, the dependency on m can always be discarded (Theorem 3). Due to lack of space, we will only prove the following weaker result:

▶ **Proposition 16.** *For every $G = (V, E, w)$ with non-negative edge-weights and sorted adjacency lists, we can compute:*

1. *a randomized 4-approximation for* Girth *in expected time $\tilde{\mathcal{O}}(n^{5/3})$;*
2. *and, for every $\varepsilon > 0$, a deterministic $(8+\varepsilon)$-approximation for* Girth *in time $\tilde{\mathcal{O}}(n^{5/3}\text{polylog}1/\varepsilon)$.*

[4] In fact, this is already done in the proof of Proposition 12, but we restate it here for completeness of the method.

Roughly, we can prove Theorem 3 by combining this above Proposition 16 with a natural modification of the algorithm presented in Section 3.2.

We will need the following well-known result in graph theory:

▶ **Theorem 17** ([4]). *Every* unweighted *graph with order n and $m \geq \left(\frac{1}{2} + o(1)\right) n^{3/2}$ edges contains a cycle of length four.*

Proof of Proposition 16. If G has $m = \tilde{\mathcal{O}}(n^{5/3})$ edges then, we can simply apply Theorem 2 for $\varepsilon = 2$ (the latter can be easily verified by scanning the adjacency lists until we read the end of it or we reach the desired upper-bound). From now on assume this is not the case and let H be induced by the $\left\lceil \left(\frac{1}{2} + o(1)\right) n^{3/2} \right\rceil$ edges of minimum weight in G.

We claim that H can be constructed in time $\tilde{\mathcal{O}}(n^{3/2})$ by using a priority queue Q. Indeed, initially we set $E(H) = \emptyset$ and for every $v \in V$ we start inserting in Q the edge of minimum-weight that is incident to v. This way, we ensure that a minimum-weight edge of $G \setminus E(H)$ is present in Q (recall that initially, $E(H) = \emptyset$, and so, $G = G \setminus E(H)$). Then, in order to preserve this above invariant, each time a minimum-weight edge uv is extracted from Q and added in H we insert in Q the remaining edge of minimum weight in Q_u and the one in Q_v (if any). – Note that in doing so, a same edge can be added in Q twice, but this has no consequence on the algorithm. –

We now apply Theorem 2 for $\varepsilon' = \varepsilon/4$ to H, and we so obtain a cycle C that is a $(2+\varepsilon/4)$-approximation of the girth of H. We claim that $w(C)$ is also a $(8+\varepsilon)$-approximation of the girth of G. In order to prove this claim, we need to consider two different cases:

- Assume there exists a shortest cycle C_0 of G such that $E(C_0) \subseteq E(H)$. By Theorem 2, $w(C) \leq (2 + \varepsilon/4)w(C_0) < (8 + \varepsilon)w(C_0)$.
- Otherwise, any shortest cycle C_0 of G has at least one edge that is not contained in H. Since edges are added by increasing weights, this implies that every shortest cycle contains an edge of weight at least $w_{\max}$, where $w_{\max}$ denotes the maximum-weight of an edge in H. In particular, the girth of G is at least $w_{\max}$. Furthermore, since H has enough edges by construction, by Theorem 17 it contains a cycle of four vertices; the latter has weight at most $4w_{\max}$. As a result, $w(C) \leq (2 + \varepsilon/4) \cdot 4w_{\max} = (8 + \varepsilon)w_{\max} \leq (8 + \varepsilon)w(C_0)$.

The above proves the claim, and so, the deterministic version of the result. In order to obtain a randomized 4-approximation, it suffices to pick $\varepsilon \leq 2$ and to output any cycle C' of H with four vertices (then, we output any of C, C' that has minimum weight). Up to some constant multiplicative increase of the number of edges to add in H, this can be done by using a randomized algorithm of Yuster and Zwick that runs in expected linear time [21, Theorem 2.9]. Note that this is the only source of randomness in the algorithm. ◀

We recall that any *unweighted* graph with $\mathcal{O}(n^{1+\frac{1}{\ell}})$ edges contains a cycle of length at most 2ℓ. We could use this density result instead of Theorem 17. In doing so, we could use a much sparser subgraph H in the proof of Proposition 16. However, our algorithm would still run in time $\tilde{\mathcal{O}}(n^{5/3})$ because the bottleneck is our call to the algorithm of Theorem 2.

6 Open problems

The most pressing question is whether we can achieve a 4/3-approximation for the girth in subquadratic time. If it is not the case then, what is the best approximation factor we can get in subquadratic time? We note that in [17], Roditty and Vassilevska Williams conjectured that we cannot achieve a $(2 - \varepsilon)$-approximation already for unweighted graphs[5].

[5] They did obtain such an algorithm for *triangle-free* graphs.

If their conjecture is true then, this would imply our algorithm is essentially optimal (at least for the non-dense graphs with $\mathcal{O}(n^{2-\varepsilon})$ edges). However, for the dense graphs with sorted adjacency lists, we left open whether a better approximation-factor than 4 can be obtained in $o(n^2)$-time. Finally, another interesting question is whether a constant-approximation for the girth can be computed in quasi linear time. We recall that this is wide open even for *unweighted* graphs [7].

References

1. D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28(4):1167–1181, 1999.
2. S. Baswana and S. Sen. Approximate distance oracles for unweighted graphs in expected O (n 2) time. *ACM Transactions on Algorithms (TALG)*, 2(4):557–577, 2006.
3. J. A. Bondy and U. S. R. Murty. *Graph theory*. Grad. Texts in Math., 2008.
4. W. Brown. On graphs that do not contain a Thomsen graph. *Canadian Mathematical Bulletin*, 9(2):1–2, 1966.
5. S. Chechik. Approximate distance oracles with constant query time. In *ACM STOC'14*, pages 654–663, 2014.
6. S. Chechik, D. Larkin, L. Roditty, G. Schoenebeck, R. Tarjan, and V. Vassilevska Williams. Better approximation algorithms for the graph diameter. In *SODA*, pages 1041–1052. SIAM, 2014.
7. S. Dahlgaard, M. Knudsen, and M. Stöckel. New Subquadratic Approximation Algorithms for the Girth. Technical report, arXiv, 2017. `arXiv:1704.02178`.
8. G. Ducoffe. Faster approximation algorithms for computing shortest cycles on weighted graphs. Technical report, arXiv, 2018. `arXiv:1810.10229`.
9. A. Itai and M. Rodeh. Finding a minimum circuit in a graph. *SIAM Journal on Computing*, 7(4):413–423, 1978.
10. A. Lingas and E. Lundell. Efficient approximation algorithms for shortest cycles in undirected graphs. *Information Processing Letters*, 109(10):493–498, 2009.
11. J. Pachocki, L. Roditty, A. Sidford, R. Tov, and V. Vassilevska Williams. Approximating cycles in directed graphs: Fast algorithms for girth and roundtrip spanners. In *SODA*, pages 1374–1392. SIAM, 2018.
12. M. Patrascu and L. Roditty. Distance Oracles beyond the Thorup–Zwick Bound. *SIAM Journal on Computing*, 43(1):300–311, 2014.
13. S. Robinson. Toward an optimal algorithm for matrix multiplication. *SIAM news*, 38(9):1–3, 2005.
14. L. Roditty, M. Thorup, and U. Zwick. Deterministic constructions of approximate distance oracles and spanners. In *ICALP*, pages 261–272. Springer, 2005.
15. L. Roditty and R. Tov. Approximating the girth. *ACM Transactions on Algorithms (TALG)*, 9(2):15, 2013.
16. L. Roditty and V. Vassilevska Williams. Minimum weight cycles and triangles: Equivalences and algorithms. In *FOCS*, pages 180–189. IEEE, 2011.
17. L. Roditty and V. Vassilevska Williams. Subquadratic time approximation algorithms for the girth. In *SODA*, pages 833–845. SIAM, 2012.
18. M. Thorup and U. Zwick. Approximate distance oracles. *Journal of the ACM (JACM)*, 52(1):1–24, 2005.
19. V. Vassilevska Williams and R. Williams. Subcubic equivalences between path, matrix and triangle problems. In *FOCS*, pages 645–654. IEEE, 2010.
20. C. Wulff-Nilsen. Approximate distance oracles with improved preprocessing time. In *ACM/SIAM SODA'12*, pages 202–208, 2012.
21. R. Yuster and U. Zwick. Finding even cycles even faster. *SIAM Journal on Discrete Mathematics*, 10(2):209–222, 1997.

Algorithmically Efficient Syntactic Characterization of Possibility Domains

Josep Díaz
Computer Science Department, Universitat Politècnica de Catalunya, Barcelona
diaz@cs.upc.edu

Lefteris Kirousis
Department of Mathematics, National and Kapodistrian University of Athens
Computer Science Department, Universitat Politècnica de Catalunya, Barcelona
lkirousis@math.uoa.gr

Sofia Kokonezi
Department of Mathematics, National and Kapodistrian University of Athens
skoko@math.uoa.gr

John Livieratos
Department of Mathematics, National and Kapodistrian University of Athens
jlivier89@math.uoa.gr

Abstract

We call *domain* any arbitrary subset of a Cartesian power of the set $\{0, 1\}$ when we think of it as reflecting abstract rationality restrictions on vectors of two-valued judgments on a number of issues. In Computational Social Choice Theory, and in particular in the theory of judgment aggregation, a domain is called a possibility domain if it admits a non-dictatorial aggregator, i.e. if for some k there exists a unanimous (idempotent) function $F : D^k \to D$ which is not a projection function. We prove that a domain is a possibility domain if and only if there is a propositional formula of a certain syntactic form, sometimes called an integrity constraint, whose set of satisfying truth assignments, or models, comprise the domain. We call *possibility* integrity constraints the formulas of the specific syntactic type we define. Given a possibility domain D, we show how to construct a possibility integrity constraint for D efficiently, i.e, in polynomial time in the size of the domain. We also show how to distinguish formulas that are possibility integrity constraints in linear time in the size of the input formula. Finally, we prove the analogous results for local possibility domains, i.e. domains that admit an aggregator which is not a projection function, even when restricted to any given issue. Our result falls in the realm of classical results that give syntactic characterizations of logical relations that have certain closure properties, like e.g. the result that logical relations component-wise closed under logical AND are precisely the models of Horn formulas. However, our techniques draw from results in judgment aggregation theory as well from results about propositional formulas and logical relations.

2012 ACM Subject Classification Theory of computation → Theory and algorithms for application domains

Keywords and phrases collective decision making, computational social choice, judgment aggregation, logical relations, algorithm complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.50

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at [3], `https://arxiv.org/abs/1901.00138`.

Funding *Josep Díaz*: Research partially supported by TIN2017-86727-C2-1-R, GRAMM.
Lefteris Kirousis: Research carried out while visiting the Computer Science Department of the Universitat Politècnica de Catalunya and supported by TIN2017-86727-C2-1-R, GRAMM.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 50; pp. 50:1–50:13

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

Acknowledgements We are grateful to Bruno Zanuttini for his comments that improved the presentation and simplified several proofs. Lefteris Kirousis is grateful to Phokion Kolaitis for initiating him to the area of Computational Social Choice Theory. We thank Eirini Georgoulaki for her valuable help in the final stages of writing this paper.

1 Introduction

We call *domain* any arbitrary subset of a Cartesian power $\{0,1\}^n$ ($n \geq 1$) when we think of it as the set of yes/no ballots, or accept/reject judgment vectors on n issues that are "rational" in the sense manifested by being a member of the subset. A domain D has a non-dictatorial aggregator if for some $k \geq 1$ there is a unanimous (idempotent) function $F : D^k \to D$ that is not a projection function. Such domains are called *possibility domains*. The theory of judgment aggregation was put in this abstract framework by Wilson [19], and then elaborated by several others (see e.g. the work by Dietrich [4] and Dokow and Holzman [6, 5]). It can be trivially shown that non-dictatorial aggregators always exist unless we demand that F is defined on an issue by issue fashion (see next section for formal definitions). Such aggregators are called Independent of Irrelevant Alternatives (IIA). In this work aggregators are assumed to be IIA.

It is a well known fact from elementary Propositional Logic that for every subset D of $\{0,1\}^n$, $n \geq 1$, i.e. for every domain, there is a Boolean formula in Conjunctive Normal Form (CNF) whose set of satisfying truth assignments, or models, denoted by $\mathrm{Mod}(\phi)$, is equal to D (see e.g. Enderton [8, Theorem 15B]). Zanuttini and Hébrard [21] give an algorithm that finds such a formula and runs in polynomial-time with respect to the size of the representation of D as input. Following Grandi and Endriss [11], we call such a ϕ an *integrity constraint* and think of it as expressing the "rationality" of D (the term comes from databases, see e.g. [7]).

We prove that a domain is a possibility domain, if and only if it admits an integrity constraint of a certain syntactic form to be precisely defined, which we call a *possibility* integrity constraint. Very roughly, possibility integrity constraints are formulas that belong to one of three types, the first two of which correspond to "easy" cases of possibility domains: (i) formulas whose variables can be partitioned into two non-empty subsets so that no clause contains variables from both sets and (ii) formulas whose clauses are exclusive OR's of their literals. The most interesting third type is comprised of formulas such that if we change the logical sign of some of their variables, we get formulas that have a Horn part and whose remaining clauses contain only negative occurrences of the variables in the Horn part. We call such formulas *renamable partially Horn*, whereas we call *partially Horn*[1] the formulas that belong to the third type without having to rename any variables. Furthermore, we show that the unified framework of Zanuttini and Hébrard [21] for producing formulas of a specific type that describe a given domain, and which entails the notion of prime formulas (i.e. formulas that we cannot further simplify its clauses; see Definition 2.11) works also in the case of possibility integrity constraints. Actually, in addition to the syntactical characterization of possibility domains, we give two algorithms: the first on input a formula decides whether it is a possibility integrity constraint in time linear in the length of the formula (notice that the definition of possibility integrity constraint entails searching over all subsets of variables of the formula); the second on input a domain D halts in time polynomial in the size of

[1] A weaker notion of Horn formulas has appeared before in the work of Yamasaki and Doshita [20]; however our notion is incomparable with theirs, in the sense that the class of partially Horn formulas in neither a subset nor a superset (nor equal) to the class $\mathbb{S}_0$ they define.

D and either decides that D is not a possibility domain or otherwise returns a possibility integrity constraint that describes D. It should be noted that the satisfiability problem remains NP-complete even when restricted to formulas that are partially Horn. However in Computational Social Choice, domains are considered to be non-empty (see paragraph preceding Example 2.6).

We then consider *local possibility domains*, that is, domains admitting IIA aggregators whose components are all different than any projection function. Such aggregators are called *locally non-dictatorial* (see [15]). Local non-dictatorial domains were introduced in [12] as *uniform possibility domains* (the definition entails also non-Boolean domains). We show that local possibility domains are described by formulas we call *local possibility integrity constraints* and again, we provide a linear algorithm that checks if a formula is a local possibility integrity constraint and a polynomial algorithm that checks if a domain is a local possibility one and, in case it is, constructs a local possibility integrity constraint that describes it.

As examples of similar classical results in the theory of Boolean relations, we mention that domains component-wise closed under $\wedge$ or $\vee$ have been identified with the class of domains that are models of Horn or dual-Horn formulas respectively (see Dechter and Pearl [1]). Also it is known that a domain is component-wise closed under the ternary sum mod 2 if and only if it is the set of models of a formula that is a conjunction of subformulas each of which is an exclusive OR (the term "ternary" refers to the number of bits to be summed). Finally, a domain is closed under the ternary majority operator if and only it is the set of models of a CNF formula where each clause has at most two literals. The latter two results are due to Schaefer [18]. The ternary majority operator is the ternary Boolean function that returns 1 on input three bits if and only if at least two of them are 1. It is also known that the respective formulas for each case can be found in polynomial time with respect to the size of D (see Zanuttini and Hébrard [21]).

Our result can be interpreted as verifying that non-dictatorial voting schemes can always be generated by integrity constraints that have a specific, easily recognizable syntactic form. This can prove valuable for applications in the field of judgment aggregation, where relations are frequently encountered in compact form, as the sets of models of integrity constraints. As examples of such applications, we mention the work of Pigozzi [16] in avoiding the *discursive dilemma*, the characterization of *safe agendas* by Grandi and Endriss [10] and that of Endriss and de Haan [9] concerning the *winner determination problem*. Our proofs draw from results in judgment aggregation theory as well as from results about propositional formulas and logical relations. Specifically, as stepping stones for our algorithmic syntactic characterization we use three results. First, a theorem implicit in Dokow and Holzman [5] stating that a domain is a possibility domain if and only if it either admits a binary (of arity 2) non-dictatorial aggregator or it is component-wise closed under the ternary direct sum. This result was generalized by Kirousis et al. [12] for domains in the non-Boolean framework. Second, a characterization of local possibility domains proven by Kirousis et al. in [12]. Lastly, the "unified framework for structure identification" by Zanuttini and Hébrard [21] (see next section for definitions).

Due to space restrictions, most proofs are omitted and can instead be found in [3].

2 Preliminaries

We first give the notation and basic definitions from Propositional Logic and judgment aggregation theory that we will use.

Let $V = \{x_1, \ldots, x_n\}$ be a set of Boolean variables. A literal is either a variable $x \in V$ (positive literal) or a negation $\neg x$ of it (negative literal). A clause is a disjunction $(l_{i_1} \vee \cdots \vee l_{i_k})$ of literals from different variables. A propositional formula ϕ (or just a "formula", without the specification "propositional", if clear from the context) in Conjunctive Normal Form (CNF) is a conjunction of clauses. A formula is called k-CNF if every clause of it contains exactly k literals. A (truth) assignment to the variables is an assignment of either 0 or 1 to each of the variables. We denote by $a(x)$ the value of x under the assignment a. Truth assignments will be identified with elements of $\{0,1\}^n$, or n-sequences of bits. The truth value of a formula for an assignment is computed by the usual rules that apply to logical connectives. The set of satisfying (returning the value 1) truth assignments, or models, of a formula, is denoted by $\mathrm{Mod}(\phi)$. In what follows, we will assume, except if specifically noted, that n denotes the number of variables of a formula ϕ and m the number of its clauses.

We say that a variable x appears *positively* (resp. *negatively*) in a clause C, if x (resp. $\neg x$) is a literal of C. A variable $x \in V$ is positively (resp. negatively) *pure* if it has only positive (resp. negative) appearances in ϕ.

A Horn clause is a clause with at most one positive literal. A dual Horn is a clause with at most one negative literal. A formula that contains only Horn (dual Horn) clauses is called Horn (dual Horn, respectively). Generalizing the notion of a clause, we will also call clauses sets of literals connected with exclusive OR (or direct sum), the logical connective that corresponds to summation in $\{0,1\}$ mod 2. Formulas obtained by considering a conjunction of such clauses are called affine. Finally, bijunctive are called the formulas whose clauses, in inclusive disjunctive form, have at most two literals. A domain $D \subseteq \{0,1\}^n$ is called Horn, dual Horn, affine or bijunctive respectively, if there is a Horn, dual Horn, affine or bijunctive formula ϕ of n variables such that $\mathrm{Mod}(\phi) = D$. In the previous section, we mentioned efficient solutions to classical syntactic characterization problems for classes of relations with given closure properties on one hand, and formulas of the syntactic forms mentioned above on the other.

We have presented the above notions and results without many details, as they are all classical results. For the notions that follow we give more detailed definitions and examples. The first one, as far as we can tell, dates back to 1978 (see Lewis [13]).

▶ **Definition 2.1.** *A formula ϕ whose variables are among the elements of the set $V = \{x_1, \ldots, x_n\}$ is called* renamable Horn*, if there is a subset $V_0 \subseteq V$ so that if we replace every appearance of every negated literal l from V_0 with the corresponding positive one and vice versa, ϕ is transformed to a Horn formula.*

The process of replacing the literals of some variables with their logical opposite ones, is called a *renaming* of the variables of ϕ.

▶ **Example 2.2.** Consider the formulas $\phi_1 = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_5)$ and $\phi_2 = (\neg x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_4 \vee x_5)$, defined over $V = \{x_1, x_2, x_3, x_4, x_5\}$.

The formula ϕ_1 is renamable Horn. To see this, let $V_0 = \{x_1, x_2, x_3, x_4\}$. By renaming these variables, we get the Horn formula $\phi_1^* = (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (x_2 \vee \neg x_3 \vee \neg x_5)$. On the other hand, it is easy to check that ϕ_2 cannot be transformed into a Horn formula for any subset of V, since for the first clause to become Horn, at least two variables from $\{x_2, x_3, x_4\}$ have to be renamed, which will make the second clause not Horn. ◇

It turns out that whether a formula is renamable Horn can be checked in linear time. There are several algorithms that do that in the literature, with the one of del Val [2] being a relatively recent such example. The original non-linear one was given by Lewis [13].

We now proceed with introducing several syntactic types of formulas:

▶ **Definition 2.3.** *A formula is called* separable *if its variables can be partitioned into two non-empty disjoint subsets so that no clause of it contains literals from both subsets.*

▶ **Example 2.4.** The formula $\phi_3 = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_4 \vee x_5)$ is separable. Indeed, for the partition $V_1 = \{x_1, x_2, x_3\}$, $V_2 = \{x_4, x_5\}$ of V, we have that no clause of ϕ_3 contains variables from both subsets of the partition. On the other hand, there is no such partition of V for neither ϕ_1 nor ϕ_2 of the previous example. ◇

The fact that separable formulas can be recognized in linear time is relatively straightforward (see Proposition 3.1 in Subsection 3.1).

We now introduce the following notions:

▶ **Definition 2.5.** *A formula ϕ is called* partially Horn *if there is a nonempty subset $V_0 \subseteq V$ such that (i) the clauses containing only variables from V_0 are Horn and (ii) the variables of V_0 appear only negatively (if at all) in a clause containing also variables not in V_0.*

If a formula ϕ is partially Horn, then any non-empty subset $V_0 \subseteq V$ that satisfies the requirements of Definition 2.5 will be called an *admissible set of variables*. Also the Horn clauses that contain variables only from V_0 will be called *admissible clauses* (the set of admissible clauses might be empty). A Horn clause with a variable in $V \setminus V_0$ will be called *inadmissible* (the reason for the possible existence of such clauses will be made clear in the following example).

Notice that a Horn formula is, trivially, partially Horn too, as is a formula that contains at least one negative pure literal. It immediately follows that the satisfiability problem remains NP-complete even when restricted to partially Horn formulas (just add a dummy negative pure literal). However, in Computational Social Choice, domains are considered to be non-empty as a non-degeneracy condition. Actually, it is usually assumed that the projection of a domain to any one of the n issues is the set $\{0, 1\}$.

▶ **Example 2.6.** We first examine the formulas of the previous examples. ϕ_1 is partially Horn, since it contains the negative pure literal $\neg x_5$. The Horn formula ϕ_1^* is also trivially partially Horn. On the other hand, ϕ_2 and ϕ_3 are not, since for every possible $V_0 \subseteq \{x_1, x_2, x_3, x_4, x_5\}$, we either get non-Horn clauses containing variables only from V_0, or variables of V_0 that appear positively in inadmissible clauses.

The formula $\phi_4 = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_3 \vee x_4)$ is partially Horn. Its first three clauses are Horn, though the third has to be put in every inadmissible set, since x_3 appears positively in the fourth clause which is not Horn. The first two clauses though constitute an admissible set of Horn clauses. Finally, $\phi_5 = (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_3 \vee x_4)$ is not partially Horn. Indeed, since all its variables appear positively in some clause, we need at least one clause to be admissible. The first two clauses of ϕ_5 are Horn, but we will show that they both have to be included in an inadmissible set. Indeed, the second has to belong to every inadmissible set since x_3 appears positively in the third, not Horn, clause. Furthermore, x_2 appears positively in the second clause, which we just showed to belong to every inadmissible set. Thus, the first clause also has to be included in every inadmissible set, and therefore ϕ_5 is not partially Horn. ◇

Accordingly to the case of renamable Horn formulas, we define:

▶ **Definition 2.7.** *A formula is called* renamable partially Horn *if some of its variables can be renamed (in the sense of Definition 2.1) so that it becomes partially Horn.*

Observe that any Horn, renamable horn or partially Horn formula is trivially renamable partially Horn. Also, a formula with at least one pure positive literal is renamable partially Horn, since by renaming the corresponding variable, we get a formula with a pure negative literal.

▶ **Example 2.8.** All formulas of the previous examples are renamable partially Horn: ϕ_1^*, ϕ_1 and ϕ_4 correspond to the trivial cases we discussed above, whereas ϕ_2, ϕ_3 and ϕ_5 all contain the pure positive literal x_4.

Lastly, we examine two more formulas: $\phi_6 = (\neg x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge \neg x_4$ is easily not partially Horn, but by renaming x_4, we obtain the partially Horn formula $\phi_6^* = (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge x_4$, where $V_0 = \{x_4\}$ is the set of admissible variables. One the other hand, the formula $\phi_7 = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$ is not renamable partially Horn. Indeed, whichever variables we rename, we end up with one Horn and one non-Horn clause, with at least one variable of the Horn clause appearing positively in the non-Horn clause. ◇

We prove, by Theorem 3.3 in Subsection 3.1 that checking whether a formula is renamable partially Horn can be done in linear time in the length of the formula.

▶ Remark 2.9. Let ϕ be a renamable partially Horn formula, and let ϕ^* be a partially Horn formula obtained by renaming some of the variables of ϕ, with V_0 being the admissible set of variables. Let also $\mathcal{C}_0$ be an admissible set of Horn clauses in ϕ^*. We can assume that only variables of V_0 have been renamed, since the other variables are not involved in the definition of being partially Horn. Also, we can assume that a Horn clause of ϕ^* whose variables appear only in clauses in $\mathcal{C}_0$ belongs to $\mathcal{C}_0$. Indeed, if not, we can add it to $\mathcal{C}_0$. ◇

▶ **Definition 2.10.** *A formula is called a* possibility integrity constraint *if it is either separable, or renamable partially Horn or affine.*

From the above and the fact that checking whether a formula is affine is easy we get Theorem 3.4 in Subsection 3.1, which states that checking whether a formula is a possibility integrity constraint can be done in polynomial time in the size of the formula.

Now, given a clause C of a formula ϕ, we say that a *sub-clause* of C is any non-empty clause created by deleting at least one literal of C. In Quine [17] and Zanuttini and Hébrard [21], we find the following definitions:

▶ **Definition 2.11.** *A clause C of a formula ϕ is a* prime implicate *of ϕ if no sub-clause of C is logically implied by ϕ. Furthermore, ϕ is* prime *if all its clauses are prime implicates of it.*

In sub-section 3.2, we use this notion in order to efficiently construct formulas whose set of models is a possibility domain.

We now come to some notions from Social Choice Theory (for an introduction, see e.g. List [14]). In the sequel, we will deal with k sequences of n-bit-vectors, each of which belongs to a fixed domain $D \subseteq \{0,1\}^n$. It is convenient to present such sequences with an $k \times n$ matrix $x_j^i, i = 1, \ldots, k, j = 1, \ldots, n$ with bits as entries. The rows of this matrix are denoted by $x^i, i = 1, \ldots, k$ and the columns by $x_j, j = 1, \ldots, n$. Each row represents a row-vector of 0/1 decisions on n issues by one of k individuals. Each column represents the column-vector of the positions of all k individuals on a particular issue.

In Social Choice Theory, $D \subseteq \{0,1\}^n$ is said to have a k-ary (of arity k) unanimous aggregator if there are exists a sequence of n k-ary Boolean functions $(f_1, \ldots, f_n)$, $f_j : \{0,1\}^k \to \{0,1\}, j = 1, \ldots, n$ such that

- all f_j are unanimous, i.e if $b_1 = \cdots = b_k$ are equal bits, then

  $$f_j(b_1, \ldots, b_k) = b_1 = \cdots = b_k, \text{ and}$$

- if for a matrix $(x^i_j)_{i,j}$ that represents the opinions of k individuals on n issues we have that the row-vectors $x^i \in D$ for all $i = 1, \ldots, k$, then

  $$(f_1(x_1), \ldots, f_n(x_n)) \in D.$$

Notice that in the second bullet above, the f_j's are applied to column-vectors, which have dimension k. The f_j's are called the *components* of the aggregator $(f_1, \ldots, f_n)$. Intuitively, an aggregator is a sequence of functions that when applied onto some rational opinion vectors of k individuals on n issues, in a issue-by-issue fashion, they return a row-vector that is still rational. From now on, we will refer to unanimous aggregators, simply as aggregators. We will also sometimes say that F is an aggregator, meaning that F is a sequence of n functions $(f_1, \ldots, f_n)$ as above.

An aggregator $(f_1, \ldots, f_n)$ is called *dictatorial* if there is a $d = 1 \ldots, k$ such that $f_1 = \cdots = f_n = \mathrm{pr}^k_d$, where $\mathrm{pr}^k_d : (b_1, \ldots, b_k) \mapsto b_d$ is the k-ary projection function on the d'th coordinate.

A k-ary aggregator is called a *projection* aggregator if each of its components is a projection function pr^k_d, for some $d = 1, \ldots, m$.

Notice that it is conceivable to have non-dictatorial aggregators that are projection aggregators.

A binary (of arity 2) Boolean function $f : \{0,1\}^2 \to \{0,1\}$ is called *symmetric* if for all pairs of bits b_1, b_2, we have that $f(b_1, b_2) = f(b_2, b_1)$. A binary aggregator is called symmetric if all its components are symmetric. Let us mention here the easily to check fact that the only unanimous binary functions are the $\wedge$, $\vee$ and the two projection functions $\mathrm{pr}^2_1, \mathrm{pr}^2_2$. Of those four, only the first two are symmetric.

▶ **Definition 2.12.** *A domain D is called a possibility domain if it has a (unanimous) non-dictatorial aggregator of some arity.*

Notice that the search space for such an aggregator is large, as the arity is not restricted. However, from [12, Theorem 3.7] (a result that follows from Dokow and Holzman [5], but without being explicitly mentioned there), we can easily get that:

▶ **Theorem 2.13** (Dokow and Holzman [5])**.** *A domain D is a possibility domain if and only if it admits either: (i) a non-dictatorial binary projection aggregator or (ii) a non-projection binary aggregator (i.e. at least one symmetric component) or (iii) a ternary aggregator all components of which are the binary addition* mod 2*.*

Nehring and Puppe [15] defined a type of non-dictatorial aggregators they called *locally non-dictatorial*. A k-ary aggregator $(f_1, \ldots, f_n)$ is locally non-dictatorial if $f_j \neq \mathrm{pr}^k_d$, for all $d \in \{1, \ldots, k\}$ and $j = 1, \ldots, n$.

▶ **Definition 2.14.** *D is a* local possibility domain (lpd) *if it admits a locally non-dictatorial aggregator.*

Consider the following ternary operators on $\{0,1\}$: (i) $\wedge^{(3)}(x,y,z) := \wedge(\wedge(x,y),z))$ (resp. for $\vee^{(3)}$), (ii) maj, where $\mathrm{maj}(x,y,z) = 1$ if and only if *at least* two elements of its input are 1 and (iii) $\oplus$, where $\oplus(x,y,z) = 1$ if an only if *exactly one or all* of the elements of its input are equal to 1. In [12], the following characterization of lpd's has been proven:

▶ **Theorem 2.15** (Kirousis et al. [12])**.** *$D \subseteq \{0,1\}$ is a local possibility domain if and only if it admits a ternary aggregator $(f_1, \ldots, f_n)$ such that $f_j \in \{\wedge^{(3)}, \vee^{(3)}, \mathrm{maj}, \oplus\}$, for $j = 1, \ldots, n$.*

3 Syntactic characterization of possibility domains by possibility integrity constraints

3.1 Identifying possibility integrity constraints

In this subsection, we show that identifying possibility integrity constraints can be done in time linear in the length of the input formula. By Definition 2.10, it suffices to show that for separable and renamable partially Horn formulas, since the corresponding problem for affine formulas is trivial.

In all that follows, we assume that we have a set of variables $V := \{x_1, \ldots, x_n\}$ and a formula ϕ defined on V that is a conjunction of m clauses $C_1, \ldots, C_m$, where $C_j = (l_{j_1}, \ldots, l_{j_{k_j}})$, $j = 1, \ldots, n$, and l_{j_s} is a positive or negative literal of x_{j_s}, $s = 1, \ldots, k_j$. We denote the set of variables corresponding to the literals of a clause C_j by $\mathrm{vbl}(C_j)$.

We begin with the result for separable formulas:

▶ **Proposition 3.1.** *There is an algorithm that, on input a formula ϕ, halts in time linear in the length of ϕ and either returns that the formula is not separable, or alternatively produces a* partition *of V in two non-empty and disjoint subsets $V_1, V_2 \subseteq V$, such that no clause of ϕ contains variables from both V_1 and V_2.*

Proof. (Sketch; detailed proof provided in [3].) Let the variables of ϕ be the vertices of a simple graph G. We connect two such vertices if they appear consecutively in a common clause of ϕ. The result is then obtained by showing that ϕ is separable if and only if G is not connected. ◀

To deal with renamable partially Horn formulas, we will start with Lewis' idea [13] of creating, for a formula ϕ, a 2SAT formula ϕ' whose satisfiability is equivalent to ϕ being renamable Horn. However, here we need to (i) look for a renaming that might transform only some clauses into Horn and (ii) deal with inadmissible Horn clauses, since such clauses can cause other Horn clauses to become inadmissible too.

▶ **Proposition 3.2.** *For every formula ϕ, there is a formula ϕ' such that ϕ is renamable partially Horn if and only if ϕ' is satisfiable.*

Proof. (Sketch; detailed proof provided in [3].) For each variable $x \in V$, we introduce a new variable x'. Intuitively, setting $x = 1$ means that x is renamed, whereas setting $x' = 1$ means that x is in V_0, but is not renamed. Finally we set both x and x' equal to 0 in case x is not in V_0. Obviously, we should not not allow the assignment $x = x' = 1$ (a variable in V_0 cannot be renamed and not renamed).

Suppose that a clause C of ϕ has the literals $x, \neg y$. If we add x to V_0 without renaming it, we should not rename y, since we would have two positive literals in an admissible clause. Also, we should not leave the latter out of V_0, since we would have a variable of V_0 appearing positively in a clause containing a variable not in V_0. Thus, we have that $x' \to y'$, which is expressed by the equivalent clause $(\neg x' \vee y')$. We add this clause to ϕ' and we proceed in this way for any possible combination of literals in a clause of ϕ. We also introduce the clauses $\neg x \vee \neg x'$, for all $x \in V$, in order to exclude the assignment $x = x' = 1$. Finally, we add the clause $\bigvee_{x \in V'} x$, to ensure that at least one variable is admissible. ◀

To compute ϕ' from ϕ, one would need quadratic time in the length of ϕ. Thus, we introduce the following linear algorithm that decides if a formula ϕ is renamable partially Horn, by tying a property of a graph constructed based on ϕ, with the satisfiability of ϕ'.

▶ **Theorem 3.3.** *There is an algorithm that, on input a formula ϕ, halts in time linear in the length of ϕ and either returns that ϕ is not renamable partially Horn or alternatively produces a subset $V^* \subseteq V$ such that the formula ϕ^* obtained from ϕ by renaming the literals of variables in V^* is partially Horn.*

Proof. (Sketch; detailed proof provided in [3].) To prove Theorem 3.3, we define a *directed bipartite* graph G, i.e. a directed graph whose set of vertices is partitioned in two sets such that no vertices belonging in the same part are adjacent. One set is comprised of the variables of ϕ', and the other of the clauses of ϕ. Each variable is connected with the clauses it appears in. Then, by computing its *strongly connected components (scc)*, i.e. its maximal sets of vertices such that every two of them are connected by a directed path, we show that at least one of them does not contain both a variable x and x' (and thus allows x to be admissible) *if and only if* ϕ is renamable partially Horn. ◀

Because checking whether a formula is affine can be trivially done in linear time, we get:

▶ **Theorem 3.4.** *There is an algorithm that, on input a formula ϕ, halts in linear time in the length of ϕ and either returns that ϕ is not a possibility integrity constraint, or alternatively, (i) either it returns that ϕ is affine or (ii) in case ϕ is separable, it produces two non-empty and disjoint subsets $V_1, V_2 \subseteq V$ such that no clause of ϕ contains variables from both V_1 and V_2 and (iii) in case ϕ is renamable partially Horn, it produces a subset $V^* \subseteq V$ such that the formula ϕ^* obtained from ϕ by renaming the literals of variables in V^* is partially Horn.*

3.2 Syntactic Characterization of possibility domains

In this subsection, we provide a syntactic characterization for possibility domains, by proving they are the models of possibility integrity constraints. Furthermore, we show that given a possibility domain D, we can produce a possibility integrity constraint, whose set of models is D, in time polynomial in the size of D. To obtain the characterization, we proceed as follows. We separately show that each type of a possibility integrity constraint of Definition 2.10 corresponds to one of the conditions of Theorem 2.13: (i) Domains admitting non-dictatorial binary projection aggragators are the sets of models of separable formulas, those admitting non-projection binary aggregators are the sets of models of renamable partially Horn formulas and (iii) affine domains are the sets of models of affine formulas.

We will need some additional notation. For a set of indices I, let $D_I := \{(a_i)_{i\in I} \mid a \in D\}$ be the projection of D to the indices of I and $D_{-I} := D_{\{1,\dots,n\}\setminus I}$. Also, for two (partial) vectors $a = (a_1, \dots, a_k) \in D_{\{1,\dots,k\}}$, $k < n$ and $b = (b_1, \dots, b_{n-k}) \in D_{\{k+1,\dots,n\}}$, we define their *concatenation* to be the vector $ab = (a_1, \dots, a_k, b_1, \dots, b_{n-k})$. Finally, given two subsets $D, D' \subseteq \{0,1\}^n$, we write that $D \approx D'$ if we can obtain D by *permuting* the coordinates of D', i.e. if $D = \{(d_{j_1}, \dots, d_{j_n}) \mid (d_1, \dots, d_n) \in D'\}$, where $\{j_1, \dots, j_n\} = \{1, \dots, n\}$.

We begin with characterizing the domains closed under a non-dictatorial projection aggregator as the models of separable formulas.

▶ **Proposition 3.5.** *D admits a binary non-dictatorial projection aggregator $(f_1, \dots, f_n)$ if and only if there exists a separable formula ϕ whose set of models equals D.*

Proof. (Sketch; detailed proof provided in [3].) First prove that D admits a binary non-dictatorial projection aggregator if an only if there exists a partition (I, J) of $\{1, \dots, n\}$ such that $D \approx D_I \times D_J$. To do that, take I to be the indices of the aggregator that are projections to the first coordinate and J that of the indices that are projections to the second coordinate.

Then, take the formulas ϕ_1 and ϕ_2 such that $\mathsf{Mod}(\phi_1) = D_I$ and $\mathsf{Mod}(\phi_2) = D_J$ and prove that $\mathsf{Mod}(\phi_1 \wedge \phi_2) = D$. ◀

We now turn our attention to domains closed under binary non projection aggregators.

▶ **Theorem 3.6.** *D admits a binary aggregator $(f_1, \ldots, f_n)$ which is not a projection aggregator if and only if there exists a renamable partially Horn formula ϕ whose set of models equals D.*

Proof. (Sketch; detailed proof provided in [3].) We first show that any domain D admitting a binary aggregator that has some components being projections to different coordinates, also admits one whose projections are all to the same coordinate. Also, given a domain admitting a binary aggregator with some components being $\vee$, we construct a domain D^* admitting a binary aggregator that all of its symmetric components are $\wedge$.

We then proceed to describing a partially Horn formula $\phi = \phi_0 \wedge \phi_1$, such that ϕ_0 is Horn and describes D_I, the projection of D to the indices corresponding to the symmetric components of $(f_1, \ldots, f_n)$. ϕ_1 is then constructed as the conjunction of smaller formulas that describe the sets of partial vectors that extend those of D_I. We also ensure that any variable of ϕ_0 appears only negatively in ϕ_1. ◀

We thus get:

▶ **Theorem 3.7.** *D is a possibility domain if and only if there exists a possibility integrity constraint ϕ whose set of models equals D.*

Proof. (Sketch; detailed proof provided in [3].) The proof follows by combining the characterization of possibility domains of Theorem 2.13, with Proposition 3.5 for separable formulas, Theorem 3.6, for renamable partially Horn formulas and the fact that an affine domain is described by an affine formula. ◀

To finish this section, we will use Zanuttini and Hébrard's "unified framework" [21]. Recall the definition of a prime formula (Def. 2.11) and consider the following proposition:

▶ **Proposition 3.8.** *Let ϕ_P be a prime formula and ϕ be a formula logically equivalent to ϕ_P. Then:*
1. *if ϕ is separable, ϕ_P is also separable and*
2. *if ϕ is renamable partially Horn, ϕ_P is also renamable partially Horn.*

Proof. (Sketch; detailed proof provided in [3].) Follows from the fact that neither *resolution* nor *omission* can destroy separability or make an admissible variable non-admissible (see also Quine [17]). ◀

We are now ready to prove our main result:

▶ **Theorem 3.9.** *There is an algorithm that, on input $D \subseteq \{0,1\}^n$, halts in time $O(|D|^2 n^2)$ and either returns that D is not a possibility domain, or alternatively outputs a possibility integrity constraint ϕ, containing $O(|D|n)$ clauses, whose set of satisfying truth assignments is D.*

Proof. Given a domain D, we first use Zanuttini and Hébrard's algorithm to check if it is affine [21, Proposition 8], and if it is, produce, in time $O(|D|^2 n^2)$ an affine formula ϕ with $O(|D|n)$ clauses, such that $\mathsf{Mod}(\phi) = D$. If it isn't, we use again Zanuttini and Hébrard's algorithm [21] to produce, in time $O(|D|^2 n^2)$, a prime formula ϕ with $O(|D|n)$ clauses, such that $\mathsf{Mod}(\phi) = D$. Then, we use the linear algorithms of Proposition 3.1 and Theorem 3.3 to check if ϕ is separable or renamable partially Horn. If it is either of the two, then ϕ is a possibility integrity constraint and, by Theorem 3.7, D is a possibility domain. Else, by Proposition 3.8, D is not a possibility domain. ◀

4 Local possibility domains

We turn now our attention to local possibility domains. As in the case of possibility domains, we want to characterize lpd's with a syntactic type of formulas.

We will first address a technical issue. Let V, V' be two disjoint sets of variables. By further generalizing the notion of a clause of a CNF formula, we say that a (V, V')-*generalized clause* is a clause of the form:

$$(l_1 \vee \cdots \vee l_s \vee (l_{s+1} \oplus \cdots \oplus l_t)),$$

where the literal l_j corresponds to variable v_j, $j = 1, \ldots, t$, $v_1, \ldots, v_s \in V$, $v_{s+1}, \ldots, v_t \in V'$ and $0 \leq s < t$. Such a clause if falsified by exactly those assignments that falsify every literal l_i, $i = 1, \ldots, s$ and satisfy an even number of literals l_j, $j = s+1, \ldots, t$. An affine clause is trivially a (V, V')-generalized clause, where all its literals correspond to variables from V'.

Consider now the following definition, which is analogous to Definition 2.10.

▶ **Definition 4.1.** *A formula ϕ is a* local possibility integrity constraint (lpic) *if there are three* pairwise disjoint *subsets $V_0, V_1, V_2 \subseteq V$, with $V_0 \cup V_1 \cup V_2 = V$, where no clause contains variables both from V_1 and V_2 and such that:*

1. *by renaming some variables of V_0, we obtain a partially Horn formula ϕ^*, whose set of admissible variables is V_0,*
2. *any clause contains at most two variables from V_1 and*
3. *the clauses containing variables from V_2 are (V_0, V_2)-generalized clauses.*

▶ **Example 4.2.** Easily, every (renamable) Horn, bijunctive or affine formula is an lpic. On the other hand, consider the following possibility integrity constraint:

$$\phi = (\neg x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_3 \vee \neg x_4).$$

ϕ is partially Horn, since it has the pure negative literal $\neg x_1$ and thus a possibility integrity constraint. But, it is not an lpic, since however we define V_0, V_1, either there will be a variable of V_0 with a positive appearence in a non-admissible clause (even after any possible renaming of the variables of V_0) and/or there will be a clause with more than two literals from V_1. ◇

By Definition 4.1, we get the following corollary:

▶ **Corollary 4.3.** *If ϕ is a local possibility integrity constraint, then it is also a possibility integrity constraint.*

Proof. Let V_0, V_1 and V_2 be as in Definition 4.1. If $V_0 \neq \emptyset$, ϕ is partially Horn. Else, if $V_0 = V_1 = \emptyset$, then ϕ is affine. On the other hand, if $V_0 = \emptyset$ and V_1 and V_2 are not, ϕ is separable. Finally, if $V_1 = V$, then ϕ is bijunctive and equivalently, 2-SAT. The result now follows by the fact that any 2-SAT formula is renamable Horn. Indeed, let α be an assignment satisfying ϕ and rename all the variables $x \in V$ such that $\alpha(x) = 1$. Then, every clause of ϕ either has a positive literal that is renamed, or a negative one that is not renamed. ◀

The first theorem we prove is that we can recognize lpic's efficiently.

▶ **Theorem 4.4.** *There is an algorithm that, on input a formula ϕ, halts in linear time in the length of ϕ and either returns that ϕ is not a local possibility constraint, or alternatively, produces the sets V_0, V_1, V_2 described in Definition 4.1.*

Proof. (Sketch; detailed proof provided in [3].) By Theorem 3.3, we check if ϕ is renamable partially Horn in time linear to its length and obtain the set V_0 of admissible variables, in case it is. Then, we can trivially check if any sub-clause, obtained by a non-admissible clause by deleting any variable from V_0 is bijunctive or affine. ◀

We now syntactically characterize lpd's as the sets of models of lpic's.

▶ **Theorem 4.5.** *A domain $D \subseteq \{0,1\}^n$ is a local possibility domain if and only if there is a local possibility integrity constraint ϕ such that $\mathsf{Mod}(\phi) = D$.*

Proof. (Sketch; detailed proof provided in [3].) The proof is a variation of the one for Theorem 3.6. ◀

We end this section by showing that, given an lpd D, we can efficiently construct an lpic ϕ such that $\mathsf{Mod}(\phi) = D$.

▶ **Theorem 4.6.** *There is an algorithm that, on input $D \subseteq \{0,1\}^n$, halts in time $O(|D|^2 n^2)$ and either returns that D is not a local possibility domain, or alternatively outputs a local possibility integrity constraint ϕ, containing $O(|D|n)$ clauses, whose set of satisfying truth assignments is D.*

Proof. (Sketch; detailed proof provided in [3].) Using Zanuttini and Hébrard's unified framework, we compute a prime formula ϕ such that $\mathsf{Mod}(\phi) = D$ in time $O(|D|^2 n^2)$ that contains $O(|D|n)$ clauses. We then check, in linear time to the length of ϕ, whether it is an lpic. ◀

Concluding remarks

It is known that any domain on n issues can be represented either by n formulas $\phi_1, \ldots, \phi_n$ (an agenda), in which case the domain is the set of binary n-vectors, the i-component of which represents the acceptance or rejection of ϕ_i in a consistent way (logic-based approach), or, alternatively, by a single formula ϕ of n variables (an integrity constraint), in which case the domain is the set of models of ϕ. In the former case, there are results, albeit of non-algorithmic nature, that give us conditions on the syntactic form of the ϕ_i's, so that the domain accepts a non-dictatorial aggregator. In this work, we give a necessary and sufficient condition on the syntactic form of a formula to be an integrity constraint of a domain that accepts a (locally) non-dictatorial aggregator. We called such formulas, (local) possibility integrity constraints. Our results are algorithmic, in the sense that (i) recognizing a (local) possibility integrity constraint can be implemented in time linear in the length of the input formula and (ii) given a (local) possibility domain, a corresponding (local) possibility integrity constraint, whose number of clauses is polynomial in the size of the domain, can be constructed in time polynomial in the size of the domain. Our proofs draw from results in judgment aggregation theory as well from results about propositional formulas and logical relations.

References

1 Rina Dechter and Judea Pearl. Structure identification in relational data. *Artificial Intelligence*, 58(1-3):237–270, 1992.

2 Alvaro del Val. On 2-SAT and renamable Horn. In *Proceedings of the National Conference on Artificial Intelligence*, pages 279–284. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2000.

3 Josep Díaz, Lefteris Kirousis, Sofia Kokonezi, and John Livieratos. Algorithmically Efficient Syntactic Characterization of Possibility Domains. *arXiv preprint*, 2019. `arXiv:1901.00138`.

4 Franz Dietrich. A generalised model of judgment aggregation. *Social Choice and Welfare*, 28(4):529–565, 2007.

5 Elad Dokow and Ron Holzman. Aggregation of binary evaluations for truth-functional agendas. *Social Choice and Welfare*, 32(2):221–241, 2009.

6 Elad Dokow and Ron Holzman. Aggregation of binary evaluations. *Journal of Economic Theory*, 145(2):495–511, 2010.

7 Ramez Elmasri and Sham Navathe. *Fundamentals of database systems*. Pearson London, 2016.

8 Herbert Enderton and Herbert B Enderton. *A mathematical introduction to logic*. Elsevier, 2001.

9 Ulle Endriss and Ronald de Haan. Complexity of the winner determination problem in judgment aggregation: Kemeny, Slater, Tideman, Young. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 117–125. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

10 Umberto Grandi and Ulle Endriss. Binary aggregation with integrity constraints. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 204, 2011.

11 Umberto Grandi and Ulle Endriss. Lifting integrity constraints in binary aggregation. *Artificial Intelligence*, 199:45–66, 2013.

12 Lefteris Kirousis, Phokion G Kolaitis, and John Livieratos. Aggregation of votes with multiple positions on each issue. In *Proceedings 16th International Conference on Relational and Algebraic Methods in Computer Science*, pages 209–225. Springer, 2017. Expanded version to appear in *ACM Transactions on Economics and Computation*.

13 Harry R Lewis. Renaming a set of clauses as a Horn set. *Journal of the ACM (JACM)*, 25(1):134–135, 1978.

14 Christian List. The theory of judgment aggregation: An introductory review. *Synthese*, 187(1):179–207, 2012.

15 Klaus Nehring and Clemens Puppe. Abstract arrowian aggregation. *Journal of Economic Theory*, 145(2):467–494, 2010.

16 Gabriella Pigozzi. Belief merging and the discursive dilemma: an argument-based account to paradoxes of judgment aggregation. *Synthese*, 152(2):285–298, 2006.

17 Willard V Quine. On cores and prime implicants of truth functions. *The American Mathematical Monthly*, 66(9):755–760, 1959.

18 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proc. of the 10th Annual ACM Symp. on Theory of Computing*, pages 216–226, 1978.

19 Robert Wilson. On the theory of aggregation. *Journal of Economic Theory*, 10(1):89–99, 1975.

20 Susumu Yamasaki and Shuji Doshita. The satisfiabilty problem for a class consisting of Horn sentences and some non-Horn sentences in proportional logic. *Information and Control*, 59(1-3):1–12, 1983.

21 Bruno Zanuttini and Jean-Jacques Hébrard. A unified framework for structure identification. *Information Processing Letters*, 81(6):335–339, 2002.

On Geometric Complexity Theory: Multiplicity Obstructions Are Stronger Than Occurrence Obstructions

Julian Dörfler
Saarland University, Saarbrücken, Germany
s8judoer@stud.uni-saarland.de

Christian Ikenmeyer
Max Planck Institute for Software Systems, Saarbrücken, Germany
cikenmey@mpi-sws.org

Greta Panova
University of Southern Californa, Los Angeles, CA, USA
University of Pennsylvania, Philadelphia, PA, USA
gpanova@usc.edu

Abstract

Geometric Complexity Theory as initiated by Mulmuley and Sohoni in two papers (SIAM J Comput 2001, 2008) aims to separate algebraic complexity classes via representation theoretic multiplicities in coordinate rings of specific group varieties. We provide the first toy setting in which a separation can be achieved for a family of polynomials via these multiplicities.

Mulmuley and Sohoni's papers also conjecture that the vanishing behavior of multiplicities would be sufficient to separate complexity classes (so-called *occurrence obstructions*). The existence of such strong occurrence obstructions has been recently disproven in 2016 in two successive papers, Ikenmeyer-Panova (Adv. Math.) and Bürgisser-Ikenmeyer-Panova (J. AMS). This raises the question whether separating group varieties via representation theoretic multiplicities is stronger than separating them via occurrences. We provide first finite settings where a separation via multiplicities can be achieved, while the separation via occurrences is provably impossible. These settings are surprisingly simple and natural: We study the variety of products of homogeneous linear forms (the so-called Chow variety) and the variety of polynomials of bounded border Waring rank (i.e. a higher secant variety of the Veronese variety).

As a side result we prove a slight generalization of Hermite's reciprocity theorem, which proves Foulkes' conjecture for a new infinite family of cases.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Algebraic complexity theory, geometric complexity theory, Waring rank, plethysm coefficients, occurrence obstructions, multiplicity obstructions

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.51

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1901.04576`.

Funding *Julian Dörfler*: Partially supported by DFG grant IK 116/2-1.
Christian Ikenmeyer: Partially supported by DFG grant IK 116/2-1.
Greta Panova: Partially funded by the NSF.

Acknowledgements This work was done in part while CI and GP were visiting the Simons Institute for the Theory of Computing.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 51; pp. 51:1–51:14

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

1 Introduction

In two landmark papers [22, 23] Mulmuley and Sohoni suggested the use of representation theoretic multiplicities to separate group varieties that correspond to complexity classes. The goal of this approach, which is called *geometric complexity theory*, is to achieve complexity lower bounds that lead to the separation of algebraic complexity classes such as VP and VNP (see [3] or [28] for the precise definitions, which will not be important in this paper). At the heart of the approach was the hope that so-called *occurrence obstructions* (see Section 2) would be sufficient to separate VP and VNP. In [16, 8] it was shown that occurrence obstructions are too weak to provide the necessary separation, at least for the group varieties that were originally proposed by Mulmuley and Sohoni. But representation theoretic multiplicities might still be able to separate VP and VNP when we look at the finer separation criterion via *multiplicity obstructions* (see also Section 2). Unfortunately, so far all known separations of group varieties via multiplicity obstructions could also in fact be obtained via occurrence obstructions, or at least there is no setting in which multiplicity obstructions are provably stronger than occurrence obstructions, see e.g. [6, 7]. Indeed, little is known about multiplicity obstructions in general, as the required multiplicities are often #P-hard to compute, see e.g. [25, 5, 2], which implies that a polynomial time algorithm for their computation can only exist if P=NP.

Scott Aaronson raised the question about the existence of a setting where multiplicity obstructions are provably more powerful than occurrence obstructions. In this paper we give the first example of such a situation in a finite setting, see Theorem 2.1 below.

Theorem 2.1 is not only about finite settings: For the first time multiplicity obstructions are used to separate families of polynomials, even though the separation is extremely modest. Prior work on obstructions focused on tensors instead of polynomials ([6, 7]).

As a side result we prove a slight generalization of Hermite's reciprocity theorem, which proves Foulkes' conjecture (see (1)) for a new infinite family of cases, see Theorem 3.4.

2 Representation theoretic obstructions

In this section we review how to separate group varieties via representation theoretic multiplicities. The setup is in complete analogy to the geometric complexity theory approach of Mulmuley and Sohoni. We then list our main result, see Theorem 2.1.

Consider the space $\mathbb{A}^n_m := \mathbb{C}[x_1, \ldots, x_m]_n$ of complex homogeneous polynomials of degree n in m variables. Let $V := \mathbb{A}^1_m$ be the space of homogeneous degree 1 polynomials. In this paper we compare two subvarieties of $\mathbb{A}^n_m$. The first is the so-called *Chow variety*

$$\mathsf{Ch}^n_m := \{\ell_1 \cdots \ell_n \mid \ell_i \in V\} \subseteq \mathbb{A}^n_m,$$

which is the set of polynomials that can be written as a product of homogeneous linear forms, see e.g. [20, §8.6]. In algebraic complexity theory this set is known as the set of polynomials that have homogeneous depth-two algebraic circuits of the form $\Pi^n\Sigma$, i.e., circuits that consists of an n-ary top product gate of linear combinations of variables. The second variety is called a *higher secant variety of the Veronese variety* and can be written as

$$\mathsf{Pow}^n_{m,k} := \overline{\{\ell_1^n + \cdots + \ell_k^n \mid \ell_i \in V\}} \subseteq \mathbb{A}^n_m,$$

which is the closure of the set of all sums of k powers of homogeneous linear forms. Note that from a general principle it follows that the Zariski closure equals the Euclidean closure in this case, see e.g. [24, §2.C] where this is shown for every constructible set. The polynomials

in $\mathsf{Pow}^n_{m,k}$ are exactly those that have *border Waring rank at most k*, see e.g. [20, §5.4]. In algebraic complexity theory this set is known as the set of polynomials that can be approximated arbitrarily closely by homogeneous depth-three powering circuits of the form $\Sigma^k\Lambda^n\Sigma$, i.e., a k-ary sum of n-th powers of linear combinations of variables.

$\mathbb{A}^n_m$ is generated as a vector space by the powers v^n, $v \in V$, see e.g. [20, Ex. 2.6.6.2]. Given two elements $g_1, g_2 \in \mathsf{GL}_m := \mathsf{GL}(V)$, and given $v \in V$, we clearly have $g_1(g_2 v) = (g_1 g_2)v$. Thus we say that V admits a GL_m-action. This natural action of GL_m on V lifts canonically to $\mathbb{A}^n_m$ via $g(v^n) := (gv)^n$, $g \in \mathsf{GL}_m$, $v \in V$, and linear continuation. Both varieties Ch^n_m and $\mathsf{Pow}^n_{m,k}$ are closed under this action, i.e., for $g \in \mathsf{GL}_m$ and $v \in \mathsf{Ch}^n_m$ we have $gv \in \mathsf{Ch}^n_m$, and analogously $v \in \mathsf{Pow}^n_{m,k}$ implies $gv \in \mathsf{Pow}^n_{m,k}$. A variety that is closed under the action of GL_m is called a GL_m-variety.

Let $\mathbb{C}[\mathbb{A}^n_m]$ denote the coordinate ring of $\mathbb{A}^n_m$, i.e., the polynomial ring in $\dim \mathbb{A}^n_m = \binom{n+m-1}{n}$ many variables, where these variables are in 1:1 correspondence to the monomials in $\mathbb{A}^n_m$. The action of GL_m on $\mathbb{A}^n_m$ lifts to a linear action of GL_m on $\mathbb{C}[\mathbb{A}^n_m]$ via the canonical pullback as follows: $(gf)(h) := f(g^{-1}h)$, $g \in \mathsf{GL}_m$, $f \in \mathbb{C}[\mathbb{A}^n_m]$, $h \in \mathbb{A}^n_m$. Moreover, the action respects the natural grading of $\mathbb{C}[\mathbb{A}^n_m]$, so that each homogeneous degree d part $\mathbb{C}[\mathbb{A}^n_m]_d$ is a finite dimensional vector space that is closed under the action of GL_m.

Recall that a finite dimensional vector space W that is closed under a linear action of GL_m is called a GL_m-*representation*. This is equivalent to the existence of a group homomorphism $\varrho : \mathsf{GL}_m \to \mathsf{GL}(W)$. If we choose bases, then we can interpret $\mathsf{GL}_m \subseteq \mathbb{C}^{m\times m}$ and $\mathsf{GL}(W) \subseteq \mathbb{C}^{\dim W \times \dim W}$ and ϱ is described by $(\dim W)^2$ many coordinate functions, which are functions in m^2 many variables. If these functions are polynomials, then we call W a *polynomial representation*. Our main representation of interest, $\mathbb{C}[\mathbb{A}^n_m]_d$, is a polynomial representation. A linear subspace of W that is closed under the action of GL_m is called a subrepresentation. Subrepresentations of polynomial representations are clearly polynomial representations again. For every GL_m-representation W we have that W and 0 are two trivial subrepresentations. If W has no other subrepresentations, then we call W *irreducible*. A linear map $\varphi : W_1 \to W_2$ between two GL_m-representations is called *equivariant* if $g\varphi(f) = \varphi(gf)$ for all $f \in W_1$, $g \in \mathsf{GL}_m$. If there exists an equivariant vector space isomorphism from W_1 to W_2, then we say that W_1 and W_2 are *isomorphic* GL_m-representations. An *m-partition* of D is a nonincreasing list of m nonnegative integers that sum up to D. Every irreducible polynomial GL_m-representation has an associated isomorphism type, which is an m-partition, see e.g. [10, Ch. 8]. Two irreducible GL_m-representations are isomorphic iff their isomorphism types coincide. We denote by $\{\lambda\}_m$ the irreducible GL_m-representation corresponding to the m-partition λ. We write $\{\lambda\} = \{\lambda\}_m$ is m is clear from the context.

The group GL_m is *linearly reductive*, which means that every GL_m-representation W decomposes into a direct sum of irreducible GL_m-representations, see e.g. [18, AII.5, Satz 4]. The number of times an irreducible representation of type λ occurs in the decomposition is called the *multiplicity* of λ in W, written $\mathrm{mult}_\lambda(W)$. Even though this decomposition is usually not unique, the notation $\mathrm{mult}_\lambda(W)$ makes sense, because the multiplicities are independent of the actual decompositions.

The multiplicity $a_\lambda(d[n]) := \mathrm{mult}_\lambda(\mathbb{C}[\mathbb{A}^n_m]_d)$ is the infamous *plethysm coefficient*, which is the object of study in Foulkes' conjecture and also in Problem 9 in Stanley's famous list of open problems [30]. If we pad an m-partition λ with $m' - m$ many zeros to obtain the m'-partitions $\lambda' = (\lambda_1, \ldots, \lambda_m, 0, \ldots, 0)$, then $\mathrm{mult}_\lambda(\mathbb{C}[\mathbb{A}^n_m]_d) = \mathrm{mult}_{\lambda'}(\mathbb{C}[\mathbb{A}^n_{m'}]_d)$, see e.g. [14, Lem. 4.3.2]. For the sake of simplicity we identify m-partitions with m'-partitions that arise from padding zeros. This justifies leaving out the parameter m in the notation $a_\lambda(d[n])$ by assuming that m is large enough. Foulkes' conjecture states that

$$\text{Conjecture}: \quad a_\lambda(n[d]) \leq a_\lambda(d[n]) \quad \text{for all } d \geq n. \tag{1}$$

Conjecture (1) is known to be true (moreover, equality holds: $a_\lambda(d[n]) = a_\lambda(n[d])$) for all 2-partitions λ, which is often called *Hermite reciprocity* [13]. We make modest progress on this conjecture by proving it for many families of 3-partitions, see Corollary 4.4.

Let Z be a GL_m-variety, e.g., $Z = \mathsf{Ch}^n_m$ or $Z = \mathsf{Pow}^n_{m,k}$. Then the vanishing ideal $I(Z) := \{f \in \mathbb{C}[\mathbb{A}^n_m] \mid \forall h \in Z : f(h) = 0\}$ is also closed under the action of GL_m, which is easy to verify: If $f(h) = 0$ for all $h \in Z$, then also $(gf)(h) = f(g^{-1}h) = 0$, because $g^{-1}h \in Z$. Since the action respects the grading, each homogeneous degree d part $I(Z)_d$ is a GL_m-representation. The coordinate ring $\mathbb{C}[Z]$ is defined as the quotient algebra $\mathbb{C}[\mathbb{A}^n_m]/I(Z)$ and each homogeneous part $\mathbb{C}[Z]_d = \mathbb{C}[\mathbb{A}^n_m]_d/I(Z)_d$ is a GL_m-representation. Equivalently, we can define $\mathbb{C}[Z]$ as the set of restrictions of functions in $\mathbb{C}[\mathbb{A}^n_m]$ to Z.

For most sets of parameters we have $\mathsf{Pow}^n_{m,k} \not\subseteq \mathsf{Ch}^n_m$, but there are some exceptions. Clearly $\mathsf{Pow}^n_{m,1} \subseteq \mathsf{Ch}^n_m$. Moreover, $\mathsf{Pow}^n_{1,k} = \mathsf{Ch}^n_1$ for all $n \geq 1$, $k \geq 1$; and $\mathsf{Pow}^1_{m,k} = \mathsf{Ch}^1_m$ for all $m \geq 1$, $k \geq 1$. It is also easy to see that $\mathsf{Pow}^2_{2,2} \subseteq \mathsf{Ch}^2_2$, because $\ell_1^2+\ell_2^2 = (\ell_1+i\ell_2)(\ell_1-i\ell_2)$, where $i^2 = -1$. More generally, $(\ell_1 + \zeta\ell_2)(\ell_1 + \zeta^2\ell_2)\cdots(\ell_1 + \zeta^n\ell_2) = \ell_1^n + \zeta^{\frac{n(n+1)}{2}}\ell_2^n$ for $\zeta^n = 1$, which implies $\mathsf{Pow}^n_{m,2} \subseteq \mathsf{Ch}^n_m$. For $m = 2$, $k \geq 1$, $n \geq 1$, we have $\mathsf{Pow}^n_{m,k} \subseteq \mathsf{Ch}^n_m$ by the fundamental theorem of algebra. These are the only exceptions, as for $n \geq 2$, $m \geq 3$, $k \geq 3$ we have $\mathsf{Pow}^n_{m,k} \not\subseteq \mathsf{Ch}^n_m$: the polynomial $x^n + y^n + z^n$ of the Fermat curve is in $\mathsf{Pow}^n_{m,k}$ and its irreducibility implies (since $n \geq 2$) that $x^n + y^n + z^n \notin \mathsf{Ch}^n_m$.

We will see that for specific settings of parameters there exist *multiplicity obstructions* that prove $\mathsf{Pow}^n_{m,k} \not\subseteq \mathsf{Ch}^n_m$, but there do not exist *occurrence obstructions* that prove this fact (see the definitions below). Our approach works as follows and is in complete analogy to the approach proposed in [22, 23] to separate group varieties arising from algebraic complexity theory. If $\mathsf{Pow}^n_{m,k} \subseteq \mathsf{Ch}^n_m$, then the restriction of functions gives a canonical GL_m-equivariant surjection $\mathbb{C}[\mathsf{Ch}^n_m]_d \twoheadrightarrow \mathbb{C}[\mathsf{Pow}^n_{m,k}]_d$. In this case, Schur's lemma (e.g. [11, Lemma 4.1.4]) implies that

$$\mathrm{mult}_\lambda(\mathbb{C}[\mathsf{Ch}^n_m]_d) \geq \mathrm{mult}_\lambda(\mathbb{C}[\mathsf{Pow}^n_{m,k}]_d). \tag{2}$$

for all m-partitions λ. Therefore, a partition λ that violates (2) proves that $\mathsf{Pow}^n_{m,k} \not\subseteq \mathsf{Ch}^n_m$. Such a λ is called a *multiplicity obstruction*. If additionally $\mathrm{mult}_\lambda(\mathbb{C}[\mathsf{Ch}^n_m]_d) = 0$, then λ is called an *occurrence obstruction*.

Since Ch^n_m and $\mathsf{Pow}^n_{m,k}$ are subvarieties of $\mathbb{A}^n_m$ and since all λ for which $\mathrm{mult}_\lambda(\mathbb{C}[\mathbb{A}^n_m]_d) > 0$ are m-partitions of dn, it follows that if $\mathrm{mult}_\lambda(\mathbb{C}[\mathsf{Ch}^n_m]_d) > 0$ or $\mathrm{mult}_\lambda(\mathbb{C}[\mathsf{Pow}^n_{m,k}]_d) > 0$, then λ is an m-partition of dn.

▶ **Theorem 2.1** (Main Theorem).

(1) *Asymptotic result: Let $m \geq 3$, $n \geq 2$, $k = d = n + 1$, $\lambda = (n^2 - 2, n, 2)$. We have $\mathrm{mult}_\lambda(\mathbb{C}[\mathsf{Ch}^n_m]_d) < \mathrm{mult}_\lambda(\mathbb{C}[\mathsf{Pow}^n_{m,k}]_d)$, i.e., λ is a multiplicity obstruction that shows $\mathsf{Pow}^n_{m,k} \not\subseteq \mathsf{Ch}^n_m$.*

(2) *Finite result: In two finite settings we can show a slightly stronger separation:*

- **(a)** *Let $k = 4$, $n = 6$, $m = 3$, $d = 7$, $\lambda = (n^2 - 2, n, 2) = (34, 6, 2)$. Then $\mathrm{mult}_\lambda(\mathbb{C}[\mathsf{Ch}^n_m]_d) = 7 < 8 = \mathrm{mult}_\lambda(\mathbb{C}[\mathsf{Pow}^n_{m,k}]_d)$, i.e., λ is a multiplicity obstruction that shows $\mathsf{Pow}^n_{m,k} \not\subseteq \mathsf{Ch}^n_m$.*
- **(b)** *Similarly, for $k = 4$, $n = 7$, $m = 4$, $d = 8$, $\lambda = (n^2 - 2, n, 2) = (47, 7, 2)$ we have $\mathrm{mult}_\lambda(\mathbb{C}[\mathsf{Ch}^n_m]_d) < 11 = \mathrm{mult}_\lambda(\mathbb{C}[\mathsf{Pow}^n_{m,k}]_d)$, i.e., λ is a multiplicity obstruction that shows $\mathsf{Pow}^n_{m,k} \not\subseteq \mathsf{Ch}^n_m$.*

Both separations (a) *and* (b) *cannot be achieved using occurrence obstructions, even for arbitrary k: for all m-partitions μ that satisfy $a_\mu(d[n]) > 0$ we have $\mathrm{mult}_\lambda(\mathbb{C}[\mathsf{Ch}^n_m]_{d'}) > 0$ in these settings.*

One would like to show that there are no occurrence obstructions in all cases (1), but this is wrong if n is not large enough with respect to m, see Prop. 3.11. Even for $m = 3$ or $m = 4$ ruling out occurrence obstructions as in (2) is done by a large-scale computer calculation which is only suitable for a finite case, but not for sequences as in (1). The papers [16, 8] rule out occurrence obstructions for families, but only in ranges where they would give very strong new algebraic circuit lower bounds, so that we expect it to be difficult to find multiplicity obstructions in those cases. Note also that [16, 8] are only dealing with *padded polynomials*, for which [17] guarantees λ to have a very restricted shape.

We expect multiplicity obstructions to be more powerful than occurrence obstructions in most cases relevant for geometric complexity theory, and Theorem 2.1 resolves the challenge of finding a setting in which the corresponding multiplicities and occurrences could actually be computed in a reasonable amount of time, while the setting is also involved enough so that a difference between occurrence obstructions and multiplicity obstructions could be witnessed.

▶ Remark 2.2. The partition $(n^2-2, n, 2)$ is known to be the type of one of Brill's classical set-theoretic equations for Ch^n_m, see [12].

3 Proof of the main theorem

The main theorem (Theorem 2.1) makes a statement about the finite situations $k = 4$, $n = 6$, $m = 3$, $d = 7$ and $k = 4$, $n = 7$, $m = 4$, $d = 8$, as well as the general situation $m \geq 3$, $n \geq 2$, $k = d = n + 1$. As a first step, in all these cases we show that

$$\mathrm{mult}_\lambda(\mathbb{C}[\mathsf{Pow}^n_{m,k}]_d) = a_\lambda(d[n]). \tag{3}$$

In the finite cases the following computer calculation suffices to prove (3).

▶ **Proposition 3.1.** $\mathrm{mult}_{(34,6,2)}(\mathbb{C}[\mathsf{Pow}^6_{3,4}]_7) = 8 = a_{(34,6,2)}(7[6])$ *and* $\mathrm{mult}_{(47,7,2)}(\mathbb{C}[\mathsf{Pow}^7_{3,4}]_8) = 11 = a_{(47,7,2)}(8[7])$.

Proof. The plethysm coefficient computations were performed with the LiE software. The rest is a small computer calculation completely analogous to the ones in [8, Sec. 6]. The details can be found in the full version of this paper. ◀

For the general situation the equality (3) is a consequence of the following result on power sums proved in [8, Prop. 3.2]:

▶ **Proposition 3.2.** *If λ is an m-partition of dn and $k \geq d$, then* $\mathrm{mult}_\lambda(\mathbb{C}[\mathsf{Pow}^n_{m,k}]_d) = a_\lambda(d[n])$.

As a second step we will use the following lemma for $\lambda = (n^2-2, n, 2)$.

▶ **Lemma 3.3** (see also [19, Sec. 9.2.3])**.** *Let λ be an m-partition and $n \geq m$. Then* $\mathrm{mult}_\lambda(\mathbb{C}[\mathsf{Ch}^n_m]_d) \leq a_\lambda(n[d])$.

Proof. Let $\mathsf{GL}_n(x_1 \cdots x_n) := \{g(x_1 \cdots x_n) \mid g \in \mathsf{GL}_n\} \subseteq \mathbb{A}^n_n$ denote the GL_n-orbit of $x_1 \cdots x_n$. We denote by $\overline{\mathsf{GL}_n(x_1 \cdots x_n)}$ the Zariski closure of this orbit, which equals its Euclidean closure by the same principles as in Section 2. Choose bases and embed $\mathbb{A}^n_m \subseteq \mathbb{A}^n_n$, so that Ch^n_m is the intersection of $\mathbb{A}^n_m$ and $\overline{\mathsf{GL}_n(x_1 \cdots x_n)}$. This implies (via argument analogous to that for the plethysm coefficient ([14, Lem. 4.3.2])) that the multiplicity of the irreducible GL_m-representation $\{\lambda\}_m$ in $\mathbb{C}[\mathsf{Ch}^n_m]_d$ equals the multiplicity of the irreducible GL_n-representation $\{\lambda\}_n$ in $\mathbb{C}[\overline{\mathsf{GL}_n(x_1 \cdots x_n)}]$. In other words $\mathrm{mult}_\lambda(\mathbb{C}[\mathsf{Ch}^n_m]_d) = \mathrm{mult}_\lambda(\mathbb{C}[\overline{\mathsf{GL}_n(x_1 \cdots x_n)}]_d)$. The vector space $\mathbb{C}[\overline{\mathsf{GL}_n(x_1 \cdots x_n)}]_d$ consists of exactly the restrictions of polynomials in $\mathbb{C}[\mathbb{A}^n_n]_d$ to

the orbit $\mathsf{GL}_n(x_1 \cdots x_n)$. The coordinate ring $\mathbb{C}[\mathsf{GL}_n(x_1 \cdots x_n)]$ is also graded and its homogeneous degree d part $\mathbb{C}[\mathsf{GL}_n(x_1 \cdots x_n)]_d$ consists of *all* homogeneous degree d regular functions on $\mathsf{GL}_n(x_1 \cdots x_n)$, in particular $\mathrm{mult}_\lambda(\mathbb{C}[\overline{\mathsf{GL}_n(x_1 \cdots x_n)}]_d) \leq \mathrm{mult}_\lambda(\mathbb{C}[\mathsf{GL}_n(x_1 \cdots x_n)]_d)$. The right-hand side can be understood via geometric invariant theory as follows (see [14, Sec. 3.4(A)]): $\mathrm{mult}_\lambda(\mathbb{C}[\mathsf{GL}_n(x_1 \cdots x_n)]_d) = \mathrm{mult}_{\lambda^*}(\mathbb{C}[\mathsf{GL}_n]_d^H)$, where $H = \{\mathrm{diag}(\alpha_1, \ldots, \alpha_n) \mid \prod_{i=1}^n \alpha_1 = 1\} \rtimes \mathfrak{S}_n \subseteq \mathsf{GL}_n$ is the stabilizer of $x_1 \cdots x_n$. The algebraic Peter-Weyl theorem (see e.g. [18, II.3.1 Satz 3], [11, Thm. 4.2.7], or [27, Ch. 7, 3.1 Thm.]) states that $\mathbb{C}[\mathsf{GL}_n] = \bigoplus_\lambda \{\lambda\} \otimes \{\lambda^*\}$ and we conclude $\mathrm{mult}_\lambda(\mathbb{C}[\mathsf{GL}_n]_d^H) = \dim\{\lambda\}^H$. There are several ways of seeing that $\dim\{\lambda\}^H = a_\lambda(n[d])$, see e.g. [19, Sec. 9.2.3] or [15, Prop. 3.3]. This proves the lemma. ◀

Now an argument using symmetric functions is used to prove the following theorem.

▶ **Theorem 3.4.** $a_{(n^2-2,n,2)}(n+1[n]) = 1 + a_{(n^2-2,n,2)}(n[n+1])$.

Theorem 3.4 is a corollary of more general results, see Corollary 4.4 in the appendix.

This finishes the proof that $(n^2-2, n, 2)$ is a multiplicity obstruction in all cases of Theorem 2.1.

No occurrence obstructions

To finish the proof of Theorem 2.1(2), it remains to show that there are no occurrence obstructions in the finite situation $n = 6$, $m = 3$ and $n = 7$, $m = 4$. We will primarily go into more detail for the first case and the second one will be proven similarly. We will do this by showing that

$$a_\mu(d[n]) > 0 \text{ implies } \mathrm{mult}_\mu(\mathbb{C}[\mathsf{Ch}_m^n]_d) > 0 \quad \text{for } n = 6,\ m = 3. \tag{4}$$

Note that this claim is independent of k. We start proving (4) by giving a complete classification of when $a_\mu(d[n]) > 0$ for the case $n = 6$, $m = 3$.

First, the following lemma states that for a few special μ the plethysm coefficient always vanishes.

▶ **Lemma 3.5.** *Let $\bar{\lambda} := (\lambda_2, \lambda_3, \ldots)$ denote λ without its first row. If λ is an m-partition of dn and $\bar{\lambda} \in \{(3,3), (3,1), (2,1), (1,1), (1)\}$, then $a_\lambda(d[n]) = 0$.*

Proof. This is proved by a finite calculation for all cases but $(3,3)$ as Thm 1.10(a) in [16]. Exactly the same calculation can be used to also prove the result for the additional partition $(3,3)$. ◀

For characterizing the set of all μ for which $a_\mu(d[n])$ is positive, we observe that they form a finitely generated *semigroup* and hence we only need to find the semigroup's generators:

$$\text{If } a_\mu(d[n]) > 0 \text{ and } a_\nu(d'[n]) > 0, \text{ then } a_{\mu+\nu}(d+d'[n]) > 0. \tag{5}$$

A detailed proof of (5) can be found for example in [1, Prop. 21.2.6].

▶ **Proposition 3.6.** *Define the set*

$X := \{(6),\ (6,6),\ (8,4),\ (10,2),\ (6,6,6),\ (8,6,4),\ (10,4,4),\ (9,6,3),\ (8,8,2),\ (10,6,2),\ (11,5,2),\ (10,7,1),\ (12,4,2),\ (11,6,1),\ (10,8),\ (14,2,2),\ (13,4,1),\ (13,5),\ (15,3),\ (8,8,8),\ (10,8,6),\ (11,7,6),\ (10,9,5),\ (11,8,5),\ (10,10,4),\ (12,7,5),\ (11,9,4),\ (13,6,5),\ (12,8,4),\ (11,10,3),\ (13,7,4),\ (12,9,3),\ (13,8,3),\ (12,10,2),\ (15,5,4),\ (14,7,3),\ (13,9,2),\ (13,10,1),\ (16,5,3),\ (15,7,2),\ (14,9,1),\ (17,4,3),\ (15,8,1),\ (15,9),\ (19,3,2),\ (18,5,1),\ (17,7),\ (10,10,10),\ (11,10,9),\ (12,10,8),\ (13,9,8),\ (12,11,7),\ (13,10,7),\ (14,9,7),\ (13,11,6),\ (15,8,7),\ (13,12,5),\ (16,7,7),\ (15,9,6),\ (14,11,5),\ (13,13,4),\ (15,10,5),\ (15,11,4),\ (14,13,3),\ (16,11,3),\ (15,13,2),\ (15,14,1),\ (17,13),\ (13,12,11),\ (14,11,11),\ (13,13,10),\ (15,11,10),\ (14,13,9),\ (16,11,9),\ (15,13,8),\ (15,14,7),\ (18,9,9),\ (15,15,6),\ (17,17,2),\ (18,17,1),\ (26,5,5),\ (15,14,13),\ (16,13,13),\ (15,15,12),\ (17,17,8),\ (18,15,15),\ (17,17,14),\ (25,23),\ (45,45)\}$.

Here we truncated trailing zeros from the 3-partitions. The set X is the set of generators of the semigroup of 3-partitions μ that have $a_\mu(d[6]) > 0$.

The proof of Proposition 3.6 proceeds in several steps.

A direct computation with the LiE software verifies $a_\mu(d[6]) > 0$ for all $\mu \in X \setminus \{(45,45)\}$. The case $d = 15$ runs into memory problems when using LiE. Other software such as Schur stops working when $d = 8$. We used the formula [32, Cor. 4.2.8] to verify $a_{(45,45)}(15[6]) > 0$.

We call the number of nonzero parts the *length* of a partition. We use a brute-force computer verification and a direct computation with LiE to show that for $d \leq 26$ every partition μ of length ≤ 2 with $a_\mu(d[6]) > 0$ is a sum of partitions from the set X. The same computation is done for all 3-partitions, but only up to $d \leq 14$. The following proposition states that these finite computations completely describe all cases.

▶ **Proposition 3.7.** *If λ is a 3-partition of $6d$, $d \geq 15$, and $\bar{\lambda} \notin \{(3,3),(3,1),(2,1),(1,1),(1)\}$, then λ is a sum of partitions from X.*

Proof. For $15 \leq d \leq 17$ we use a computer calculation to show that we can write every such partition λ as a sum of partitions from X. For $d > 17$ we prove this inductively by showing that we can write every 3-partition λ of $6d$ with $\bar{\lambda} \notin \{(3,3),(3,1),(2,1),(1,1),(1)\}$ as a sum of one of the partitions $(6),(6,6)$ or $(6,6,6)$ and a smaller λ' with again $\bar{\lambda'} \notin \{(3,3),(3,1),(2,1),(1,1),(1)\}$.

Let c_i denote the number of columns in λ with exactly i boxes for $i \in \{1,2,3\}$. Since we have at least 108 boxes in λ, the pigeonhole principle implies that at least one must be true: $c_1 \geq 6$, $c_2 \geq 10$ or $c_3 \geq 10$.

In the case $c_1 \geq 6$ we have $\lambda = \lambda' + (6)$ with λ' being a sum of elements from X since $\bar{\lambda'} = \bar{\lambda}$. In the case $c_2 \geq 10$ we have $\lambda = \lambda' + (6,6)$ with λ' being a sum of elements from X as $\lambda'_2 \geq 4$. In the case $c_3 \geq 10$ we have $\lambda = \lambda' + (6,6,6)$ with λ' being a sum of elements from X as $\lambda'_3 \geq 4$. ◀

This finishes the proof of Proposition 3.6.

To prove (4) it is sufficient (and necessary) to show that $\mathrm{mult}_\mu(\mathbb{C}[\mathsf{Ch}^n_m]_d) > 0$ for all $\mu \in X$, because a semigroup property analogous to (5) holds (the same proof applies, e.g. [1, Prop. 21.2.6]):

$$\text{If } \mathrm{mult}_\mu(\mathbb{C}[\mathsf{Ch}^n_m]_d) > 0 \text{ and } \mathrm{mult}_\nu(\mathbb{C}[\mathsf{Ch}^n_m]_{d'}) > 0, \text{ then } \mathrm{mult}_{\mu+\nu}(\mathbb{C}[\mathsf{Ch}^n_m]_{d+d'}) > 0. \tag{6}$$

If the length of μ is at most 2, we use the following general result.

▶ **Proposition 3.8.** *Let μ be a 3-partition of length at most 2. If $a_\mu(d[n]) > 0$, then $\mathrm{mult}_\mu(\mathbb{C}[\mathsf{Ch}^n_m]_d) > 0$.*

Proof. We use an inheritance result: If for a 2-partition μ we have $\mathrm{mult}_\mu(\mathbb{C}[\mathsf{Ch}^n_2]_d) > 0$, and ν is the 3-partition that arises from μ by adding a single 0, then $\mathrm{mult}_\nu(\mathbb{C}[\mathsf{Ch}^n_3]_d) > 0$. The proof is completely analogous to other inheritance results, see e.g. [14, Lemma 4.3.2 or Sec. 5.3]. Now for 2-partitions μ we have $a_\mu(d[n]) = \mathrm{mult}_\mu(\mathbb{C}[\mathsf{Ch}^n_2]_d)$, because every homogeneous polynomial in 2 variables decomposes as a product of homogeneous linear polynomials by the fundamental theorem of algebra, see also e.g. [19, Exa. 9.1.1.8]. This is how the Hermite reciprocity can be proved. An even simpler argument works if μ has length 1. ◀

We finish the proof of (4) by using a computer calculation to verify that for all 3-partitions $\mu \in X$ of length 3 we have $\mathrm{mult}_\mu(\mathbb{C}[\mathsf{Ch}^6_3]) > 0$, see Proposition 5.1.

This finishes the proof of Theorem 2.1(2a). The proof of Theorem 2.1(2b) is completely analogous as follows. Let $m = 4$, $n = 7$.

▶ **Lemma 3.9.** *Let* $\bar{\lambda} := (\lambda_2, \lambda_3, \ldots)$ *denote* λ *without its first row. If* λ *is an* m*-partition of* dn *and* $\bar{\lambda} \in Y$ *for* $Y := \{(1), (1,1), (1,1,1), (2,1), (2,1,1), (2,2,1), (3,1), (3,1,1), (3,2,1), (3,3), (3,3,1), (3,3,2), (3,3,3), (4,1,1), (4,3,3), (5,1,1), (5,5,5), (6,1,1)\}$*, then* $a_\lambda(d[n]) = 0$.

Proof. This is proven exactly like Lemma 3.5. ◀

The semigroup of 4-partitions λ that have $a_\lambda(d[7]) > 0$ has 948 generators, listed in the full version of this paper. They form a set that we call X.

We again use a direct computation with the LiE software to verify $a_\mu(d[7]) > 0$ for all $\mu \in X \setminus \{(49,49), (24,24,23,23)\}$. For both the remaining partitions $\mu \in \{(49,49), (24,24,23,23)\}$ we prove $\mathrm{mult}_\mu(\mathbb{C}[\mathsf{Ch}_4^7]_d) > 0$ using our computer calculations which also implies $a_\mu(d[7]) > 0$.

To prove those are all the generators we use the following proposition which is proved completely analogously to Proposition 3.7.

▶ **Proposition 3.10.** *If* λ *is a* 4*-partition of* $7d$*,* $d \geq 14$*, and* $\bar{\lambda} \notin Y$*, then* λ *is a sum of partitions from* X*.*

For the next finite case ($n = 7$, $k = d = 8$, $m = 5$) we reached the computational limit of our implementation. Here we were able to find 5016 generating partitions of the semigroup of 4-partitions μ that have $a_\mu(d[7]) > 0$. Unfortunately these do not generate everything excluding the exceptions yet. We were able to verify for 5000 generating partitions μ that $\mathrm{mult}_\mu(\mathbb{C}[\mathsf{Ch}_m^n]_d) > 0$. For the remaining ones, we used up to 200 GB of RAM, but this was not sufficient.

Some occurrence obstructions

As we degenerate the parameter settings and let n get closer to m, multiplicity obstructions tend to become occurrence obstructions. More precisely, for $m = 3$ and values of $n < 6$, and for $(m, n) = (4, 6)$, some multiplicity obstructions are actually also occurrence obstructions, as the following proposition shows.

▶ **Proposition 3.11.** *The following partitions give occurrence obstructions that show* $\mathsf{Pow}_{m,d}^n \not\subseteq \mathsf{Ch}_m^n$.

m	n	λ	d	$a_\lambda(d[n])$	$a_\lambda(n[d])$
3	2	$(2,2,2)$	3	1	0
3	3	$(7,3,2)$	4	1	0
3	4	$(11,9,8)$	7	1	0
3	5	$(12,9,9)$	6	1	0
4	6	$(14,14,13,13)$	9	11	0

Proof. The plethysm coefficient computations were performed with the LiE software. Lemma 3.3 implies that $\mathrm{mult}_\lambda(\mathbb{C}[\mathsf{Ch}_m^n]_d) \leq a_\lambda(n[d]) = 0$. Proposition 3.2 implies $\mathrm{mult}_\lambda(\mathbb{C}[\mathsf{Pow}_{m,d}^n]_d) > 0$. ◀

See [4, Prop. 4] for additional occurrence obstructions in the case $n = 3$.

4 Plethysm inequalities

We are interested in the plethysm coefficients $a_\lambda(d[m])$ for certain values of λ and d, m. Here we compute such values for infinite families of parameters and in particular, prove Theorem 3.4.

We will work over the ring of symmetric functions Λ, defined as the ring of formal power series (in finitely or infinitely many variables) which are invariant under any transposition of the variables. For the definitions and main identities see e.g. [29]. Plethysms of symmetric functions are described also there in Appendix 2 of Chapter 7, here we review the necessary definitions.

The characters of the irreducible GL_r–module W_λ are the Schur functions $s_\lambda(x_1, \ldots, x_r)$, where $x_1, \ldots, x_r$ correspond to the eigenvalues of the conjugacy class representative from GL_r. Their combinatorial interpretation is as the generating function over all semi-standard Young tableaux with entries $1, ..., r$, but we will use certain determinantal formulas as described below. The complete homogeneous symmetric functions h_ℓ are defined as $s_{(\ell)}$ and are the characters of the Sym^ℓ module. The $\mathrm{Sym}^d(\mathrm{Sym}^n(\mathbb{C}^r))$ module is obtained as the composition of the two representations. The image in $\mathrm{Sym}^n(\mathbb{C}^r)$ of a diagonal matrix from GL_r with entries (i.e. eigenvalues) $x_1, \ldots, x_r$ on the diagonal has eigenvalues all the $N := \binom{n+r-1}{r-1}$ degree n monomials in $x_1, \ldots, x_r$. Hence, the character of the representation $\mathrm{Sym}^d(\mathrm{Sym}^n(\mathbb{C}^r))$ of GL_N can be obtained by evaluating the character h_d of Sym^d at the monomials, i.e. the eigenvalues above. This gives us the definition of the *symmetric function plethysm* $h_d[h_n(x_1, \ldots, x_r)]$, that is, the evaluation of h_d on the variables consisting of all degree n monomials, i.e. $h_d[h_n(x_1, \ldots, x_r)] := h_d(x_1^n, x_1^{n-1}x_2, x_1^{n-1}x_3, \ldots, x_1^{\alpha_1} \cdots x_r^{\alpha_r}, \ldots)$, where $\alpha = (\alpha_1, \ldots, \alpha_r)$ runs over all compositions of n.

In general, knowing the character of a representation contains all the information to obtain the multiplicities of the irreducible decomposition via the inner product of characters. As the Schur functions s_λ are the irreducible characters for GL_r, the inner product is equivalent to an inner product in the ring Λ, where $\{s_\lambda\}_\lambda$ is an orthonormal basis. In other words, the multiplicity of the Weyl module of weight λ is given by the multiplicity of the Schur function s_λ in the expansion of $h_d[h_m]$. We will now compute this via the inner product in the ring Λ of symmetric functions, using some basic properties of this ring as found in [29] and [21].

We have that $a_\lambda(d[n])$ is the multiplicity of $\{\lambda\}$ in $\mathrm{Sym}^d\mathrm{Sym}^n$, translated into characters this is also the coefficient at s_λ of the expansion of $h_d[h_n]$ in Schur function. By their orthonormalitiy, this is the same as

$$a_\lambda(d[n]) = \langle s_\lambda, h_d[h_n] \rangle \tag{7}$$

We now invoke various symmetric function identities in order to compute the above inner product. The Schur functions s_λ can be expressed via the Jacobi-Trudi formula (see again [29, Ch. 7]) as a signed sums of homogeneous symmetric functions, namely

$$s_\lambda = \det\left[h_{\lambda_i - i + j}\right]_{i,j=1}^{\ell(\lambda)}, \tag{8}$$

the inner product (7) can then be computed via a signed sum of inner products of the form $\langle h_\mu, h_d[h_n] \rangle$. We remark that the orthogonal dual basis for the complete homogeneous symmetric functions is the monomial symmetric functions, i.e. $\langle h_\mu, m_\nu \rangle = \delta_{\mu,\nu}$, so we need to express $h_d[h_n]$ in terms of the monomial symmetric functions, defined by

$$m_\nu(x_1, \ldots, x_r) := \sum_{\sigma \in S_r(\nu)} x_1^{\nu_{\sigma(1)}} x_2^{\nu_{\sigma(2)}} \cdots x_r^{\nu_{\sigma(r)}},$$

where the sum ranges over all distinct permutations of $(\nu_1, \nu_2, \ldots, \nu_r)$ and ν is completed with 0s to the length r. Since he monomial symmetric functions form a basis for Λ, we can expand any symmetric function in it uniquely. Let $h_d[h_m] = \sum_\nu c_\nu m_\nu$, for some constants c_ν (i.e. the coefficients in this expansion). Since each m_ν has a unique leading monomial (in the lexicographic order) $x_1^{\nu_1}x_2^{\nu_2}\cdots$, finding c_ν is equivalent to extracting the coefficient at the single monomial $x_1^{\nu_1}\cdots$ from the monomial expansion of the corresponding symmetric function as a polynomial, i.e. $c_\nu = (x_1^{\nu_1}x_2^{\nu_2}\ldots)@h_d[h_n(x_1, x_2, \ldots)]$, where to avoid confusion with the plethysm notation we denote by $(X)@f$ the coefficient of the monomial X in the monomial expansion of the polynomial f.

Let ν be a partition of length ℓ. By the above remarks we need to consider only the truncated expansion $h_d[h_n(x_1, \ldots, x_\ell)]$ as only the monomials in $x_1, \ldots, x_\ell$ will be relevant.

We have the following formula for the h's, see e.g. [29]:

$$h_N(x_1, \ldots, x_r) = \sum_{(b):b_1+b_2+\cdots=N} x_1^{b_1}x_2^{b_2}\cdots,$$

where $(b) = (b_1, b_2, \ldots, b_r)$ runs over all (weak) compositions of N. Hence, assuming some total ordering for compositions α^i of n, we have

$$h_d[h_n(x_1, \ldots, x_r)] = h_d[\ldots, x^{\alpha^i}, \ldots] = \sum_{(b):|b|=d} x^{\sum_i b_i\alpha^i}.$$

Thus for the coefficients c_ν we have:

$$c_\nu(d, n) := (x^\nu)@h_d[h_n] = \langle h_\nu, h_d[h_n]\rangle = \#\{(b) : |b| = d, \sum_i b_i\alpha^i = \nu\} \tag{9}$$

By the Jacobi-Trudi identity (8) this gives a formula for computing the plethysm coeff. as

$$a_\lambda(d[n]) = \langle \det\left[h_{\lambda_i-i+j}\right]_{i,j=1}^{\ell(\lambda)}, h_d[h_n]\rangle = \sum_{\pi\in S_{\ell(\lambda)}} sgn(\pi)c_{\lambda+\pi-(1,2,\ldots)}(d, n), \tag{10}$$

where the permutations π are viewed as vectors with entries $1, 2, \ldots, \ell(\lambda)$

We now turn towards the proof of Theorem 3.4 and consider s_λ for $\lambda = (\lambda_1, \lambda_2, 2)$ for some $k \geq 2$. By the Jacobi-Trudi identity (10) we need to compute only c_ν for ν having at most 3 parts, with $\nu_3 = 0, 1, 2$. Let $p_r(a, b)$ denote the number of partitions of r which fit inside an $a \times b$ rectangle, it's generating function is the q-binomial coefficient (see [31]): $\binom{a+b}{a}_q = \frac{(1-q)\cdots(1-q^{a+b})}{(1-q)\cdots(1-q^a)(1-q)\cdots(1-q^b)} = \sum_{r=0}^{ab} p_r(a, b)q^r$

▶ **Proposition 4.1.** *We have the following generating function identities for $c_\nu(d, n)$, where $\ell(\nu) \leq 3$ and $\nu_3 \leq 2$:*

$$c_{(L,k,2)} = (q^k)@\left(\tbinom{n}{1}_q\tbinom{n+d-2}{n}_q + \tbinom{n-1}{1}_q\tbinom{n+d-1}{n}_q + q\tbinom{n}{2}_q\tbinom{n+d-2}{n}_q\right)$$
$$c_{(L,k,1)} = (q^k)@\tbinom{n}{1}_q\tbinom{n+d-1}{n}_q$$
$$c_{(L,k,0)} = (q^k)@\tbinom{n+d}{n}_q = p_k(n, d)$$

Proof. By formula (9), we have $c_{(L,k,0)} = \#\{(b) : |b| = d, \sum b_i\alpha^i = (L, k)\}$.

Hence, the only α^i involved are of the form $\alpha^i = (n - a_i, a_i)$, and after renumerating, we can assume $a_i = i$. So we are counting compositions b of d, s.t. $\sum_i b_i i = k$ for $i = 0 \ldots n$. This is exactly the same as specifying an integer partition γ of k by the number of its parts, i.e. $\gamma = (0^{b_0}, 1^{b_1}, \ldots, n^{b_n})$, such that $b_0 + \cdots + b_n = d$. These restrictions are equivalent to γ fitting inside an $n \times d$ box, and the number of such γ is exactly $p_k(d, n)$.

Next, when the second part in ν is 1, we have the following. Since $\nu_3 = 1$, the condition $\sum_i b_i\alpha_3^i = \nu_3 = 1$ implies that there is a single i, such that $b_i\alpha_3^i \neq 0$, and in fact must be 1, so $b_i = \alpha_3^i = 1$. After renumeration, we can assume that $i = 0$ (for separation purposes) with $b_0 = \alpha_3^0 = 1$ and $\alpha^0 = (n-1-r, r, 1)$ for $r = 0 \dots n-1$. For the remaining bs and αs we have $\sum_i b_i\alpha^i = (L,k) - (n-1-r, r) = (L+r-n+1, k-r)$ with $b_1 + \dots = d-1$, and $|\alpha^i| = n$. This number is now, by the previous case, $(q^{k-r})@\binom{n+d-1}{n}_q$. The total number is thus

$$c_{(L,k,1)} = \sum_{r=0}^{n-1}(q^{k-r})@\tbinom{n+d-1}{n}_q = (q^k)@\sum_{r=0}^{n-1} q^r\tbinom{n+d-1}{n}_q = (q^k)@\tbinom{n}{1}_q\tbinom{n+d-1}{n}_q.$$

Finally, when $\nu_3 = 2$ we have the following two distinct options:

Either there is an index i, such that $b_i\alpha_3^i = \nu_3 = 2$, or $i < j$ with $b_i\alpha_3^i = 1$ and $b_j\alpha_3^j = 1$.

In the first case we have $b_i\alpha_i^3 = 2$ – either $b_i = 2$, in which case $\alpha^i = (n-1-r, 1)$ and the rest of the b's sum to $d - b_i = d-2$, which brings us to the previous case (of $(L,k,1)$), so the number is

$$(q^k)@\tbinom{n}{1}_q\tbinom{n+d-2}{n}_q.$$

Otherwise, $b_i = 1$ and $\alpha_3^i = 2$. As in the case $\nu_3 = 1$, let $i = 0$ and $\alpha^0 = (n-2-r, r, 2)$, $b_0 = 1$, so we are looking for the number of $(b_1, \dots)$ with $|b| = d-1$ and such that $\sum_i b_i\alpha^i = (L-n+r+2, k-r)$ for all possible $r = 0, \dots, n-2$. So this is

$$\sum_{r=0}^{n-2}(q^{k-r})@\tbinom{n+d-1}{n}_q = (q^k)@\tbinom{n-1}{1}_q\tbinom{n+d-1}{n}_q$$

Last, when there are $i < j$ with $b_i\alpha_3^i = 1$ and $b_j\alpha_3^j = 1$, let $i = -1$, $j = 0$ (again, renumerating for simplicity), with $\alpha^{-1} = (n-1-r_1, r_1, 1)$ and $\alpha^0 = (n-1-r_2, r_2, 1)$ with $0 \le r_1 < r_2 \le n-1$. We thus have for the remaining α and bs that $b_1 + \dots = d-2$, and $\sum_i b_i\alpha^i = (L-(n-1-r_1)-(n-1-r_2), k-r_1-r_2)$. By the first case, this is $(q^{k-r_1-r_2})@\binom{n+d-2}{n}_q$. Summing over all possible $0 < r_1 < r_2 \le n-1$, we have

$$\begin{aligned}(q^k)@\sum_{0\le r_1<r_2\le n-1} q^{r_1+r_2}\binom{n+d-2}{n}_q &= (q^k)@q\sum_{0\le r_1\le r_2-1\le n-2} q^{r_1+(r_2-1)}\binom{n+d-2}{n}_q \\ &= (q^k)@q\binom{n-2+2}{2}_q\binom{n+d-2}{n}_q,\end{aligned}$$

where the last identity follows from interpreting (r_2-1, r_1) as a partition in the $2 \times n-2$ rectangle. Summing over all the cases considered here, we get the desired total coefficient. ◀

▶ **Proposition 4.2.** *The plethysm coefficient for $\lambda = (L, r, 2)$ is equal to*

$$a_\lambda(d[n]) = (q^{r+1})@\left(\tbinom{n+d-2}{n}_q\tfrac{q(1-q^n)(1-q^2+q-q^n)}{1-q^2} + \tbinom{n+d-1}{n}_q(q^{n+1}-1) + (1-q)\tbinom{n+d}{n}_q\right)$$

Proof. Following equation (10), we have that
$a_\lambda(d[n]) = c_{(L,r,2)} - c_{(L,r+1,1)} - c_{(L+1,r-1,2)} + c_{(L+1,r+1,0)} + c_{(L+2,r-1,1)} - c_{(L+2,r,0)}$.
Substituting the formulas for the c's from Proposition 4.1, and observing that $(q^{r+j})@f = (q^r)@q^{-j}f$ for any j, we have that

$$\begin{aligned}a_\lambda(d[n]) = (q^{r+1})@\big((q-q^2)\big(\tbinom{n}{1}_q\tbinom{n+d-2}{n}_q + \tbinom{n-1}{1}_q\tbinom{n+d-1}{n}_q + q\tbinom{n}{2}_q\tbinom{n+d-2}{n}_q\big) \\ +(q^2-1)\tbinom{n}{1}_q\tbinom{n+d-1}{n}_q + (1-q)\tbinom{n+d}{n}_q\big)\end{aligned}$$

Simplifying the above expression by grouping terms for the same binomial coefficients together we obtain
$a_\lambda(d[n]) = (q^{r+1})@\left(\tbinom{n+d-2}{n}_q\tfrac{q(1-q^n)(1-q^2+q-q^n)}{1-q^2} + \tbinom{n+d-1}{n}_q(q^{n+1}-1) + (1-q)\tbinom{n+d}{n}_q\right)$. ◀

▶ **Proposition 4.3.** *Let $\lambda = (L, r, 2)$. We have that*

$$a_\lambda(d[n]) - a_\lambda(n[d]) = (q^r) \mathbin{@} \tbinom{n+d-2}{n-1}_q (q^n - q^d) \tfrac{(1-q^{d-1})(1-q^{n-1})}{(1-q^d)(1-q^n)}.$$

Proof. Set $[a]!_q := (1-q)\cdots(1-q^a)$, a variant of the usual factorial q-analogue but multiplied by $(1-q)^a$, and consider the desired difference via the formula in Proposition 4.2:

$$\begin{aligned} a_\lambda(d[n]) - a_\lambda(n[d]) = (q^{r+1}) \mathbin{@} & \\ \Big\{ & \Big(\tbinom{n+d-2}{n}_q \tfrac{q(1-q^n)(1-q^2+q-q^n)}{1-q^2} + \tbinom{n+d-1}{n}_q (q^{n+1}-1) \\ & - \tbinom{n+d-2}{d}_q \tfrac{q(1-q^d)(1-q^2+q-q^d)}{1-q^2} - \tbinom{n+d-1}{d}_q (q^{d+1}-1) \Big) \\ & = \tfrac{[n+d-2]!_q}{[n-2]!_q[d-2]!_q} \tfrac{q(q^{n-1}-q^{d-1})}{(1-q^{n-1})(1-q^{d-1})} - \tfrac{[n+d-1]!_q}{[n-1]!_q[d-1]!_q} \tfrac{(1-q)(q^n-q^d)}{(1-q^n)(1-q^d)} \\ & = \tbinom{n+d-2}{n-1}_q (q^n - q^d) \left(1 - \tfrac{(1-q^{n+d-1})(1-q)}{(1-q^n)(1-q^d)}\right) \\ & = \tbinom{n+d-2}{n-1}_q (q^n - q^d) \tfrac{q(1-q^{d-1})(1-q^{n-1})}{(1-q^d)(1-q^n)} \Big\} \end{aligned}$$

Finally, observe that the RHS is a polynomial divisible by q, so the coefficient at q^{r+1} is the same as the coefficient at q^r after dividing by q. ◀

We are now ready to prove Theorem 3.4 as a Corollary of the above computations:

▶ **Corollary 4.4.** *[Theorem 3.4] Let $d = n+1$ and $\lambda = (n^2+n-2-r, r, 2)$. Then $a_\lambda((n+1)[n]) - a_\lambda(n[n+1]) \geq 0$, with*

$$a_\lambda((n+1)[n]) - a_\lambda(n[n+1]) = \begin{cases} 0, & \text{when } r < n, \\ 1, & \text{when } r = n, \\ > 0, & \text{when } r > n \text{ and } n \geq 7, \end{cases}$$

with the exception in the last case when $n = 8$, and $r = 35$ when $a_{(35,35,2)}(9[8]) = a_{(35,35,2)}(8[9])$.

Proof. Then by the Proposition 4.3 we have

$$\begin{aligned} a_\lambda(n+1[n]) - a_\lambda(n[n+1]) &= (q^r) \mathbin{@} \tbinom{2n-1}{n-1}_q (q^n - q^{n+1}) \tfrac{(1-q^n)(1-q^{n-1})}{(1-q^{n+1})(1-q^n)} \\ &= (q^r) \mathbin{@} \tbinom{2n-1}{n-1}_q (q^n - q^{n+1}) \tfrac{(1-q^{n-1})}{1-q^{n+1}} = (q^r) \mathbin{@} \tbinom{2n-1}{n-2}_q (q^n - q^{n+1}) \end{aligned}$$

The last line follows by absorbing the fraction into the q-binomial coefficient. It is now evident, that since the q-binomial coefficient expands into a polynomial of q (with coefficients given by $p_*(n-2, n+1)$), multiplying it with q^n or q^{n+1} gives two polynomials whose lowest order terms are q^n and q^{n+1} respectively. So if $r < n$, there is no term of such degree, and the coefficient is 0. when $r = n$ we see that such term can only come from the first polynomial's first (lowest order) term, which is exactly q^n since $\tbinom{2n-1}{n-2}_q q^n = q^n(1 + q + 2q^2 + \cdots) = q^n + O(q^{n+1})$. Therefore we obtain the case $r = n$.

Let now $r > n$, and set $r = n + k + 1$ for some $k \geq 0$. We have that

$$\begin{aligned} a_\lambda((n+1)[n]) - a_\lambda(n[n+1]) &= (q^{k+1}) \mathbin{@} \tbinom{2n-1}{n-2}_q - (q^k) \mathbin{@} \tbinom{2n-1}{n-2}_q \\ &= p_{k+1}(n+1, n-2) - p_k(n+1, n-2) \\ &= g((n^2-n-3-k, k+1), (n+1)^{n-2}, (n+1)^{n-2}) > 0, \end{aligned}$$

where g denotes the Kronecker coefficient for the symmetric group $\mathfrak{S}_n$ for the 3 given partititons, and the last identity and the strict positivity are shown to hold for $n \geq 9$ in [26], and the other cases are verified by direct expansion of the q-binomial coefficients. In particular, we have that $p_{26}(9,6) = 227 = p_{27}(9,6)$ which gives the only exceptional 0 plethysm. ◀

5 Computer calculations

▶ **Proposition 5.1.** *If X is defined as in Proposition 3.6, then for all $\mu \in X$ of length 3 we have* $\mathrm{mult}_\mu(\mathbb{C}[\mathsf{Ch}_3^6]) > 0$.

If X is the set of generators of the semigroup of 4-partitions μ that have $a_\mu(d[7]) > 0$ (see full version of this paper), then for all $\mu \in X$ we have $\mathrm{mult}_\mu(\mathbb{C}[\mathsf{Ch}_4^7]) > 0$.

Proof. Proposition 5.1 is proved by a computer calculation that is a refinement and speedup of the computation performed in [9]. Indeed, a run of the method from [9] would take significantly too long to prove Proposition 5.1 in any reasonable time. Our new method makes extensive use of memory resources, while the method from [9] uses almost no memory. The description of the computation can be found in the full version of this paper. ◀

References

1 Markus Bläser and Christian Ikenmeyer. Introduction to geometric complexity theory. lecture notes, summer 2017 at Saarland University, version July 25, 2018, `http://people.mpi-inf.mpg.de/~cikenmey/teaching/summer17/introtogct/gct.pdf`, 2018.

2 Emmanuel Briand, Rosa Orellana, and Mercedes Rosas. Reduced Kronecker coefficients and counter-examples to Mulmuley's strong saturation conjecture SH. *Computational Complexity*, 18:577–600, 2009.

3 Peter Bürgisser, Michael Clausen, and M. Amin Shokrollahi. *Algebraic complexity theory*, volume 315 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1997. With the collaboration of Thomas Lickteig.

4 Peter Bürgisser, Jesko Hüttenhain, and Christian Ikenmeyer. Permanent versus determinant: not via saturations. *Proceedings of the American Mathematical Society*, 145(3):1247–1258, 2017.

5 Peter Bürgisser and Christian Ikenmeyer. The complexity of computing Kronecker coefficients. In *FPSAC 2008, Valparaiso-Viña del Mar, Chile, DMTCS proc. AJ*, pages 357–368, 2008.

6 Peter Bürgisser and Christian Ikenmeyer. Geometric Complexity Theory and Tensor Rank. *Proceedings 43rd Annual ACM Symposium on Theory of Computing 2011*, pages 509–518, 2011.

7 Peter Bürgisser and Christian Ikenmeyer. Explicit Lower Bounds via Geometric Complexity Theory. *Proceedings 45th Annual ACM Symposium on Theory of Computing 2013*, pages 141–150, 2013.

8 Peter Bürgisser, Christian Ikenmeyer, and Greta Panova. No occurrence obstructions in geometric complexity theory. *Journal of the AMS*, 32:163–193, 2019. An earlier version was presented at the IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS) 2016 in New Brunswick, New Jersey.

9 Man-Wai Cheung, Christian Ikenmeyer, and Sevak Mkrtchyan. Symmetrizing tableaux and the 5th case of the Foulkes conjecture. *Journal of Symbolic Computation*, 80(3):833–843, 2016. `doi:10.1016/j.jsc.2016.09.002`.

10 W. Fulton. *Young tableaux*, volume 35 of *London Mathematical Society Student Texts*. Cambridge University Press, Cambridge, 1997.

11 Roe Goodman and Nolan R. Wallach. *Symmetry, representations, and invariants*, volume 255 of *Graduate Texts in Mathematics*. Springer, Dordrecht, 2009.

12 Yonghui Guan. Brill's equations as a GL(V)-module. *Linear Algebra and its Applications*, 548:273–292, 2018. doi:10.1016/j.laa.2018.02.026.

13 Charles Hermite. Sur la théorie des fonctions homogenes à deux indéterminées. *Cambridge and Dublin Mathematical Journal*, 9:172–217, 1854.

14 Christian Ikenmeyer. *Geometric Complexity Theory, Tensor Rank, and Littlewood-Richardson Coefficients*. PhD thesis, Institute of Mathematics, University of Paderborn, 2012. URL: http://nbn-resolving.de/urn:nbn:de:hbz:466:2-10472.

15 Christian Ikenmeyer. GCT and symmetries. unpublished lecture notes, version from January 29, 2018.

16 Christian Ikenmeyer and Greta Panova. Rectangular Kronecker Coefficients and Plethysms in Geometric Complexity Theory. *Advances in Mathematics*, 319:40–66, 2017. An earlier version was presented at the IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS) 2016 in New Brunswick, New Jersey.

17 Harlan Kadish and J. M. Landsberg. Padded Polynomials, Their Cousins, and Geometric Complexity Theory. *Communications in Algebra*, 42(5):2171–2180, 2014. doi:10.1080/00927872.2012.758268.

18 Hanspeter Kraft. *Geometrische Methoden in der Invariantentheorie*. Friedr. Vieweg und Sohn Verlagsgesellschaft, Braunschweig, 1985.

19 J. M. Landsberg. *Geometry and Complexity Theory*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2017. doi:10.1017/9781108183192.

20 Joseph Landsberg. *Tensors: Geometry and Applications*, volume 128 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, Rhode Island, 2011.

21 I. G. Macdonald. *Symmetric functions and Hall polynomials*. Oxford Mathematical Monographs. The Clarendon Press, Oxford University Press, New York, second edition, 1995. With contributions by A. Zelevinsky, Oxford Science Publications.

22 K.D. Mulmuley and M. Sohoni. Geometric Complexity Theory. I. An approach to the P vs. NP and related problems. *SIAM J. Comput.*, 31(2):496–526 (electronic), 2001.

23 K.D. Mulmuley and M. Sohoni. Geometric Complexity Theory. II. Towards explicit obstructions for embeddings among class varieties. *SIAM J. Comput.*, 38(3):1175–1206, 2008.

24 D. Mumford. *Algebraic geometry. I: Complex projective varieties*. Classics in mathematics. Springer-Verlag, Berlin, 1995. Reprint of the 1976 edition in Grundlehren der mathematischen Wissenschaften, vol. 221.

25 Hariharan Narayanan. On the complexity of computing Kostka numbers and Littlewood-Richardson coefficients. *J. Algebraic Combin.*, 24(3):347–354, 2006.

26 Igor Pak and Greta Panova. Strict unimodality of q-binomial coefficients. *C. R. Math. Acad. Sci. Paris*, 351(11-12):415–418, 2013. doi:10.1016/j.crma.2013.06.008.

27 Claudio Procesi. *Lie groups*. Universitext. Springer, New York, 2007. An approach through invariants and representations.

28 Ramprasad Saptharishi. A survey of lower bounds in arithmetic circuit complexity. , version 3.1.4, 2017. URL: https://github.com/dasarpmar/lowerbounds-survey.

29 Richard P. Stanley. *Enumerative combinatorics. Vol. 2*, volume 62 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 1999. With a foreword by Gian-Carlo Rota and appendix 1 by Sergey Fomin.

30 Richard P. Stanley. Positivity problems and conjectures in algebraic combinatorics. In *Mathematics: frontiers and perspectives*, pages 295–319. Amer. Math. Soc., Providence, RI, 2000.

31 Richard P. Stanley. *Enumerative combinatorics. Vol. 1*. Cambridge University Press, Cambridge, 2011. second edition.

32 Bernd Sturmfels. *Algorithms in Invariant Theory*. Texts & Monographs in Symbolic Computation. Springer, 2008.

The Arboricity Captures the Complexity of Sampling Edges

Talya Eden
Tel Aviv University, Tel Aviv, Israel
talyaa01@gmail.com

Dana Ron
Tel Aviv University, Tel Aviv, Israel
danaron@tau.ac.il

Will Rosenbaum
Max Planck Institute for Informatics, Saarbrücken, Germany
will.rosenbaum@gmail.com

Abstract

In this paper, we revisit the problem of sampling edges in an unknown graph $G = (V, E)$ from a distribution that is (pointwise) almost uniform over E. We consider the case where there is some a priori upper bound on the arboriciy of G. Given query access to a graph G over n vertices and of average degree d and arboricity at most α, we design an algorithm that performs $O\left(\frac{\alpha}{d} \cdot \frac{\log^3 n}{\varepsilon}\right)$ queries in expectation and returns an edge in the graph such that *every* edge $e \in E$ is sampled with probability $(1 \pm \varepsilon)/m$. The algorithm performs two types of queries: degree queries and neighbor queries. We show that the upper bound is tight (up to poly-logarithmic factors and the dependence in ε), as $\Omega\left(\frac{\alpha}{d}\right)$ queries are necessary for the easier task of sampling edges from any distribution over E that is close to uniform in total variational distance. We also prove that even if G is a tree (i.e., $\alpha = 1$ so that $\frac{\alpha}{d} = \Theta(1)$), $\Omega\left(\frac{\log n}{\log\log n}\right)$ queries are necessary to sample an edge from any distribution that is pointwise close to uniform, thus establishing that a poly$(\log n)$ factor is necessary for constant α. Finally we show how our algorithm can be applied to obtain a new result on approximately counting subgraphs, based on the recent work of Assadi, Kapralov, and Khanna (ITCS, 2019).

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Approximation algorithms analysis; Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Sketching and sampling

Keywords and phrases sampling, graph algorithms, arboricity, sublinear-time algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.52

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1902.08086`.

Funding *Talya Eden*: Supported by the Azrieli fellowship program for graduate students, by the Sephora Scholarship and by the Weinstein Graduate Studies Prize.
Dana Ron: Partially supported by the Israel Science Foundation grant No. 1146/18.

1 Introduction

Let $G = (V, E)$ be a graph over n vertices and m edges. We consider the problem of sampling an edge in G from a pointwise-almost-uniform distribution over E. That is, for each edge $e \in E$, the probability that e is returned is $(1 \pm \varepsilon)/m$, where ε is a given approximation parameter. An algorithm for performing this task has random access to the vertex set $V = \{1, \ldots, n\}$ and can perform queries to G. The allowed queries are (1) *degree queries*

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 52; pp. 52:1–52:14

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

denoted $\mathsf{deg}(v)$ (what is the degree, $d(v)$, of a given vertex v) and (2) *neighbor queries* denoted $\mathsf{nbr}(v,i)$ (what is the i^{th} neighbor of v).[1] We refer to this model as *the uniform vertex sampling model*.

Sampling edges almost uniformly is a very basic sampling task. In particular it gives the power to sample vertices with probability approximately proportional to their degree, which is a useful primitive. Furthermore, there are sublinear algorithms that are known to work when given access to uniform edges (e.g., [1, 2]) and can be adapted to the case when the distribution over the edges is almost uniform (see Section 1.4 for details). An important observation is that in many cases it is crucial that the sampling distribution is pointwise-close to uniform rather than close with respect to the Total Variation Distance (henceforth TVD) – see the discussion in [15, Sec. 1.1].

Eden and Rosenbaum [15] recently showed that $\Theta^*(\sqrt{m}/d)$ queries are both sufficient and necessary for sampling edges almost uniformly, where $d = 2m/n$ denotes the average degree in the graph. (We use the notation O^* to suppress factors that are polylogarithmic in n and polynomial in $1/\varepsilon$.) The instances for which the task is difficult (i.e., for which $\Omega(\sqrt{m}/d)$ queries are necessary), are characterized by having very dense subgraphs, i.e., a subgraph with average degree $\Theta(\sqrt{m})$. Hence, a natural question is whether it is possible to achieve lower query complexity when some a apriori bound on the density of subgraphs is known. A well studied measure for bounded density "everywhere" is the graph *arboricity* (see Definition 1.2 below). Indeed there are many natural families of graphs that have bounded arboricity such as graphs of bounded degree, bounded treewidth or bounded genus, planar graphs, graphs that exclude a fixed minor and many other graphs. In the context of social networks, preferential attachment graphs and additional generative models exhibit bounded arboricity [3, 6, 5], and this has also been empirically validated for many real-world graphs [18, 16, 22].

We describe a new algorithm for sampling edges almost uniformly whose runtime is $O^*(\alpha/d)$ where α is an upper bound on the arboricity of G. In the extremal case that $\alpha = \Theta(\sqrt{m})$, the runtime of our algorithm is the same as that of [15] (up to poly-log factors). For smaller α, our algorithm is strictly faster. In particular for $\alpha = O(1)$, the new algorithm is *exponentially* faster than that of [15]. We also prove matching lower bounds, showing that for all ranges of α, our algorithm is query-optimal, up to polylogarithmic factors and the dependence in $1/\varepsilon$.

Furthermore, while not as simple as the algorithm of [15], our algorithm is still easy to implement and does not incur any large constants in the query complexity and running time, thus making in it suitable for practical applications.

1.1 Problem definition

In order to state our results precisely, we define the notion of pointwise-closeness of probability distributions (cf. [15]) and arboricity of a graph.

▶ **Definition 1.1.** *Let D be a fixed probability distribution on a finite set X. We say that a probability distribution $\widehat{D}$ is **pointwise ε-close** to D if for all $x \in X$,*

$$\left|\widehat{D}(x) - D(x)\right| \leq \varepsilon D(x)\,, \quad \textit{or equivalently} \quad 1-\varepsilon \leq \frac{\widehat{D}(x)}{D(x)} \leq 1+\varepsilon\,.$$

[1] If $i > d(v)$ then a special symbol, e.g. $\perp$, is returned.

If $D = U$, the uniform distribution on X, then we say that $\widehat{D}$ is ***pointwise ε-close to uniform****.*

For the sake of conciseness, from this point on, unless explicitly stated otherwise, when we refer to an edge sampling algorithm, we mean an algorithm that returns edges according to a distribution that is pointwise-close to uniform.

▶ **Definition 1.2.** *Let $G = (V, E)$ be an undirected graph. A forest $F = (V_F, E_F)$ (i.e., a graph containing no cycles) with vertex set $V_F = V$ and edge set $E_F \subseteq E$ is a* ***spanning forest*** *of G. We say that a family of spanning forests $F_1, F_2, \ldots, F_k$* ***covers*** *G if $E = \bigcup_{i=1}^{k} E_{F_i}$. The* ***arboricity*** *of G, denoted $\alpha(G)$, is the minimum k such that there exists a family of spanning forests of size k that covers G.*

An edge-sampling algorithm is given as input an approximation parameter $\varepsilon \in (0, 1)$ and a parameter α which is an upper bound on the arboricity of G. The algorithm is required to sample edges according a distribution that is pointwise ε-close to uniform. To this end the algorithm is given query access to G. In particular we consider the aforementioned uniform vertex sampling model.

1.2 Results

We prove almost matching upper and lower bounds on the query complexity of sampling an edge according to a distribution that is pointwise-close to uniform when an upper bound on the arboricity of the graph is known. The first lower bound stated below (Theorem 2) holds even for the easier task of sampling from a distribution that is close to uniform in TVD.

▶ **Theorem 1.** *There exists an algorithm $\mathcal{A}$ that for any n, m, α, and graph $G = (V, E)$ with n nodes, m edges, and arboricity at most α, satisfies the following. Given n and α, $\mathcal{A}$ returns an edge $e \in E$ sampled from a distribution $\widehat{U}$ that is pointwise ε-close to uniform using $O\left(\frac{\alpha}{d} \cdot \frac{\log^3 n}{\varepsilon}\right)$ degree and neighbor queries in expectation, where $d = m/n$.*

In Section 1.5.1 we provide a high-level presentation of the algorithm referred to in Theorem 1 and discuss how it differs from the algorithm in [15] (for the case that there is no given upper bound on the arboricity).

We next state our lower bound, which matches the upper bound in Theorem 1 up to a polylogarithmic dependence on n (for constant ε).

▶ **Theorem 2.** *Fix $\varepsilon \leq 1/6$ and let n, m and α be parameters such that $\alpha = \alpha(n) \leq \sqrt{m}$ and $m \leq n\alpha$. Let $\mathcal{G}^{\alpha}_{n,m}$ be the family of graphs with n vertices, m edges and arboricity at most α. Then any algorithm $\mathcal{A}$ that for any $G \in \mathcal{G}^{\alpha}_{n,m}$ samples edges in G from a distribution that is ε-close to uniform in total variation distance – and in particular, any distribution that is pointwise ε-close to uniform – requires $\Omega(\alpha/d)$ queries in expectation.*

When α is a constant then (assuming that $m = \Omega(n)$) the lower bound in Theorem 2 is simply $\Omega(1)$, while Theorem 1 gives an upper bound of $O(\log^3 n)$ (for constant ϵ). We prove that an almost linear dependence on $\log n$ is necessary, even for the case that $\alpha = 1$ (where the graph is a tree).

▶ **Theorem 3.** *Fix $\varepsilon \leq 1/6$, and let $\mathcal{T}_n$ be the family of trees on n vertices. Then any algorithm $\mathcal{A}$ that for any $G \in \mathcal{T}_n$ samples edges in G from a distribution that is pointwise ε-close to uniform requires $\Omega\left(\frac{\log n}{\log\log n}\right)$ queries in expectation.*

We note that both of our lower bounds hold when the algorithm is also given access to *pair queries*, $\mathsf{pair}(u, v)$, which state whether u and v share an edge.

1.3 Discussion of the results

Comparison to known results. The simplest algorithm for sampling edges uniformly is based on rejection sampling. Namely, it repeats the following until an edge is output: Sample a uniform vertex u, flip a coin with bias $d(u)/d_{\max}$, where $d_{\max}$ is the maximum degree, and if the outcome is HEADS, then output a random edge (u, v) incident to u. The expected complexity of rejection sampling grows like $d_{\max}/d$, that is, linearly with the maximum degree. As noted earlier, Eden and Rosenbaum [11] show that this dependence on the maximum degree is not necessary (for approximate sampling), as $O^*(\sqrt{m}/d)$ queries and time always suffice, even when the maximum degree is not bounded (e.g., is $\Theta(n)$). However, if the maximum degree is bounded, and in particular if $d_{\max} = o(\sqrt{m})$, then rejection sampling has lower complexity than the [11] algorithm (e.g., in the case that the graph is close to regular, so that when $d_{\max} = O(d)$, we get complexity $O(1)$).

Since the arboricity of a graph is both upper bounded by $d_{\max}$ and by $\sqrt{m}$, our algorithm can be viewed as "enjoying both worlds". Furthermore, our results can be viewed as showing that the appropriate complexity measure for sampling edges is not the maximum degree but rather the maximum average degree (recall that the arboricity α measures the maximum density of any subgraph of G – for a precise statement, see Theorem 4).

Sampling vs. Estimating. As shown by Eden, Ron and Seshadhri [11], $\Theta^*(\alpha/d)$ is also the complexity of estimating the number of edges in a graph when given a bound α on the arboricity of the graph. Their algorithm improves on the previous known bound of $O^*(\sqrt{m}/d)$ by Feige [17] and Goldreich and Ron [19] (when the arboricity is $o(\sqrt{m})$). However, other than the complexity, our algorithm for sampling edges and the algorithm of [11] for estimating the number of edges do not share any similarities, in particular, as the result of [11] is allowed to "ignore" an ε-fraction of the graph edges.

Furthermore, while the complexity of sampling and of approximate counting of edges are the same (up to $\log n$ and $1/\varepsilon$ dependencies), we have preliminary results showing that for other subgraphs this is not necessarily the case. Specifically, there exist graphs with constant arboricity for which estimating the number of triangles can be done using $O^*(1)$ queries, but pointwise-close to uniform sampling requires $\Omega(n^{1/4})$ queries in expectation.

On the necessity of being provided with an upper bound on the arboricity. While our algorithm does not require to be given any bound on the average degree d, it must be provided with an upper bound α on the arboricity of the given graph. To see why this is true, consider the following two graphs. The first graph consists of a perfect matching between its vertices, so that both its average degree and its arboricity are 1. For $\tilde{\alpha} > 1$, the second graph consists of a perfect matching over $n - n/\tilde{\alpha}$ vertices and an $\tilde{\alpha}$-regular graph over the remaining $n/\tilde{\alpha}$ vertices. This graph has an average degree of roughly 2, and arboricity $\tilde{\alpha}$. If an edge-sampling algorithm is not provided with an appropriate upper bound on the arboricity, but is still required to run in (expected) time that grows like the ratio between the arboricity and the average degree, then the algorithm can be used to distinguish between the two graphs. However, assuming a random labeling of the vertices of the two graphs, this cannot be done in time $o(\tilde{\alpha})$.[2]

Pointwise closeness vs. closeness with respect to the TVD. The lower bound of Theorem 2 holds for sampling from a distribution that is close to uniform with respect to TVD,

[2] We note that this construction can be extended to work for any $d \leq \tilde{\alpha}$.

and *a fortiori* to sampling from pointwise-almost-uniform distributions. In contrast, the lower bound of Theorem 3 does not apply to sampling edges from a distribution that is ε-close to uniform in TVD. Indeed, a simple rejection sampling procedure (essentially ignoring all nodes with degrees greater than $1/\varepsilon$) can sample edges from a distribution that is ε-close to uniform in TVD using $O(1/\varepsilon)$ queries in expectation. Thus, Theorem 3 gives a separation between the tasks of sampling from distributions that are pointwise-close to uniform versus close to uniform in TVD. The general upper and lower bounds of Theorems 1 and 2 show that the separation between the complexity of these tasks can be at most poly-logarithmic for any graph.

1.4 An application to approximately counting subgraphs

In a recent paper [2], Assadi, Kapralov, and Khanna made significant progress on the question of counting arbitrary subgraphs in a graph in sublinear time. Specifically, they provide an algorithm that estimates the number of occurrences of any arbitrary subgraph H in G, denoted by $\#H$, to within a $(1 \pm \varepsilon)$-approximation with high probability. The running time of their algorithm is $O^*\left(\frac{m^{\rho(H)}}{\#H}\right)$, where $\rho(H)$ is the fractional edge cover of H.[3] Their algorithm assumes access to uniform edge samples in addition to degree, neighbor and pair queries. As noted in [2], their algorithm can be adapted to work with edge samples that are pointwise ε-close to uniform (where this is not true for edge samples that are only ε-close to uniform in TVD – e.g., when all the occurrences of H are induced by an ε-fraction of the edges). Invoking the algorithm of [2], and replacing each edge sample with an invocation of Sample-edge results in the following corollary.

▶ **Corollary 1.** *Let G be a graph $G = (V, E)$ with n nodes, m edges, and arboricity at most α. There exists an algorithm that, given $n, \alpha, \varepsilon \in (0, 1)$, a subgraph H and query access to G, returns a $(1 \pm \varepsilon)$ approximation of the number of occurrences of H in G, denoted $\#H$. The expected query complexity and running time of the algorithm are*

$$O^*\left(\min\left\{m, \frac{n\alpha \cdot m^{\rho(H)-1}}{\#H}\right\}\right) \quad \text{and} \quad O^*\left(\frac{n\alpha \cdot m^{\rho(H)-1}}{\#H}\right),$$

respectively, where $\rho(H)$ denotes the fractional edge cover of H, and the allowed queries are degree, neighbor and pair queries.

Thus, by combining our result with [2], we extend the known results for approximately counting the number of subgraphs in a graph in the uniform vertex sampling model. Furthermore, for graphs in which $m = \Theta(n\alpha)$, we obtain the same query complexity and running time of [2] without the assumption that the algorithm has access to uniform edge samples.

1.5 A high-level presentation of the algorithm and lower bounds

1.5.1 The algorithm

While our results concern *un*directed graphs $G = (V, E)$, it will be helpful to view each edge $\{u, v\} \in E$ as a pair of *ordered* edges (u, v) and (v, u). Sampling an edge (almost) uniformly is equivalent to sampling a vertex with probability (almost) proportional to its degree. Hence

[3] The fractional edge cover of a graph $H = (V_H, E_H)$ is a mapping $\psi : E_H \to [0, 1]$ such that for each vertex $a \in V_H$, $\sum_{e \in E_H, a \in e} \psi(e) \geq 1$. The fractional edge-cover number $\rho(H)$ of H is the minimum value of $\sum_{e \in E_H} \psi(e)$ among all fractional edge covers ψ.

we focus on the latter task. A single iteration of the algorithm we describe either returns a vertex or outputs FAIL. We show that the probability that it outputs FAIL is not too large, and that conditioned on the algorithm returning a vertex, each vertex v is returned with probability proportional to its degree, up to a factor of $(1 \pm \varepsilon)$.

Our starting point is a structural decomposition result for graphs with bounded arboricity (Lemma 2.3). Our decomposition defines a partition of the graph's vertices into *levels* $L_0, L_1, \ldots, L_\ell$. For parameters θ and β, the first layer L_0 consists of all vertices with degree at most θ, and for $i > 0$, level L_i contains all vertices v that do not belong to previous levels $L_0, \ldots, L_{i-1}$, but have at least $(1-\beta)d(v)$ neighbors in these levels. We prove that that for any graph with arboricity at most α, for $\theta = \Theta(\alpha \log n/\varepsilon)$ and $\beta = \Theta(\varepsilon/\log n)$, there exists such a partition into layers with $\ell = O(\log n)$ levels. We stress that the algorithm does not actually construct such a partition, but rather we use the partition in our analysis of the algorithm.[4]

In order to gain intuition about the algorithm and its analysis, suppose that all vertices in L_0 have degree exactly θ, and that all edges in the graph are between vertices in consecutive layers. Consider the following *random walk* algorithm. The algorithm first selects an index $j \in [0, \ell]$ uniformly at random. It then selects a vertex u_0 uniformly at random. If $u_0 \in L_0$, then it performs a random walk of length j starting from u_0 (otherwise it outputs FAIL). If the walk did not pass through any vertex in L_0 (with the exception of the starting vertex u_0), then the algorithm returns the final vertex reached.

First observe that for every $u \in L_0$, the probability that u is returned is $\frac{1}{\ell+1} \cdot \frac{1}{n}$ (the probability that the algorithm selected $j = 0$ and selected u as u_0). This equals $\frac{d(u)}{(\ell+1)\cdot\theta\cdot n}$ (by our assumption that $d(u) = \theta$ for every $u \in L_0$). Now consider a vertex $v \in L_1$. The probability that v is returned is at least $\frac{1}{\ell+1} \cdot \frac{(1-\beta)d(v)}{n} \cdot \frac{1}{\theta} = \frac{(1-\beta)d(v)}{(\ell+1)\cdot\theta\cdot n}$ (the probability that the algorithm selected $j = 1$, then selected one of v's neighbors $u \in L_0$, and finally selected to take the edge between u and v). In general, our analysis shows that for every i and every $v \in L_i$, the probability that v is returned is at least $\frac{(1-\beta)^i d(v)}{(\ell+1)\cdot\theta\cdot n}$. On the other hand, we show that for every vertex v, the probability that v is returned is at most $\frac{d(v)}{(\ell+1)\cdot\theta\cdot n}$. By the choice of θ and β we get that each vertex v is returned with probability in the range $[(1-\varepsilon)d(v)\rho(\varepsilon, n), d(v)\rho(\varepsilon, n)]$ for $\rho(\varepsilon, n) = \Theta(\varepsilon/(\alpha n \log^2 n))$. By repeating the aforementioned random-walk process until a vertex is returned – $\Theta\left(\frac{\alpha n}{m} \cdot \frac{\log^2 n}{\varepsilon}\right) = \Theta\left(\frac{\alpha}{d} \cdot \frac{\log^2 n}{\varepsilon}\right)$ times in expectation – we obtain a vertex that is sampled with probability proportional to its degree, up to $(1 \pm \varepsilon)$.

We circumvent the assumption that $d(u) = \theta$ for every $u \in L_0$ by rejection sampling: In the first step, if the algorithm samples $u_0 \in L_0$, then it continues with probability $d(u)/\theta$ and fails otherwise. The assumption that all edges are between consecutive levels is not necessary for the analysis described above to hold. The crucial element in the analysis is that for every vertex $v \notin L_i$, where $i > 0$, at least $(1-\beta)$ of the neighbors of v belong to $L_0, \ldots, L_{i-1}$. This allows us to apply the inductive argument for the lower bound on the probability that v is returned when we average over all choices of j (the number of steps in the random walk). For precise details of the algorithm and its analysis, see Section 2.

We briefly discuss the relation between our algorithm for bounded-arboricity graphs, which we denote by $\mathcal{A}_{ba}$ and the algorithm presented in [15] (for the case that no upper bound is given on the arboricity), which we denote by $\mathcal{A}_{ua}$. The algorithm $\mathcal{A}_{ua}$ can be

[4] This decomposition is related to the forest decomposition of Barenboim and Elkin [4]. The main difference, which is essential for our analysis, is that the partition we define is based on the number of neighbors that a vertex has in lower levels *relative* to its degree, while in [4] the partition is based on the absolute number of neighbors in higher levels.

viewed as considering a partition of the graph vertices into just two layers according to a degree threshold of roughly $\sqrt{m/\varepsilon}$. It performs a random walk similarly to $\mathcal{A}_{ba}$, but where the walk has either length 0 or 1. This difference in the number of layers and the length of the walk, is not only quantitative. Rather, it allows $\mathcal{A}_{ua}$ to determine to which layer a vertex belongs simply according to its degree. This is not possible in the case of $\mathcal{A}_{ba}$ (with the exception of vertices in L_0). Nonetheless, despite the apparent "blindness" of $\mathcal{A}_{ba}$ to the layers it traverses in the random walk, we can show the following: Choosing the length of the random walk uniformly at random and halting in case the walk returns to L_0, ensures that each vertex is output with probability approximately proportional to its degree.

1.5.2 The lower bounds

In order to prove Theorem 2, we employ the method of [14] – which builds upon the paradigm introduced in [7] – based on communication complexity. The idea of the proof is to reduce from the two-party communication complexity problem of computing the disjointness function. The reduction is such that (1) any algorithm that samples edges from an almost-uniform distribution reveals the value of the disjointness function with sufficiently large probability, and (2) every allowable query can be simulated in the two-party communication setting using little communication.

As opposed to the proof of the lower bound for general α, in the case of $\alpha = 1$ we did not find a way to employ the communication-complexity method. Instead, we design a direct, albeit somewhat involved, proof from first-principles.

Specifically, in order to prove Theorem 3, we consider a complete tree in which each internal vertex has degree $\log n$ (so that its depth is $\Theta\left(\frac{\log n}{\log\log n}\right)$). We then consider the family of graphs that correspond to all possible labelings of such a tree. As noted in Section 1.5.1, sampling edges almost uniformly is equivalent to sampling vertices with probability approximately proportional to their degree. In particular, in our construction, the label of the root should be returned with probability approximately $\log n/n$. We show that any algorithm that succeeds in returning the label of the root of the tree with the required probability must perform $\Omega\left(\frac{\log n}{\log\log n}\right)$ queries.

To this end we define a *process* $\mathcal{P}$ that interacts with any algorithm $\mathcal{A}$, answering $\mathcal{A}$'s queries while constructing a uniform random labeling of the vertices and edges in the tree. The vertices of the tree are assigned random labels in $[n]$, and for each vertex v in the tree, its incident edges are assigned random labels in $[d(v)]$. We say that $\mathcal{A}$ *succeeds*, if after the interaction ends, $\mathcal{A}$ outputs the label of the root of the tree, as assigned by $\mathcal{P}$.

Let $L = \frac{\log n}{C \cdot \log\log n}$ be the lower bound we would like to prove, where C is a sufficiently large constant (so that in particular, L is a (small) constant fraction of the depth of the tree). Intuitively, $\mathcal{A}$ would like to "hit" a vertex at depth at most L and then "walk up the tree" to the root. There are two sources of uncertainty for $\mathcal{A}$. One is whether it actually hits a vertex at depth at most L, and the second is which edges should be taken to go up the tree. Our lower bound argument mainly exploits the second uncertainty, as we sketch next.

The process P starts with an unlabeled tree (of the aforementioned structure), and assigns labels to its vertices and edges in the course of its interaction with $\mathcal{A}$. Recall that $\mathcal{P}$ answers the queries of $\mathcal{A}$ while constructing a uniform labeling. Therefore, whenever $\mathcal{A}$ asks a query involving a *new* label (i.e., that has not yet appeared in its queries or answers to them), the vertex to which this label is assigned, should be uniformly selected among all vertices that are not yet labeled. We shall say that a vertex is *critical* it its depth is at most L. As long as no critical vertex is hit, $\mathcal{A}$ cannot reach the root. This implies that if no critical vertex is

hit in the course of its queries, then the probability that $\mathcal{A}$ succeeds is $O(1/n)$.

While the probability of hitting a critical vertex is relatively small, it is not sufficiently small to be deemed negligible. However, suppose that $\mathcal{A}$ hits a critical vertex u at depth $\Delta < L$, which occurs with probability roughly $(\log n)^{\Delta}/n$. Then, conditioned on this event, each of the $(\log n)^{\Delta}$ edge-labeled paths from u is equally likely to lead to the root, and the labels of vertices on these paths are uniformly distributed, thus intuitively conveying no information regarding the "right path".

A subtlety that arises when formalizing this argument is the following. Suppose that in addition to hitting a critical vertex u, $\mathcal{A}$ hits another vertex, v, which is not necessarily critical, but is at distance less than L from u (and in particular has depth at most $2L$, which we refer to as *shallow*). Then, a path starting from v might meet a path starting from u, hence adding a conditioning that makes the above argument (regarding uniform labelings) imprecise. We address this issue by upper bounding the probability of such an event (i.e., of hitting both a critical vertex and a shallow vertex), and accounting for an event of this type as a success of $\mathcal{A}$.

1.6 Related work

Some of the works presented below were already mentioned earlier in the introduction, but are provided in this subsection for the sake of completeness.

The work most closely related to the present work is the recent paper of Eden and Rosenbaum [15]. In [15], the authors proved matching upper and lower bounds of $\Theta^*(\sqrt{m}/d)$ for the problem of sampling an edge from an almost-uniform distribution in an arbitrary graph using degree, neighbor, and pair queries.

The problem of sampling edges in a graph is closely related to the problem of estimating m, the number of edges in the graph. In [17], Feige proved an upper bound of $O^*(\sqrt{m}/d)$ for obtaining a $(2+\varepsilon)$-factor multiplicative approximation of m using only degree queries,[5] and shows that it is not possible to go below a factor of 2 with a sublinear number of degree queries. In [19], Goldreich and Ron showed that $\Theta^*(\sqrt{m}/d)$ queries are necessary and sufficient to obtain a $(1+\varepsilon)$-factor approximation of m if neighbor queries are also allowed.

Several works prove matching upper and lower bounds on the query complexity of counting the number of triangles [8], cliques [13], and star-graphs of a given size [20] using degree, neighbor, and pair queries (when the latter are necessary). Eden, Ron, and Seshadhri devised algorithms for estimating the number of k-cliques [12] and moments of the degree distribution [11] whose runtimes are parameterized by the arboricity α of the input graph (assuming a suitable upper bound for α is given to the algorithm as input). These algorithms outperform the lower bounds of [13] and [20] (respectively) in the case where $\alpha \ll \sqrt{m}$. In [9], Eden, Levi, and Ron described an efficient algorithm for distinguishing graphs with arboricity at most α from those that are far from any graph with arboricity 3α.

Two recent works [1, 2] consider a query model that allows uniform random edge sampling in addition to degree, neighbor, and pair queries. In this model, Aliakbarpour et al. [1] described an algorithm for estimating the number of star subgraphs. In the same model, Assadi et al. [2] devised an algorithm that relies on uniform edge samples as a basic query to approximately count the number of instances of an arbitrary subgraph in a graph. The

[5] To be precise, Feige [17] shows that, given a lower bound d_0 on the average degree, $O(\sqrt{n/d_0}/\varepsilon)$ degree queries are sufficient. If such a lower bound is not provided to the algorithm, then a geometric search can be performed, as shown in [19].

results in [1] and [2] imply that uniform edge samples afford the model strictly more power: the sample complexity of the algorithm of [1] outperforms the lower bound of [20] for the same task, and the sample complexity of the algorithm of [2] outperforms the lower bound of [13] for estimating the number of cliques. (The results of [20, 13] are in the uniform vertex sampling model.)

Organization

Due to space constraints, in this extended abstract we provide full details only for the proof of our upper bound (Theorem 1). The proofs of Theorems 2 and 3 can be found in [10].

2 The Algorithm

In this section we describe an algorithm that samples an edge e from an arbitrary graph G with arboricity at most α, according to a pointwise almost uniform distribution. Theorem 1 follows from our analysis of the algorithm. In what follows, for integers $i \leq j$, we use $[i, j]$ to denote the set of integers $\{i, \ldots, j\}$, and for a vertex v we let $\Gamma(v)$ denote its set of neighbors.

As noted in the introduction, sampling edges from a uniform distribution is equivalent to sampling vertices proportional to their degrees. Indeed, if each vertex v is sampled with probability $d(v)/2m$, then choosing a random neighbor $w \in \Gamma(v)$ uniformly at random returns the (directed) edge $e = (v, w)$ with probability $1/2m$. Thus, it suffices to sample each vertex $v \in V$ with probability (approximately) proportional to its degree.

2.1 Decomposing graphs of bounded arboricity

Before describing the algorithm, we describe a decomposition of a graph G into *layers* depending on its arboricity. We begin by recalling the following characterization of arboricity due to Nash-Williams [21].

▶ **Theorem 4** (Nash-Williams [21]). *Let $G = (V, E)$ be a graph. For a subgraph H of G, let n_H and m_H denote the number of vertices and edges, respectively, in H. Then $\alpha(G) = \max_H \{\lceil m_H/(n_H - 1)\rceil\}$, where the maximum is taken over all subgraphs H of G.*

▶ **Definition 2.1.** *Let $G = (V, E)$ be a graph, and $\theta \in \mathbf{N}$, $\beta \in (0, 1)$ parameters. We define a (θ, β)-**layering** in G to be the sequence of non-empty disjoint subsets $L_0, L_1, \ldots, L_\ell \subseteq V$ defined by $L_0 = \{v \in V \mid d(v) \leq \theta\}$ and for $i \geq 1$,*

$$L_{i+1} = \left\{v \notin L_0 \cup L_1 \cup \cdots \cup L_i \;\middle|\; |\Gamma(v) \cap (L_0 \cup \cdots \cup L_i)| \geq (1 - \beta)d(v)\right\}.$$

*That is, L_0 consists of all vertices of degree at most θ, and a vertex v is in L_{i+1} if i is the smallest index for which a $(1 - \beta)$-fraction of v's neighbors resides in $L_0 \cup L_1 \cup \cdots \cup L_i$. We say that G admits a (θ, β)-**layered partition of depth** ℓ if we have $V = L_0 \cup L_1 \cup \cdots \cup L_\ell$.*

▶ **Notation 2.2.** *For a fixed i, we denote $L_{\leq i} = L_0 \cup L_1 \cup \cdots \cup L_i$, and similarly for $L_{<i}$, $L_{\geq i}$, and $L_{>i}$. We use the notation $d_i(v)$ to denote $|\Gamma(v) \cap L_i|$ and similarly for $d_{\leq i}(v)$ and $d_{\geq i}(v)$.*

▶ **Lemma 2.3.** *Suppose G is a graph with arboricity at most α. Then G admits a (θ, β)-layered partition of depth ℓ for $\theta = 4\alpha\lceil \log n\rceil/\varepsilon$, $\beta = \varepsilon/2\lceil \log n\rceil$, and $\ell \leq \lceil \log n\rceil$.*

Proof. For each i, let $W_i = V \setminus (L_0 \cup L_1 \cup \cdots \cup L_{i-1})$ be the set of vertices not in levels $0, 1, \ldots, i - 1$. Let $m(W_i)$ denote the number of edges in the subgraph of G induced by W_i.

For any fixed i and $v \in W_{i+1}$, we have $d_{<i}(v) < (1-\beta)d(v)$ because $v \notin L_{\leq i}$. Therefore, v has at least $\beta d(v) > \beta\theta$ neighbors in W_i. Summing over vertices $v \in W_{i+1}$ gives

$$m(W_i) = \frac{1}{2}\sum_{v\in W_i} d_{\geq i}(v) \geq \frac{1}{2}\sum_{v\in W_{i+1}} d_{\geq i}(v) > \frac{1}{2}\,|W_{i+1}| \cdot \beta\theta\,. \tag{1}$$

On the other hand, since G has arboricity at most α, Theorem 4 implies that $m(W_i) \leq \alpha\,|W_i|$. Plugging this into Equation (1), we find that $\frac{|W_{i+1}|}{|W_i|} \leq \frac{2\alpha}{\beta\theta} = \frac{1}{2}$, where the inequality is by the choice of β and θ. Therefore, for $\ell \leq \lceil \log n \rceil$, we have that $W_{\ell+1} = \emptyset$, implying that $V = L_0 \cup L_1 \cup \cdots \cup L_\ell$. ◀

2.2 Algorithm description

The algorithm exploits the structure of graphs G with arboricity at most α described in Lemma 2.3. More precisely, as the algorithm does not have direct access to this structure, the structure is used explicitly only in the analysis of the algorithm. Let $L_0, L_1, \ldots, L_\ell$ be a (θ,β)-layered partition of V with $\theta = 4\alpha\lceil\log n\rceil/\varepsilon$, $\beta = \varepsilon/\lceil\log n\rceil$, and $\ell = \lceil\log n\rceil$. Vertices $v \in L_0$ are sampled with probability exactly proportional to their degree using a simple rejection sampling procedure, Sample-$L_0(G,\theta)$. In order to sample vertices in layers L_i for $i > 0$, our algorithm performs a random walk starting from a random vertex in L_0 chosen with probability proportional to its degree. Specifically, the algorithm Sample-edge(G,α) chooses a length j to the random walk uniformly in $[0,\ell]$. The subroutine Random-walk(G,θ,j) performs the random walk for j steps, or until a vertex $v \in L_0$ is reached in some step $i > 0$. If the walk returns to L_0, the subroutine aborts and does not return any vertex. (This behavior ensures that samples are not too biased towards vertices in lower layers.) Otherwise, Random-walk returns the vertex at which the random walk halts. Our analysis shows that the probability that the random walk terminates at any vertex $v \in V$ is approximately proportional to $d(v)$ (Corollary 2.7), although Sample-edge may fail to return any edge with significant probability. Finally, we repeat Sample-edge until it successfully returns a vertex.

Sample-edge(G,α,ε)

1. Let $\theta = 4\alpha\lceil\log n\rceil/\varepsilon$ and let $\ell = \lceil\log n\rceil$.
2. Choose a number $j \in [0,\ell]$ uniformly at random.
3. Invoke Random-walk(G,θ,j) and let v be the returned vertex if one was returned. Otherwise, **return** FAIL.
4. Sample a uniform neighbor w of v and **return** $e = (v,w)$.

Random-walk(G,θ,j)

1. Invoke Sample-$L_0(\theta)$ and let v_0 be the returned vertex if one was returned. Otherwise, **return** FAIL.
2. For $i = 1$ to j do
   a. Sample a random neighbor v_i of v_{i-1}.
   b. If $v_i \in L_0$ then **return** FAIL.
3. **Return** v_j.

Sample-$L_0(G,\theta)$

1. Sample a vertex $u \in V$ uniformly at random and query for its degree.
2. If $d(u) > \theta$ **return** FAIL.
3. **Return** u with probability $\frac{d(u)}{\theta}$, and with probability $1 - \frac{d(u)}{\theta}$ **return** FAIL.

▶ **Definition 2.4.** *We let $P_j[v]$ denote the probability that* Random-walk *returns v, when invoked with parameters G, θ and $j \in [0,\ell]$. We also let $P_{\leq j}[v] \stackrel{def}{=} \sum_{i=0}^{j} P_i[v]$ and similarly for $P_{\geq j}$.*

▶ **Lemma 2.5.** *Let ℓ be as set in Step 1 of* Sample-edge *and let $P_{\leq j}$ be as defined in Definition 2.4. For all $v \in V$, $P_{\leq \ell}[v] \leq \frac{d(v)}{n\theta}$.*

Proof. We argue by induction on j that for any $j \in [0,\ell]$, $P_{\leq j}[v] \leq d(v)/n\theta$. For the case $j = 0$, it is immediate from the description of Random-walk and Sample-L_0 that $P_0[v] = d(v)/(n\theta)$ if $v \in L_0$ and $P_0[v] = 0$ otherwise. Further, for $v \in L_0$, due to Step 2b, $P_i(v) = 0$ for all $i > 0$, so that the lemma holds for all $v \in L_0$. Now suppose that for all $v \in V$ we have $P_{\leq j-1}(v) \leq d(v)/n\theta$. Then for any fixed $v \notin L_0$ we compute

$$P_{\leq j}[v] = \sum_{i=1}^{j} P_i[v] = \sum_{i=1}^{j} \sum_{u\in\Gamma(v)} P_{i-1}[u]\frac{1}{d(u)} = \sum_{u\in\Gamma(v)} \frac{1}{d(u)} \sum_{i=0}^{j-1} P_i[u]$$
$$= \sum_{u\in\Gamma(v)} \frac{1}{d(u)} P_{\leq j-1}[u] \leq \sum_{u\in\Gamma(v)} \frac{1}{d(u)} \frac{d(u)}{n\theta} = \frac{d(v)}{n\theta}.$$

The second equality holds by the definition of Random-walk, and the one before the last inequality holds by the inductive hypothesis. ◀

▶ **Lemma 2.6.** *Let ℓ be as set in Step 1 of* Sample-edge*. For every $j \in [\ell]$, $v \in L_j$ and $k \in [j,\ell]$, we have $P_{\leq k}[v] \geq \frac{(1-\beta)^j d(v)}{n\theta}$.*

Proof. We prove the claim by induction on j. For $j = 0$ and $k = 0$, by the description of Random-walk and Sample-L_0, for every $v \in L_0$,

$$P_0[v] = \frac{d(v)}{n\theta} . \tag{2}$$

For $j = 0$ and $0 < k \leq \ell$,

$$P_{\leq k}[v] = \sum_{i=0}^{k} P_i[v] = P_0[v] + \sum_{i=1}^{k} P_i[v] = \frac{d(v)}{n\theta} , \tag{3}$$

where the last equality is due to Step 2b in Random-walk.

For $j = 1$ and $1 \leq k \leq \ell$, for every $v \in L_1$, according to Step 2b in the procedure Random-walk, $P_0[v] = 0$. Also, for every $u \notin L_0$, $P_0[u] = 0$, since by Step 2 in Sample-L_0 it always holds that v_0 is in L_0. Therefore,

$$P_1[v] = \sum_{u\in\Gamma(v)} P_0[u]\cdot\frac{1}{d(u)} = \sum_{u\in\Gamma(v)\cap L_0} P_0[u]\cdot\frac{1}{d(u)} = \sum_{u\in\Gamma(v)\cap L_0} \frac{d(u)}{n\theta}\cdot\frac{1}{d(u)} = \frac{d_0(v)}{n\theta}, \tag{4}$$

where the second to last inequality is by Equation (2). By the definition of L_1, for every $v \in L_1$, $d_0(v) \geq (1-\beta)d(v)$, and it follows that $P_{\leq k}[v] \geq P_1[v] \geq (1-\beta)d(v)/(n\theta)$.

We now assume that the claim holds for all $i \leq j-1$ and $k \in [i,\ell]$, and prove that it holds for j and for every $k \in [j,\ell]$. By the induction hypothesis and the definition of L_j, for any $v \in L_j$ we have

$$P_{\leq k}[v] \geq P_{\leq j}[v] = \sum_{u\in\Gamma(v)} P_{\leq j-1}[u]\cdot\frac{1}{d(u)} \geq \sum_{i=0}^{j-1} \sum_{u\in\Gamma(v)\cap L_i} P_{\leq j-1}[u]\cdot\frac{1}{d(u)}$$
$$\geq \sum_{i=0}^{j-1} \sum_{u\in\Gamma(v)\cap L_i} \frac{(1-\beta)^i d(u)}{n\theta}\cdot\frac{1}{d(u)} \geq \frac{(1-\beta)^{j-1} d_{\leq j-1}(v)}{n\theta} \geq \frac{(1-\beta)^j d(v)}{n\theta}.$$

Hence, the claim holds for every $j \in [\ell]$ for every $k \in [j,\ell]$. ◀

▶ **Corollary 2.7.** *For any graph G with arboricity at most α, the procedure Sample-edge when invoked with G, α and ε, returns each edge in the graph with probability in the range $\left[\frac{1-\varepsilon/2}{\rho}, \frac{1}{\rho}\right]$ for $\rho = n\theta(\ell+1)$, $\theta = 4\alpha\lceil\log n\rceil/\varepsilon$ and $\ell = \lceil\log n\rceil$.*

Proof. Consider a specific edge $e^* = (v^*, w^*)$, and let i be the index such that $v^* \in L_i$. By the description of the procedure Sample-edge, the procedure Random-walk is invoked with an index j that is chosen uniformly in $[0, \ell]$. Hence, the probability that v^* is returned by Random-walk in Step 3 is $\Pr[v = v^*] = \frac{1}{\ell+1}\sum_{j=0}^{\ell} P_j[v] = \frac{1}{\ell+1}P_{\leq\ell}[v]$. By Lemma 2.5, $P_{\leq\ell}[v] \leq \frac{d(v)}{n\theta}$, and by Lemma 2.6, $P_{\leq\ell}[v^*] \geq \frac{(1-\beta)^\ell d(v^*)}{n\theta}$, where the probability is over the random coins of the procedures Sample-edge and Random-walk. Hence,

$$\Pr[v = v^*] \in [(1-\beta)^\ell, 1] \cdot \frac{d(v^*)}{n\theta(\ell+1)},$$

implying that for $\rho = n\theta(\ell+1)$,

$$\Pr[(v^*, w^*) \text{ is the returned edge}] \in [(1-\beta)^\ell, 1] \cdot \frac{1}{n\theta(\ell+1)} \in \left[\frac{1-\varepsilon/2}{\rho}, \frac{1}{\rho}\right], \tag{5}$$

where the last inequality is by the setting of $\beta = \varepsilon/2\lceil\log n\rceil$. ◀

Proof of Theorem 1. Consider the algorithm that repeatedly calls Sample-edge(G, α) until an edge e is successfully returned. For a single invocation of Sample-edge and fixed edge e let A_e denote the event that Sample-edge returns e. By Corollary 2.7 we have that $\Pr[A_e] \geq (1-\varepsilon)/n\theta(\ell+1)$. Further, for any edge $e' \neq e$ the events A_e and $A_{e'}$ are disjoint, so we bound

$$\Pr[\text{Sample-edge returns an edge}] \;=\; \Pr\left[\bigcup_{e\in E} A_e\right] \;=\; \sum_{e\in E}\Pr[A_e] \;\geq\; \frac{(1-\varepsilon)m}{n\theta(\ell+1)}\,.$$

The expected number of iterations until Sample-edge succeeds is the reciprocal of this probability, so

$$\mathbf{E}[\#\text{ invocations until success}] \leq \frac{n\theta(\ell+1)}{(1-\varepsilon)m} = O\left(\frac{n\alpha}{m\varepsilon}\cdot\log^2 n\right).$$

Since each invocation of Sample-edge uses $O(\log n)$ queries, the expected number of queries before an edge is returned is $O(\frac{n\alpha}{\varepsilon m}\cdot\log^3 n)$.

Finally, when conditioned on a successful invocation of Sample-edge, Corollary 2.7 implies that for any $e, f \in E$ the probabilities p_e, p_f of returning e and f, respectively, satisfy

$$1 - \varepsilon/2 \leq \frac{p_e}{p_f} \leq \frac{1}{1-\varepsilon/2} \leq 1 + \varepsilon.$$

Therefore, the induced distribution P over edges returned by a successful invocation of Sample-edge is pointwise ε-close to uniform, which gives the desired result. ◀

References

1 Maryam Aliakbarpour, Amartya Shankha Biswas, Themis Gouleakis, John Peebles, Ronitt Rubinfeld, and Anak Yodpinyanee. Sublinear-Time Algorithms for Counting Star Subgraphs via Edge Sampling. *Algorithmica*, pages 1–30, 2017. doi:10.1007/s00453-017-0287-3.

2 Sepehr Assadi, Michael Kapralov, and Sanjeev Khanna. A Simple Sublinear-Time Algorithm for Counting Arbitrary Subgraphs via Edge Sampling. In *ITCS*, volume 124 of *LIPIcs*, pages 6:1–6:20. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019.

3 Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

4 Leonid Barenboim and Michael Elkin. Sublogarithmic distributed MIS algorithm for sparse graphs using Nash-Williams decomposition. *Distributed Computing*, 22(5-6):363–379, 2010. doi:10.1007/s00446-009-0088-2.

5 Reinhard Bauer, Marcus Krug, and Dorothea Wagner. Enumerating and generating labeled k-degenerate graphs. In *Proceedings of the Meeting on Algorithm Engineering & Expermiments*, pages 90–98. Society for Industrial and Applied Mathematics, 2010.

6 Michael Baur, Marco Gaertler, Robert Görke, Marcus Krug, and Dorothea Wagner. Generating graphs with predefined k-core structure. In *Proceedings of the European Conference of Complex Systems*. Citeseer, 2007.

7 Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. *Computational Complexity*, 21(2):311–358, 2012.

8 Talya Eden, Amit Levi, Dana Ron, and C Seshadhri. Approximately counting triangles in sublinear time. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 614–633. IEEE, 2015.

9 Talya Eden, Reut Levi, and Dana Ron. Testing bounded arboricity. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2081–2092, 2018. doi:10.1137/1.9781611975031.136.

10 Talya Eden, Dana Ron, and Will Rosenbaum. The Arboricity Captures the Complexity of Sampling Edges, 2019. arXiv:1902.08086.

11 Talya Eden, Dana Ron, and C. Seshadhri. Sublinear Time Estimation of Degree Distribution Moments: The Degeneracy Connection. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:13, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2017.7.

12 Talya Eden, Dana Ron, and C. Seshadhri. Faster sublinear approximations of k-cliques for low arboricity graphs. *CoRR*, abs/1811.04425, 2018. arXiv:1811.04425.

13 Talya Eden, Dana Ron, and C. Seshadhri. On approximating the number of k-cliques in sublinear time. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 722–734, 2018. doi:10.1145/3188745.3188810.

14 Talya Eden and Will Rosenbaum. Lower Bounds for Approximating Graph Parameters via Communication Complexity. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20-22, 2018 - Princeton, NJ, USA*, pages 11:1–11:18, 2018. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.11.

15 Talya Eden and Will Rosenbaum. On Sampling Edges Almost Uniformly. In Raimund Seidel, editor, *1st Symposium on Simplicity in Algorithms (SOSA 2018)*, volume 61 of *OpenAccess Series in Informatics (OASIcs)*, pages 7:1–7:9, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/OASIcs.SOSA.2018.7.

16 David Eppstein and Darren Strash. Listing all maximal cliques in large sparse real-world graphs. In *International Symposium on Experimental Algorithms*, pages 364–375. Springer, 2011.

17 Uriel Feige. On sums of independent random variables with unbounded variance and estimating the average degree in a graph. *SIAM Journal on Computing*, 35(4):964–984, 2006.

18 Gaurav Goel and Jens Gustedt. Bounded arboricity to determine the local structure of sparse graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 159–167. Springer, 2006.

19 Oded Goldreich and Dana Ron. Approximating average parameters of graphs. *Random Struct. Algorithms*, 32(4):473–493, 2008.

20 Mira Gonen, Dana Ron, and Yuval Shavitt. Counting stars and other small subgraphs in sublinear-time. *SIAM Journal on Discrete Mathematics*, 25(3):1365–1411, 2011.

21 C. St. JA. Nash-Williams. Edge-disjoint spanning trees of finite graphs. *Journal of the London Mathematical Society*, 1(1):445–450, 1961.

22 Kijung Shin, Tina Eliassi-Rad, and Christos Faloutsos. Patterns and anomalies in k-cores of real-world graphs with applications. *Knowledge and Information Systems*, 54(3):677–710, 2018.

A Nearly-Linear Time Algorithm for Submodular Maximization with a Knapsack Constraint

Alina Ene
Department of Computer Science, Boston University, MA, USA
aene@bu.edu

Huy L. Nguyen
College of Computer and Information Science, Northeastern University, Boston, MA, USA
hlnguyen@cs.princeton.edu

Abstract

We consider the problem of maximizing a monotone submodular function subject to a knapsack constraint. Our main contribution is an algorithm that achieves a nearly-optimal, $1 - 1/e - \epsilon$ approximation, using $(1/\epsilon)^{O(1/\epsilon^4)} n \log^2 n$ function evaluations and arithmetic operations. Our algorithm is impractical but theoretically interesting, since it overcomes a fundamental running time bottleneck of the multilinear extension relaxation framework. This is the main approach for obtaining nearly-optimal approximation guarantees for important classes of constraints but it leads to $\Omega(n^2)$ running times, since evaluating the multilinear extension is expensive. Our algorithm maintains a fractional solution with only a constant number of entries that are strictly fractional, which allows us to overcome this obstacle.

2012 ACM Subject Classification Theory of computation → Submodular optimization and polymatroids

Keywords and phrases submodular maximization, knapsack constraint, fast algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.53

Category Track A: Algorithms, Complexity and Games

Related Version `https://arxiv.org/abs/1709.09767`

Funding *Alina Ene*: Partially supported by NSF CAREER grant CCF-1750333 and NSF grant CCF-1718342.
Huy L. Nguyen: Partially supported by NSF CAREER grant CCF-1750716.

Acknowledgements This work was done in part while the authors were visiting the Simons Institute for the Theory of Computing.

1 Introduction

A set function $f : 2^V \to \mathbb{R}$ is *submodular* if for every $A, B \subseteq V$, we have $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$. Submodular functions naturally arise in a variety of contexts, both in theory and practice. Submodular functions capture many well-studied combinatorial functions including cut functions of graphs and digraphs, weighted coverage functions, as well as continuous functions including the Shannon entropy and log-determinants. Submodular functions are used in a wide range of application domains from machine learning to economics. In machine learning, it is used for document summarization [9], sensor placement [7], exemplar clustering [3], potential functions for image segmentation [4], etc. In an economics context, it can be used to model market expansion [2], influence in social networks [5], etc. The core mathematical problem underpinning many of these applications is the meta problem of maximizing a submodular objective function subject to some constraints.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 53; pp. 53:1–53:12


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

A common approach to submodular maximization is a two-step framework based on the multilinear extension F of f, a continuous function that extends f to the domain $[0,1]^V$. The program first (1) maximizes $F(x)$ subject to a continuous relaxation of the constraint and then (2) rounds the solution x to an integral vector satisfying the constraint. This paradigm has been very successful and it has led to the current best approximation algorithms for a wide variety of constraints including cardinality constraints, knapsack constraints, matroid constraints, etc. One downside with this approach is that in general, evaluating the multilinear extension is expensive and it is usually approximately evaluated. To achieve the desirable approximation guarantees, the evaluation error needs to be very small and in a lot of cases, the error needs to be $O(n^{-1})$ times the function value. Thus, even an efficient algorithm with $O(n)$ queries to the multilinear extension would require $\Omega(n^2)$ running time.

In this work, we develop a new algorithm that achieves $1 - 1/e - \epsilon$ approximation for maximizing a monotone submodular function subject to a knapsack constraint. The basic approach is still based on the multilinear extension but the algorithm ensures that the number of fractional coordinates is constant, which allows evaluating the multilinear extension exactly in constant number of queries to the original function. This approach allows us to bypass the obstructions discussed above and get nearly linear running time.

▶ **Theorem 1.** *There is an algorithm for maximizing a monotone submodular function subject to a knapsack constraint that achieves a $1 - 1/e - \epsilon$ approximation using $(1/\epsilon)^{O(1/\epsilon^4)} n \log n$ function evaluations and $(1/\epsilon)^{O(1/\epsilon^4)} n \log^2 n$ arithmetic operations.*

For simplicity, when stating running times, we assume that each call to the value oracle of f takes constant time, since for the algorithms discussed the number of evaluations dominates the running time up to logarithmic factors. Previously, Wolsey [11] gives an algorithm with a $1 - 1/e^\beta \approx 0.35$ approximation, where β is the unique root of the equation $e^x = 2 - x$. Building on the work of Khuller *et al.* for the maximum k-coverage problem [6], Sviridenko [10] gives an algorithm with a $1 - 1/e$ approximation that runs in $O(n^5)$ time. Badanidiyuru and Vondrak [1] give an algorithm with a $1 - 1/e - \epsilon$ approximation running in $n^2(\log n/\epsilon)^{O(1/\epsilon^8)}$ time. Our work builds on [1] and we discuss the relationship between the two algorithms in more detail in Section 1.1.

Kulik *et al.* [8] obtain a $1 - 1/e - \epsilon$ approximation for d knapsack constraints in time $\Omega(n^{d/\epsilon^4})$ that comes from enumerating over d/ϵ^4 items. The techniques in this paper could likely be extended to obtain an algorithm for the continuous problem of maximizing the multilinear extension subject to d knapsack constraints, with a running time that is exponential in d and nearly-linear in n. We leave it as an open problem whether the rounding can also be extended to multiple knapsack constraints.

Remark on the algorithm of [1]. We note that there are some technical issues in the algorithm proposed in [1] for a knapsack constraint. The main issue, which was pointed out by Yoshida [12], arises in the partitioning of the items into large and small items: an item e is small if it has value $f(\{e\}) \leq \epsilon^6 f(\mathrm{OPT})$ and cost $c_e \leq \epsilon^4$, and it is large otherwise. The algorithm enumerates the marginal values of the large items and thus the set of large items was intended to be of size $\mathrm{poly}(1/\epsilon)$. But this may not be true in general, as there could be many items in OPT with singleton value greater than $\epsilon^6 f(\mathrm{OPT})$. On the other hand, the assumption that the small items have small singleton values is crucial to ensuring that the algorithm obtains a good value from the small items. Another issue arises in the rounding algorithm. The fractional solution is rounded using a rounding algorithm for a partition matroid that treats the parts independently. But in this setting an item participates in several parts and we need to ensure that it is not selected more than once.

1.1 Our techniques

As in the classical knapsack problem with a linear objective, the algorithms achieving optimal approximation are based on enumeration techniques. One such approach is to enumerate the most valuable items in OPT (in the submodular problem, we can determine which items of OPT are valuable based on the Greedy ordering of OPT, see (1)) and greedily pack the remaining items based on the marginal gain to cost density. This approach leads to the optimal $1 - 1/e$ approximation provided that we enumerate 3 items [10]. The running time of the resulting algorithm is $O(n^5)$ and it can be improved to $O(n^4 \log(n/\epsilon)/\epsilon)$ time at a loss of ϵ in the approximation.

A different approach, inspired by the algorithms for the classical knapsack problem that use dynamic programming over the (appropriately discretized) profits of the items, is to enumerate over the marginal gains of the valuable items of OPT. Unlike the classical setting with linear profits, it is considerably more challenging to leverage such an approach in the submodular setting. Badanidiyuru and Vondrak [1] propose a new approach based on this enumeration technique and continuous density Greedy with a running time of $n^2 \left(\frac{\log n}{\epsilon}\right)^{O\left(\frac{1}{\epsilon^8}\right)}$, which overcame the $\Omega(n^4)$ running time barrier for the approaches that are based on enumerating items.

In this work, we build on the approach introduced by [1] and we obtain a faster running time of $\left(\frac{1}{\epsilon}\right)^{O\left(\frac{1}{\epsilon^4}\right)} n \log^2 n$. Our algorithm is impractical due to the high dependency on ϵ, but it is theoretically interesting. Obtaining near-optimal approximations in nearly-linear time for submodular maximization has been out of reach for all but a cardinality constraint.

Obtaining a fast running time poses several conceptual and technical challenges, and we highlight some of them here. Let us denote the valuable items of OPT as OPT_1, and let $\mathrm{OPT}_2 = \mathrm{OPT} \setminus \mathrm{OPT}_1$. For our algorithm, the set OPT_1 has $\mathrm{poly}(1/\epsilon)$ items and we can handle them by enumerating over their marginal gains, appropriately discretized. Similarly to [1], we use the guessed marginal gains to pack items that are competitive with OPT_1: for each guessed marginal gain, we find the cheapest item whose marginal gain is at least the guessed value, and we add ϵ of the item to the fractional solution. The continuous approach is necessary for ensuring that we obtain a good approximation, but it is already introducing the following conceptual and technical difficulties:

- *We do not know how much budget is available for the remaining items.* Since we packed the items fractionally, we will need to perform the rounding to find out which of the items will be in the final solution and their total budget. But we cannot do the rounding before packing the remaining items. Additionally, we cannot afford to guess the budget of OPT_1, even approximately.
- *In the continuous setting, evaluating the multilinear extension takes $\Omega(n^2)$ time in general.*
- *We will need to ensure that we can round the resulting fractional solution.*

A key idea in our algorithm, and an important departure from the approach of [1], is to *integrally* pack the remaining items using density Greedy with lazy evaluations to obtain a nearly-linear running time. The resulting fractional solution has only a constant number of entries that are strictly fractional, and we show that this is beneficial both in terms of running time and rounding: we can evaluate the multilinear extension in constant time and we can exploit the special structure of the solution to round. However, the first difficulty mentioned above remains a significant conceptual barrier for realizing this plan: if we cannot get a handle on how much budget to allocate to density Greedy, we will not be able to round the solution without violating the budget or losing value. Our solution here is based on the following insights.

Algorithm 1 KNAPSACK(f, ϵ).

$t \leftarrow 1/\epsilon^3$
$r \leftarrow 1/\epsilon$
$M \leftarrow \Theta(f(\mathrm{OPT}))$
$S_{\mathrm{best}} \leftarrow \emptyset$
Try all possible sequences:
  $\{v_{p,i}\}$: $p \in \{1, 2, \ldots, 1/\epsilon\}$, $i \in \{1, 2, \ldots, t\}$, $v_{p,i} \in \{0, \epsilon M/t, 2\epsilon M/t, \ldots, 1\}$
  $\{W_p\}$: $p \in \{1, 2, \ldots, 1/\epsilon\}$, $W_p \in \{0, \epsilon M, 2\epsilon M, \ldots, M\}$
  $\{w_{p,i}\}$: $p \in \{1, 2, \ldots, 1/\epsilon\}$, $i \in \{1, 2, \ldots, r+1\}$, $w_{p,i} \in \{0, \epsilon^2 W_p/r, 2\epsilon^2 W_p/r, \ldots, W_p\}$
for every choice $\{v_{p,i}\}$, $\{W_p\}$, $\{w_{p,i}\}$ **do**
  $x \leftarrow \mathrm{KNAPSACKGUESS}(f, \epsilon, \{v_{p,i}\}, \{W_p\}, \{w_{p,i}\})$
  $S \leftarrow \mathrm{ROUND}(x)$
  if $f(S) > f(S_{\mathrm{best}})$ **then**
    $S_{\mathrm{best}} \leftarrow S$
  end if
end for
Return S_{best}

First, note that we may assume that every item in OPT_2 has a cost that is small relative to the total budget of OPT_2: there can only be a small number of heavy items and each of them has small marginal gain on top of OPT_1, and thus we can discard them without losing too much in the approximation. Moreover, if there are no heavy items at all, we can show that density Greedy will not exceed the budget. Thus, if we knew the budget of OPT_2, we could remove all of the heavy items and run density Greedy on the remaining items.

Unfortunately, we cannot guess the budget of OPT_2 since there are too many possible choices. Instead, note that, since the cost of an item is its marginal value divided by its density, a heavy item has large value or small density. If it has small density then intuitively Greedy will not pick it. The problematic items are the ones that have large marginal values, as density Greedy may pick them and they may be too heavy. Unfortunately, we cannot filter out all the items with large marginal value, since those items may include items in OPT_2 (note that even though every item in OPT_2 has small marginal value on top of OPT_1, it can have large marginal value on top of our current fractional solution that does not necessarily contain OPT_1). Now the key observation is that the number of such items is small, and we can handle them with additional guessing.

The final step of the algorithm is to round the fractional solution to a feasible integral solution. Here we take advantage of the fact that the only entries that are strictly fractional were introduced in the OPT_1 stages of the algorithm. The fractional items can be mapped to the items in OPT_1 in such a way that every item in OPT_1 is assigned a fractional mass of at most 1 coming from items with smaller or equal cost. Thus, for each item in OPT_1, we want to select one of the items fractionally assigned to it. This is reminiscent of a partition matroid and thus a natural approach is to use a matroid rounding algorithm such as pipage rounding or swap rounding. However, an item may be fractionally assigned to more than one item in OPT_1, and we need to ensure that the rounding does not select the same item for different items in OPT_1. We show that we can do so using a careful application of swap rounding.

Algorithm 2 KnapsackGuess$(f, \epsilon, \{v_{p,i}\}, \{W_p\}, \{w_{p,i}\})$.

$t \leftarrow 1/\epsilon^3$
$r \leftarrow 1/\epsilon$
$x_0 \leftarrow 0$
for $p = 1, 2, \ldots, 1/\epsilon$ **do**
  $y^{(p,0)} \leftarrow x_{p-1}$
  $A_p \leftarrow \emptyset$
  for $i = 1, 2, \ldots, t$ **do**
    $a_{p,i} \leftarrow$ element with minimum size c_e in $\{e \notin A_p \colon F(y^{(p,i-1)} \vee \mathbf{1}_e) - F(y^{(p,i-1)}) \geq v_{p,i}\}$
    $y^{(p,i)} \leftarrow y^{(p,i-1)} + \epsilon \mathbf{1}_{a_{p,i}}$
    $A_p \leftarrow A_p \cup \{a_{p,i}\}$
  end for
  if $W_p = 0$ **then**
    Continue to the next phase $p + 1$
  end if
  $z^{(p,0)} \leftarrow y^{(p,t)}$
  $B_p \leftarrow \emptyset$
  Let r_p be the smallest $i \in \{0, 1, \ldots, r\}$ such that $w_{p,i+1} \leq \epsilon(1-\epsilon)W_p/r$. If no such i exists, let $r_p = r$. ⟨⟨ r_p `is the number of large value elements in` OPT_2 ⟩⟩
  for $i = 1, 2, \ldots, r_p$ **do**
    $b_{p,i} \leftarrow$ element with minimum size c_e in $\{e \colon F(z^{(p,i-1)} \vee \mathbf{1}_e) - F(z^{(p,i-1)}) \geq w_{p,i}\}$
    $z^{(p,i)} \leftarrow z^{(p,i-1)} \vee \mathbf{1}_{b_{p,i}}$
    $B_p \leftarrow B_p \cup \{b_{p,i}\}$
    if $F(z^{(p,i)}) - F(z^{(p,0)}) \geq \epsilon(1 - 12\epsilon)W_p$ **then**
      Set $x_p \leftarrow z^{(p,i)}$ and continue to phase $p + 1$
    end if
  end for
  if $F(z^{(p,r_p)}) - F(z^{(p,0)}) < \epsilon(1 - 12\epsilon)W_p$ **then**
    $V' \leftarrow V \setminus \{e \colon F(z^{(p,r_p)} \vee \mathbf{1}_e) - F(z^{(p,r_p)}) \geq \epsilon W_p/r\}$
    $C_p \leftarrow$ DensityGreedy$(f, z^{(p,r_p)}, \epsilon(1 - 12\epsilon)W_p - F(z^{(p,r_p)}) + F(z^{(p,0)}), V')$
    $x_p \leftarrow z^{(p,r_p)} \vee \mathbf{1}_{C_p}$
  end if
end for
Return $x_{1/\epsilon}$

2 The algorithm

We consider the problem of maximizing a monotone submodular function subject to a single knapsack constraint. Each element $e \in V$ has a cost $c_e \in \mathbb{R}_+$, and the goal is to find a set $\mathrm{OPT} \in \mathrm{argmax}\{f(S) \colon \sum_{e \in S} c_e \leq 1\}$. We assume that the knapsack capacity is 1, which we may assume without loss of generality by scaling the cost of each element by the knapsack capacity. We let $F : [0,1]^V \to \mathbb{R}_+$ denote the multilinear extension f. For every $x \in [0,1]^V$, we have

$$F(x) = \sum_{S \subseteq V} f(S) \prod_{e \in S} x_e \prod_{e \notin S} (1 - x_e) = \mathbb{E}[f(R(x))],$$

where $R(x)$ is a random set that includes each element $e \in V$ independently with probability x_e. For two vectors x and y, we let $x \vee y$ denote the vector such that $(x \vee y)_i = \max\{x_i, y_i\}$ for all $i \in V$.

We fix an optimal solution to the problem that we denote by OPT. We assume that the algorithm knows a constant approximation of $f(\mathrm{OPT})$; such an approximation can be obtained in nearly linear time by taking the best of the following two solutions: the solution obtained by running Density Greedy (implemented using lazy evaluations, similarly to Algorithm 3) and the solution consisting of the best single element. Let $f(\mathrm{OPT}) \geq M \geq (1-\epsilon)f(\mathrm{OPT})$ denote the algorithm's guess for the optimal value. There are $O(1/\epsilon)$ choices for M given the constant approximation of $f(\mathrm{OPT})$.

We order OPT as $o_1, o_2, \ldots, o_{|\mathrm{OPT}|}$, where

$$o_i = \mathrm{argmax}_{o \in \mathrm{OPT}}(f(\{o_1, \ldots, o_{i-1}\} \cup \{o\}) - f(\{o_1, \ldots, o_{i-1}\})) \tag{1}$$

Let $t = O(1/\epsilon^3)$, $\mathrm{OPT}_1 = \{o_1, o_2, \ldots, o_t\}$, and $\mathrm{OPT}_2 = \mathrm{OPT} \setminus \mathrm{OPT}_1$.

We emphasize that we use the above ordering of OPT and the partition of OPT into OPT_1 and OPT_2 only for the analysis and to motivate the choices of the algorithm. In particular, the algorithm does not know this ordering or partition.

It is useful to filter out from OPT_2 the items that have large cost, more precisely, cost greater than $\epsilon^2(1 - c(\mathrm{OPT}_1))$. Since every element $o \in \mathrm{OPT}_2$ satisfies $f(\mathrm{OPT}_1 \cup \{o\}) - f(\mathrm{OPT}_1) \leq \epsilon^3 f(\mathrm{OPT}_1)$ and there are at most $1/\epsilon^2$ such elements, this will lead to only an $\epsilon f(\mathrm{OPT})$ loss. For ease of notation, we use OPT_2 to denote the set without these elements, i.e., we assume that $c_o \leq \epsilon^2(1 - c(\mathrm{OPT}_1))$ for every $o \in \mathrm{OPT}_2$.

Algorithm 1 gives a precise description of the algorithm. The algorithm guesses a sequence of values as follows.

Guessed values. Throughout the paper, we assume for simplicity that $1/\epsilon$ is an integer. Recall that $t = 1/\epsilon^3$. Let $r = 1/\epsilon$ (r is an upper bound on the number of items of OPT_2 that have large marginal value in each phase).

- A sequence $\{v_{1,1}, v_{1,2}, \ldots, v_{1/\epsilon,t}\}$ where $v_{p,i} \in \{0, \epsilon M/t, 2\epsilon M/t, \ldots, M\}$ is an integer multiple of $\epsilon M/t$, for all integers p and i such that $1 \leq p \leq 1/\epsilon$ and $1 \leq i \leq t$. The value $v_{p,i}$ is an approximate guess for the marginal value of $o_i \in \mathrm{OPT}_1$ during phase p. There are $t/\epsilon + 1 = 1/\epsilon^4 + 1$ choices for each $v_{p,i}$ and thus there are $(1/\epsilon^4 + 1)^{1/\epsilon^4} = (1/\epsilon)^{O(1/\epsilon^4)}$ possible sequences.
- A sequence $\{W_1, W_2, \ldots, W_{1/\epsilon}\}$ where $W_p \in \{0, \epsilon M, 2\epsilon M, \ldots, M\}$ is an integer multiple of ϵM, for all integers p such that $1 \leq p \leq 1/\epsilon$. The value W_p is an approximate guess for the total marginal value of OPT_2 in phase p. There are $1/\epsilon + 1$ choices for each W_p and thus there are $(1/\epsilon + 1)^{1/\epsilon}$ possible sequences.
- A sequence $\{w_{1,1}, w_{1,2}, \ldots, w_{1/\epsilon,1/\epsilon+1}\}$ where $w_{p,i} \in \{0, \epsilon^2 W_p/r, 2\epsilon^2 W_p/r, \ldots, W_p\}$ is an integer multiple of $\epsilon^2 W_p/r$, for all integers p and i such that $1 \leq p, i \leq 1/\epsilon$ (the value W_p is the same as in the sequence above). The values $w_{p,i}$, where $i \in \{1, 2, \ldots, 1/\epsilon\}$, are approximate guesses for the marginal values of the items in OPT_2 with large marginal value in phase p. There are $r/\epsilon^2 + 1 = 1/\epsilon^3 + 1$ choices for each $w_{p,i}$ and thus there are $(1/\epsilon)^{O(1/\epsilon^2)}$ possible sequences.

The algorithm enumerates all possible such sequences. For each choice, the algorithm works as follows. Let $\{v_{p,i}\}$, $\{W_p\}$, and $\{w_{p,i}\}$ denote the current sequences. The algorithm performs $1/\epsilon$ phases. Each phase is comprised of three stages, executed in sequence in this order: an OPT_1 stage, a stage for the large value items in OPT_2, and a Density Greedy stage. We describe each of these stages in turn.

The $\mathbf{OPT_1}$ stage of phase p. This stage uses the values $\{v_{p,i} : 1 \le i \le t\}$ as follows. We perform t iterations. In each iteration i, we consider the items not selected in previous iterations that have marginal value at least $v_{p,i}$ on top of the current solution, i.e., $F(x \vee \mathbf{1}_e) - F(x) \ge v_{p,i}$. Among these items, we select the item with minimum cost and increase its fractional value by ϵ. Together, the t iterations select t different items and increase their fractional value by ϵ.

The stage of phase p for the large value items in $\mathbf{OPT_2}$. This stage uses the value W_p and the values $\{w_{p,i} : 1 \le i \le 1/\epsilon\}$ as follows. We perform at most r iterations. In each iteration i, we find the minimum cost element that has marginal value at least $w_{p,i}$ on top of the current solution, and we integrally select this item. (Note that this is similar to the OPT_1 stage, except that we select items integrally.) At the end of the stage, if the items selected in this phase have total marginal gain at least $\epsilon(1-12\epsilon)W_p$, then we end phase p and proceed to the next phase. Otherwise, the algorithm proceeds to the Density Greedy stage.

The Density Greedy stage of phase p. If the previous stage did not reach a total marginal gain of at least $\epsilon(1-12\epsilon)W_p$, we run the discrete Density Greedy algorithm until we reach a gain of $\epsilon(1-12\epsilon)W_p$. Before running Density Greedy, we remove from consideration all elements whose marginal value is at least $\epsilon W_p/r$. In every step, the Density Greedy algorithm fully selects the item with largest density, i.e., ratio of marginal value to cost.

In order to achieve nearly linear time, we implement the Density Greedy algorithm using approximate lazy evaluations as shown in Algorithm 3. We maintain the items in a priority queue sorted by density. We initialize the marginal values and the densities with respect to the initial solution. In each iteration of the algorithm, we find an item whose density with respect to the current solution is within a factor of $(1-\epsilon)$ of the maximum density as follows. We remove the item at the top of the queue. The marginal value of the item may be stale, so we evaluate its marginal gain with respect to the current solution. If the new marginal gain is within a factor of $(1-\epsilon)$ of the old marginal gain, it follows from submodularity that the density of the item is within a factor of $(1-\epsilon)$ of the maximum density, and we select the item. If the marginal gain has changed by a factor larger than $(1-\epsilon)$, we update the density and reinsert the item in the queue. We also keep track of how many times each item's density has been updated and, if an item has been updated more than $2\ln(n/\epsilon)/\epsilon$ times, we discard the item since it can no longer contribute a significant value to the solution.

Rounding the fractional solution. After $1/\epsilon$ phases, we obtain a fractional solution with $O(1/\epsilon^4)$ fractional entries. We round the resulting fractional solution to an integral solution using swap rounding, as shown in Algorithm 4.

The following theorem states our main result for the fractional solution. We will use the second guarantee to obtain a fast rounding algorithm (see Section 3). We defer the proof of the theorem to the full version of the paper.

▶ **Theorem 2.** *There are choices for the guessed values $\{v_{p,i}\}$, $\{W_p\}$, and $\{w_{p,i}\}$ for which Algorithm 2 returns a fractional solution x with the following properties:*

(1) $F(x) \ge \left(1 - \frac{1}{e} - O(\epsilon)\right) f(\mathrm{OPT})$;

(2) *Let E be the set of all items $e \in V$ such that $0 < x_e < 1$. There exists a mapping $\sigma : E \times \{1, 2, \ldots, 1/\epsilon\} \to \mathrm{OPT}_1$ with the following properties:*

- **(a)** *For every element $e \in E$ and every phase $p \in \{1, 2, \ldots, 1/\epsilon\}$ such that $e \in A_p$, $\sigma(e,p)$ is defined and $c(e) \le c(\sigma(e,p))$.*
- **(b)** *For every element $o \in \mathrm{OPT}_1$, there are at most $1/\epsilon$ pairs (e,p) such that $\sigma(e,p) = o$.*

Algorithm 3 LazyDensityGreedy(f, x, W, V').

$S_0 \leftarrow \emptyset$
$D \leftarrow \emptyset$
$u(e) \leftarrow 0$ for all $e \in V'$
$v(e) \leftarrow F(x \vee \mathbf{1}_e) - F(x)$ for all $e \in V'$
Maintain the elements in a priority queue sorted in decreasing order by key, where the key of each element e is initialized to its density $\frac{v(e)}{c(e)}$
for $i = 1, 2, \ldots$ **do**
  while true **do**
    if queue is empty **then**
      return S_{i-1}
    end if
    Remove the element e from the priority queue with maximum key
    $v'(e) \leftarrow F(x \vee \mathbf{1}_{S_{i-1} \cup \{e\}}) - F(x \vee \mathbf{1}_{S_{i-1}})$
    $u(e) \leftarrow u(e) + 1$
    if $v(e) \geq (1 - \epsilon)v'(e)$ **then**
      $e_i \leftarrow e$
      $v(e) \leftarrow v'(e)$
      $S_i \leftarrow S_{i-1} \cup \{e_i\}$
      if $F(x \vee 1_{S_i}) - F(x) \geq W$ **then**
        return S_i
      end if
      Exit the while loop and continue to iteration $i + 1$
    else
      if $u(e) \leq \frac{2\ln(n/\epsilon)}{\epsilon}$ **then**
        $v(e) \leftarrow v'(e)$
        Reinsert e into the queue with key $\frac{v'(e)}{c(e)}$
      else
        $D \leftarrow D \cup \{e\}$
      end if
    end if
  end while
end for

3 Rounding algorithm and analysis of the final solution

In this section, we analyze the rounding algorithm (Algorithm 4) that rounds the fractional solution x guaranteed by Theorem 2. We round the fractional entries of x as follows. We initialize $\hat{x} = x$. For analysis purposes, we initialize $O = \mathrm{OPT}_1$. We sort the fractional elements in non-increasing order according to their cost. While there are fractional elements, we repeatedly move fractional mass between the two elements with highest cost as follows. Let e_1 and e_2 be the fractional elements with the highest and second-highest cost, respectively. We consider two cases:

Case 1: $\hat{x}_{e_1} + \hat{x}_{e_2} \leq 1$. With probability $\hat{x}_{e_1}/(\hat{x}_{e_1} + \hat{x}_{e_2})$, we update $\hat{x}_{e_1} \leftarrow \hat{x}_{e_1} + \hat{x}_{e_2}$ and $\hat{x}_{e_2} \leftarrow 0$; with the remaining probability, we update $\hat{x}_{e_2} \leftarrow \hat{x}_{e_1} + \hat{x}_{e_2}$ and $\hat{x}_{e_1} \leftarrow 0$. If an element becomes integral, we remove it from the list. For analysis purposes, if an element is rounded up to 1, we pair it up with the element $o_1 \in O$ with highest cost, and we update $O \leftarrow O \setminus \{o_1\}$.

Algorithm 4 ROUND(x).

Let $\sigma_1, \ldots, \sigma_k$ be the fractional coordinates of x.
Sort $\sigma_1, \ldots, \sigma_k$ so that $c_{\sigma_1} \leq c_{\sigma_2} \leq \cdots \leq c_{\sigma_k}$.
while $k > 0$ **do**
  if $k = 1$ **then**
    $x_{\sigma_1} \leftarrow 1$
    return x
  end if
  if $x_{\sigma_k} + x_{\sigma_{k-1}} > 1$ **then**
    Pick $u \in \{0, 1\}$ randomly such that $\Pr[u = 1] = \frac{1 - x_{\sigma_{k-1}}}{2 - x_{\sigma_k} - x_{\sigma_{k-1}}}$
    if $u = 1$ **then**
      $x_{\sigma_k} \leftarrow 1$
      $x_{\sigma_{k-1}} \leftarrow x_{\sigma_{k-1}} + x_{\sigma_k} - 1$
      $k \leftarrow k - 1$
    else
      $x_{\sigma_{k-1}} \leftarrow 1$
      $x_{\sigma_k} \leftarrow x_{\sigma_{k-1}} + x_{\sigma_k} - 1$
      $\sigma_{k-1} \leftarrow \sigma_k$
      $k \leftarrow k - 1$
    end if
  else
    Pick $u \in \{0, 1\}$ randomly such that $\Pr[u = 1] = \frac{x_{\sigma_k}}{x_{\sigma_k} + x_{\sigma_{k-1}}}$
    if $u = 1$ **then**
      $x_{\sigma_k} \leftarrow x_{\sigma_{k-1}} + x_{\sigma_k}$
      $x_{\sigma_{k-1}} \leftarrow 0$
      $\sigma_{k-1} \leftarrow \sigma_k$
      $k \leftarrow k - 1$
    else
      $x_{\sigma_{k-1}} \leftarrow x_{\sigma_{k-1}} + x_{\sigma_k}$
      $x_{\sigma_k} \leftarrow 0$
      $k \leftarrow k - 1$
    end if
    if $x_{\sigma_k} = 1$ **then**
      $k \leftarrow k - 1$
    end if
  end if
end while

Case 2: $\hat{x}_{e_1} + \hat{x}_{e_2} > 1$. With probability $(1 - \hat{x}_{e_2})/(2 - \hat{x}_{e_1} - \hat{x}_{e_2})$, we update $\hat{x}_{e_1} \leftarrow 1$ and $\hat{x}_{e_2} \leftarrow \hat{x}_{e_1} + \hat{x}_{e_2} - 1$; with the remaining probability, we update $\hat{x}_{e_2} \leftarrow 1$ and $\hat{x}_{e_1} \leftarrow \hat{x}_{e_1} + \hat{x}_{e_2} - 1$. If an element becomes integral, we remove it from the list. For analysis purposes, if an element is rounded up to 1, we pair it up with an element in O as follows. If the element e_1 with the highest cost is rounded up to 1, we pair up e with the element $o_1 \in O$ with highest cost, and we update $O \leftarrow O \setminus \{o_1\}$. If the element e_2 with the second-highest cost is rounded up to 1, we pair up e_2 with the element $o_2 \in O$ with the second-highest cost, and we update $O \leftarrow O \setminus \{o_2\}$.

If there is only one fractional entry then we can round this entry up to 1 and pair up this element with the element $o_1 \in O$ with highest cost.

We now turn to the analysis of the rounding. We first show that the expected value of the rounded solution is at least $F(x)$. We then show that the cost of the fractional elements that were rounded up to 1 is at most $c(\mathrm{OPT}_1)$, thus ensuring that the final rounded solution is feasible.

▶ **Lemma 3.** $\mathbb{E}[F(\hat{x})] \geq F(x)$.

Proof. Note that each iteration updates the solution as follows: $\hat{x}' = \hat{x} + \delta(\mathbf{1}_{e_1} - \mathbf{1}_{e_2})$, where δ is a random value satisfying $\mathbb{E}_\delta[\hat{x}'] = \hat{x}$. The multilinear extension is convex along the direction $\mathbf{1}_e - \mathbf{1}_{e'}$ for every pair of elements e and e'. Therefore $\mathbb{E}_\delta[F(\hat{x}')] \geq F(\mathbb{E}_\delta[\hat{x}']) = F(\hat{x})$, and the claim follows by induction. ◀

▶ **Lemma 4.** *Let $\hat{E}$ be the set of elements corresponding to the fractional entries that were rounded to* 1. *We have $c(\hat{E}) \leq c(\mathrm{OPT}_1)$.*

Proof. The lemma follows from the following invariant maintained by the algorithm for the partially rounded solution $\hat{x}$ and the set $O \subseteq \mathrm{OPT}_1$:

Invariant: Let $o_1, o_2, \ldots, o_p$ be the elements of O, labeled such that $c_{o_1} \geq c_{o_2} \geq \cdots \geq c_{o_p}$. Let $e_1, e_2, \ldots, e_\ell$ be the elements corresponding to the fractional entries of $\hat{x}$, labeled such that $c_{e_1} \geq c_{e_2} \geq \cdots \geq c_{e_\ell}$. We define the following grouping of the elements $e_1, e_2, \ldots, e_\ell$ where each group contributes a fractional mass of 1 and each element belongs to at most two groups. Consider the interval $[0, \sum_{i=1}^{\ell} x_{e_i}]$ that is divided among the elements as follows: $[0, x_{e_1})$ corresponds to e_1 and, for all $2 \leq i \leq \ell$, $[\sum_{j=1}^{i-1} x_{e_j}, \sum_{j=1}^{i} x_{e_j})$ corresponds to e_i. The elements that overlap with the interval $[i-1, i)$ define the i-th group. The invariant is that $\hat{x}$ and O satisfy the following properties:

(1) $\sum_{i=1}^{\ell} \hat{x}_{e_i} \leq |O|$, and

(2) for every $i \geq 1$ and each element e in the i-th group, we have $c_e \leq c_{o_i}$.

We will show the invariant using induction on the number of iterations. We start by showing the invariant at the beginning of the rounding algorithm. We can show the invariant for x and OPT_1 using Theorem 2.

▷ Claim 5. The invariant holds for x and OPT_1.

Proof. Recall that each phase p of the KnapsackGuess algorithm selects a set A_p of elements and it increases the values of each of these elements by ϵ. Thus the fractional value x_{e_i} of each element $e_i \in E$ is equal to ϵ times the number of phases p such that $e_i \in A_p$. Moreover, by Theorem 2, there is a mapping $\sigma : \{e_1, \ldots, e_\ell\} \times \{1, 2, \ldots, 1/\epsilon\} \to \mathrm{OPT}_1$ such that, for each phase p such that $e_i \in A_p$, $\sigma(e_i, p)$ exists and $c(e_i) \leq c(\sigma(e_i, p))$.

We can think of each element e_i having x_{e_i}/ϵ copies and each element $o \in \mathrm{OPT}_1$ having $|\sigma^{-1}(o)| \leq 1/\epsilon$ copies. By letting $\tilde{E}$ and $\tilde{O}$ be the copies of the elements in E and OPT_1 (respectively), we can equivalently view σ as a bijection between $\tilde{E}$ and $\tilde{O}$ with the property that, if $\sigma((e, i)) = (o, j)$ then $c(e) \leq c(o)$. We may also assume that the elements of O with the highest costs have $1/\epsilon$ copies, i.e., there exists an index p' such that $o_1, \ldots, o_{p'}$ have $1/\epsilon$ copies and $o_{p'+1}, \ldots, o_p$ have zero copies; we can ensure this property by reassigning pairs in $\tilde{E}$ to elements of O with higher cost. Thus, if we sort $\tilde{E}$ and $\tilde{O}$ in non-increasing order according to costs, σ maps the first $1/\epsilon$ elements of $\tilde{E}$ to o_1, the next $1/\epsilon$ elements to o_2, etc. Since the i-th consecutive block of $1/\epsilon$ elements of $\tilde{E}$ represents the fractional mass of the i-th group of elements, the second property of the invariant follows. The first property of the invariant follows from the fact that $\frac{\|x\|_1}{\epsilon} = |\tilde{E}| = |\tilde{O}| \leq \frac{|\mathrm{OPT}_1|}{\epsilon}$. ◁

Now consider some iteration of the rounding algorithm, and suppose that the invariant holds at the beginning of the iteration. The invariant guarantees that the total fractional mass $\|\hat{x}\|_1$ is at most $|O|$ and, if we sort the fractional elements in non-increasing order according to the cost, the first unit of fractional mass can be assigned to the element o_1 with highest cost in O, the next unit of fractional mass can be assigned to the element o_2 with second-highest cost in O, etc. We will use such an assignment to argue that the invariant is preserved.

Suppose we are in Case 1, i.e., $\hat{x}_{e_1} + \hat{x}_{e_2} \leq 1$, where e_1 and e_2 are the fractional elements with the highest and second-highest cost. Let o_1 be the element of O with the highest cost. Since $\hat{x}_{e_1} + \hat{x}_{e_2} \leq 1$, it follows from the invariant that the entire fractional mass of $\hat{x}_{e_1} + \hat{x}_{e_2}$ is assigned to o_1. Since the rounding step moves fractional mass between e_1 and e_2, this property will continue to hold after the rounding step. If neither e_1 nor e_2 is rounded to 1, the updated fractional solution clearly satisfies the invariant. Therefore we may assume that one of e_1, e_2 is rounded to 1, and thus we must have had $\hat{x}_{e_1} + \hat{x}_{e_2} = 1$ before the rounding. Since o_1 is assigned a fractional mass of 1 in total, e_1 and e_2 are the only elements assigned to o_1. Therefore, after removing o_1, e_1, and e_2, the remaining fractional entries and the set $O \setminus \{o_1\}$ satisfy the invariant.

Suppose we are in Case 2, i.e., $1 < \hat{x}_{e_1} + \hat{x}_{e_2} \leq 2$, where e_1 and e_2 are the fractional elements with the highest and second-highest cost, respectively. Let o_1 and o_2 be the elements of O with the highest and second-highest cost, respectively. It follows from the invariant that the fractional mass $\hat{x}_{e_1} + \hat{x}_{e_2}$ is assigned to o_1 and o_2 as follows: the 1 unit of fractional mass assigned to o_1 is comprised of $\hat{x}_{e_1}$ from e_1 and $1 - \hat{x}_{e_2}$ from e_2, and o_2 is assigned the remaining $\hat{x}_{e_1} + \hat{x}_{e_2} - 1$ fractional mass of e_2. The rounding step either rounds e_1 to 1 by moving $1 - \hat{x}_{e_1}$ mass from e_2 to e_1 or it rounds e_2 to 1 by moving $1 - \hat{x}_{e_2}$ mass from e_1 to e_2. In the former case, after removing e_1 and o_1, the remaining fractional entries and the set $O \setminus \{o_1\}$ satisfy the invariant. Therefore we may assume that it is the latter, i.e., we round e_2 to 1 and we remove e_2 and o_2. In this case, the fractional values on the elements $e_3, e_4, \ldots$ move forward by $1 - \hat{x}_{e_2}$ to fill in the space vacated by e_2. We can also move forward their assignment to $O \setminus \{o_2\}$: e_1 remains entirely assigned to o_1 as before, and the assignment of each of the elements $e_3, e_4, \ldots$ is shifted forward. Since we remove one unit from both the total fractional mass and O, every remaining element becomes assigned to an element of $O \setminus \{o_2\}$ whose cost is at least as much as the element of O that it was previously assigned. Therefore the invariant is preserved. ◀

References

1. Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2014.
2. Shaddin Dughmi, Tim Roughgarden, and Mukund Sundararajan. Revenue Submodularity. *Theory of Computing*, 8(1):95–119, 2012.
3. Ryan Gomes and Andreas Krause. Budgeted Nonparametric Learning from Data Streams. In *International Conference on Machine Learning (ICML)*, pages 391–398, 2010.
4. Stefanie Jegelka and Jeff A. Bilmes. Submodularity beyond submodular energies: Coupling edges in graph cuts. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.
5. David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 137–146, 2003.
6. Samir Khuller, Anna Moss, and Joseph Seffi Naor. The budgeted maximum coverage problem. *Information processing letters*, 70(1):39–45, 1999.

7 Andreas Krause, Ajit Paul Singh, and Carlos Guestrin. Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. *Journal of Machine Learning Research*, 9:235–284, 2008.

8 Ariel Kulik, Hadas Shachnai, and Tami Tamir. Approximations for monotone and nonmonotone submodular maximization with knapsack constraints. *Mathematics of Operations Research*, 38(4):729–739, 2013.

9 Hui Lin and Jeff A. Bilmes. Multi-document Summarization via Budgeted Maximization of Submodular Functions. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, pages 912–920, 2010.

10 Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

11 Laurence A Wolsey. Maximising real-valued submodular functions: Primal and dual heuristics for location problems. *Mathematics of Operations Research*, 7(3):410–425, 1982.

12 Yuichi Yoshida. Maximizing a Monotone Submodular Function with a Bounded Curvature under a Knapsack Constraint. *CoRR*, abs/1607.04527, 2016. `arXiv:1607.04527`.

Towards Nearly-Linear Time Algorithms for Submodular Maximization with a Matroid Constraint

Alina Ene
Department of Computer Science, Boston University, MA, USA
aene@bu.edu

Huy L. Nguyen
College of Computer and Information Science, Northeastern University, Boston, MA, USA
hlnguyen@cs.princeton.edu

Abstract

We consider fast algorithms for monotone submodular maximization subject to a matroid constraint. We assume that the matroid is given as input in an explicit form, and the goal is to obtain the best possible running times for important matroids. We develop a new algorithm for a *general matroid constraint* with a $1 - 1/e - \epsilon$ approximation that achieves a fast running time provided we have a fast data structure for maintaining an approximately maximum weight base in the matroid through a sequence of decrease weight operations. We construct such data structures for graphic matroids and partition matroids, and we obtain the *first algorithms* for these classes of matroids that achieve a nearly-optimal, $1 - 1/e - \epsilon$ approximation, using a nearly-linear number of function evaluations and arithmetic operations.

2012 ACM Subject Classification Theory of computation → Submodular optimization and polymatroids

Keywords and phrases submodular maximization, matroid constraints, fast running times

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.54

Category Track A: Algorithms, Complexity and Games

Related Version `https://arxiv.org/abs/1811.07464`

Funding *Alina Ene*: Partially supported by NSF CAREER grant CCF-1750333 and NSF grant CCF-1718342.
Huy L. Nguyen: Partially supported by NSF CAREER grant CCF-1750716.

Acknowledgements This work was done in part while the authors were visiting the Simons Institute for the Theory of Computing.

1 Introduction

In this paper, we consider fast algorithms for monotone submodular maximization subject to a matroid constraint. Submodular maximization is a central problem in combinatorial optimization that captures several problems of interest, such as maximum coverage, facility location, and welfare maximization. The study of this problem dates back to the seminal work of Nemhauser, Wolsey and Fisher from the 1970's [20, 21, 12]. Nemhauser *et al.* introduced a very natural Greedy algorithm for the problem that iteratively builds a solution by selecting the item with the largest marginal gain on top of previously selected items, and they showed that this algorithm achieves a $1 - 1/e$ approximation for a cardinality constraint and a $1/2$ approximation for a general matroid constraint. The maximum coverage problem is a special case of monotone submodular maximization with a cardinality constraint and it is $1 - 1/e$

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 54; pp. 54:1–54:14


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

hard to approximate [10], and thus the former result is optimal. Therefore the main question that was left open by the work of Nemhauser *et al.* is whether one can obtain an optimal, $1 - 1/e$ approximation, for a general matroid constraint.

In a celebrated line of work [6, 24], Calinescu *et al.* developed a framework based on continuous optimization and rounding that led to an optimal $1 - 1/e$ approximation for the problem. The approach is to turn the discrete optimization problem of maximizing a submodular function f subject to a matroid constraint into a continuous optimization problem where the goal is to maximize the multilinear extension F of f (a continuous function that extends f) subject to the matroid polytope (a convex polytope whose vertices are the feasible integral solutions). The continuous optimization problem can be solved approximately within a $1 - 1/e$ factor using a continuous Greedy algorithm [24] and the resulting fractional solution can be rounded to an integral solution without any loss [1, 6, 8]. The resulting algorithm achieves the optimal $1 - 1/e$ approximation in polynomial time.

Unfortunately, a significant drawback of this approach is that it leads to very high running times. Obtaining fast running times is a fundamental direction both in theory and in practice, due to the numerous applications of submodular maximization in machine learning, data mining, and economics [18, 17, 14, 16, 9]. This direction has received considerable attention [2, 11, 3, 19, 5, 7], but it remains a significant challenge for almost all matroid constraints.

Before discussing these challenges, let us first address the important questions on how the input is represented and how we measure running time. The algorithms in this paper as well as prior work assume that the submodular function is represented as a value oracle that takes as input a set S and returns $f(S)$. For all these algorithms, the number of calls to the value oracle for f dominates the running time of the algorithm (up to a logarithmic factor), and thus we assume for simplicity that each call takes constant time.

The algorithms fall into two categories with respect to how the matroid is represented: the *independence oracle* algorithms assume that the matroid is represented using an oracle that takes as input a set S and returns whether S is feasible (independent); the *representable matroid* algorithms assume that the matroid is given as input in an explicit form. The representable matroid algorithms can be used for only a subclass of matroids, namely those that can be represented as a linear matroid over vectors in some field[1], but this class includes practically-relevant matroids such as the uniform, partition, laminar, graphical, and general linear matroids. The oracle algorithms apply to all matroids, but they are *unlikely to lead to the fastest possible running times:* even an ideal algorithm that makes only $O(k)$ independence calls has a running time that is $\Omega(k^2)$ in the independence oracle model (each oracle call needs to read its input, which takes $\Theta(k)$ time in the worst case), even if the matroid is a representable matroid such as a partition or a graphic matroid. This is because Thus there have always been parallel lines of research for representable matroids and general matroids.

This work falls in the first category, i.e., we assume that the matroid is given as input in an explicit form, and the goal is to obtain the best possible running times. Note that, although all the representable matroids are linear matroids, it is necessary to consider each class separately, since they have very different running times to even verify if a given solution is feasible: for simple explicit matroids such as a partition or a graphic matroid, checking whether a solution is feasible takes $O(n)$ time, where n is the size of the ground set of the matroid; for general explicit matroids represented using vectors in some field, checking whether a solution is feasible takes $O(k^\omega)$ time, where k is the rank of the matroid and ω is the exponent for fast matrix multiplication.

[1] In a linear matroid, the ground set is a collection of n vectors and a subset of the vectors is feasible (independent) if the vectors are linearly independent.

Since in many practical settings only nearly-linear running times are feasible, an important question to address is:

For which matroid constraints can we obtain
a nearly-optimal $1 - 1/e - \epsilon$ *approximation in nearly-linear time?*

Prior to this work, the only example of such a constraint was a cardinality constraint. For a partition matroid constraint, the fastest running time is $\Omega(n^{3/2})$ in the worst case when $k = \Omega(n)$ [5]. For a graphical matroid constraint, no faster algorithms are known than a general matroid, and the running time is $\Omega(n^2)$ in the worst case when $k = \Omega(n)$ [3]. Obtaining a best-possible, nearly-linear running time has been very challenging even for these classes of matroids for the following reasons:

The continuous optimization is a significant time bottleneck. The continuous optimization problem of maximizing the multilinear extension subject to the matroid polytope is an integral component in all algorithms that achieve a nearly-optimal approximation guarantee. However, the multilinear extension is expensive to evaluate even approximately. To achieve the nearly-optimal approximation guarantees, the evaluation error needs to be very small and in a lot of cases, the error needs to be $O(n^{-1})$ times the function value. As a result, a single evaluation of the multilinear extension requires $\Omega(n)$ evaluations of f. Thus, even a very efficient algorithm with $O(n)$ queries to the multilinear extension would require $\Omega(n^2)$ running time.

Rounding the fractional solution is a significant time bottleneck as well. Consider a matroid constraint of rank k. The fastest known rounding algorithm is the swap rounding, which requires k swap operations: in each operation, the algorithm has two bases B_1 and B_2 and needs to find $x \in B_1, y \in B_2$ such that $B_1 \setminus \{x\} \cup \{y\}$ and $B_2 \setminus \{y\} \cup \{x\}$ are bases. The typical implementation is to pick some $x \in B_1$ and try all y in B_2, which requires us to check independence for k solutions. Thus, overall, the rounding algorithm checks independence for $\Omega(k^2)$ solutions. Furthermore, each feasibility check takes $\Omega(k)$ time just to read the input. Thus a generic rounding for a matroid takes $\Omega(k^3)$ time.

Thus, in order to achieve a fast overall running time, one needs fast algorithms for both the continuous optimization and the rounding. In this work, we provide such algorithms for partition and graphic matroids, and we obtain the first algorithms with nearly-linear running times. At the heart of our approach is a general, nearly-linear time reduction that reduces the submodular maximization problem to two data structure problems: maintain an approximately maximum weight base in the matroid through a sequence of decrease-weight operations, and maintain an independent set in the matroid that allows us to check whether an element can be feasibly added. This reduction applies to any representable matroid, and thus it opens the possibility of obtaining faster running times for other classes of matroids.

1.1 Our contributions

We now give a more precise description of our contributions. We develop a new algorithm for maximizing the multilinear extension subject to a *general matroid constraint* with a $1 - 1/e - \epsilon$ approximation that achieves a fast running time provided we have fast data structures with the following functionality:

- *A maximum weight base data structure:* each element has a weight, and the goal is to maintain an approximately maximum weight base in the matroid through a sequence of operations, where each operation can only decrease the weight of a single element;

- *An independent set data structure* that maintains an independent set in the matroid and supports two operations: add an element to the independent set, and check whether an element can be added to the independent set while maintaining independence.

▶ **Theorem 1.** *Let f be a monotone submodular function and let $\mathcal{M}$ be a matroid on a ground set of size n. Let F be the multilinear extension of f and $P(\mathcal{M})$ be the matroid polytope of $\mathcal{M}$. Suppose that we have a data structure for maintaining a maximum weight base and independent set as described above. There is an algorithm for the problem $\max_{x \in P(\mathcal{M})} F(x)$ that achieves a $1 - 1/e - \epsilon$ approximation using $O(n \operatorname{poly}(\log n, 1/\epsilon))$ calls to the value oracle for f, data structure operations, and additional arithmetic operations.*

Using our continuous optimization algorithm and additional results that are included in the full version of this paper, we obtain the first nearly-linear time algorithms for both the discrete and continuous problem with a graphic and a partition matroid constraint. In the graphic matroid case, the maximum weight base data structure is a dynamic maximum weight spanning tree (MST) data structure and the independent data structure is a dynamic connectivity data structure (e.g., union-find), and we can use existing data structures that guarantee a poly-logarithmic amortized time per operation [15, 13, 23]. For a partition matroid, we provide data structures with a constant amortized time per operation. We also address the rounding step and provide a nearly-linear time algorithm for rounding a fractional solution in a graphic matroid. A nearly-linear time rounding algorithm for a partition matroid was provided in [5].

▶ **Theorem 2.** *There is an algorithm for maximizing a monotone submodular function subject to a generalized partition matroid constraint that achieves a $1 - 1/e - \epsilon$ approximation using $O(n \operatorname{poly}(1/\epsilon, \log n))$ function evaluations and arithmetic operations.*

▶ **Theorem 3.** *There is an algorithm for maximizing a monotone submodular function subject to a graphic matroid constraint that achieves a $1 - 1/e - \epsilon$ approximation using $O(n \operatorname{poly}(1/\epsilon, \log n))$ function evaluations and arithmetic operations.*

Previously, the best running time for a partition matroid was $\Omega(n^{3/2} \operatorname{poly}(1/\epsilon, \log n))$ in the worst case when $k = \Omega(n)$ [5]. The previous best running time for a graphic matroid is the same as the general matroid case, which is $\Omega(n^2 \operatorname{poly}(1/\epsilon, \log n))$ in the worst case when $k = \Omega(n)$ [3].

As shown by Vondrak [24], there is an efficient reduction from the *submodular welfare maximization* problem to the submodular maximization problem with a partition matroid constraint. Using this reduction and our algorithm for a partition matroid, we obtain a nearly-linear time algorithm for welfare maximization as well.

▶ **Theorem 4.** *There is a $1 - 1/e - \epsilon$ approximation algorithm for submodular welfare maximization using $O(n\text{poly}(1/\epsilon, \log n))$ function evaluations and arithmetic operations.*

We conclude with a formal statement of the contributions made in this paper on which the results above are based.

▶ **Theorem 5.** *There is a dynamic data structure for maintaining a maximum weight base in a partition matroid through a sequence of decrease weight operations with an $O(1)$ amortized time per operation.*

▶ **Theorem 6.** *There is a randomized algorithm based on swap rounding for the graphic matroid polytope that takes as input a point x represented as a convex combination of bases and rounds it to an integral solution S such that $\mathbb{E}[f(S)] \geq F(x)$. The running time of the algorithm is $O(nt \log^2 n)$, where t is the number of bases in the convex combination of x.*

1.2 Technical overview

The starting point of our approach is the work [5]. They observed that the running time of the continuous algorithm using the multilinear extension of [3] depends on the value of the maximum weight base when the value is measured in the modular approximation $f'(S) = \sum_{e \in S} f(e)$. It is clear that this approximation is at least the original function and it can be much larger. They observed that the running time is proportional to the ratio between the maximum weight base when weights are measured using the modular approximation compared with the optimal solution when weights are measured using the original function. On the other hand, in the greedy algorithm, the gain in every greedy step is proportional to the maximum weight base when weights are measured using the modular approximation. Thus, the discrete greedy algorithm makes fast progress precisely when the continuous algorithm is slow and vice versa. Therefore, one can start with the discrete greedy algorithm and switch to the continuous algorithm when the maximum weight solution is small even when weights are measured using the modular approximation.

Our algorithm consists of two key components: (1) a fast dynamic data structure for maintaining an approximate maximum weight base through a sequence of greedy steps, and (2) an algorithm that makes only a small number of queries to the data structure. Even if fast dynamic data structures are available, previous algorithms including that of [5] cannot achieve a fast time, since they require $\Omega(nk)$ queries to the data structure: the algorithm of [5] maintains the marginal gain for every element in the current base and it updates them after each greedy step; since each greedy step changes the marginal gain of every element in the base, this approach necessitates $\Omega(k)$ data structure queries per greedy step.

Our new approach uses random sampling to ensure that the number of queries to the data structure is nearly-linear. After each greedy step, our algorithm randomly samples elements from the base to check and update the marginal gains. Because of sampling, it can only ensure that at least $1/2$ of the elements in every value range have good estimates of their values. However, this is sufficient for maintaining an approximate maximum weight base. The benefit is that the running time becomes much faster: the number of checks that do not result in updates is small and if we make sure that an update only happens when the marginal gain change by a factor $1 - \epsilon$ then the total number of updates is at most $O(n \log n/\epsilon)$. Thus we obtain an algorithm with only a nearly-linear number of data structure queries and additional running time for *any matroid constraint*.

Our approach reduces the algorithmic problem to the data structure problem of maintaining an approximate maximum weight base through a sequence of value updates. In fact, the updates are only decrement in the values and thus can utilize even the decremental data structures as opposed to fully dynamic ones. In the case of a partition matroid constraint, one can develop a simple ad-hoc solution. In the case of a graphic matroid, one can use classical data structures for maintaining minimum spanning trees [15].

In both cases, fast rounding algorithms are also needed. The work [5] gives an algorithm for the partition matroid. We give an algorithm for the graphic matroid based on swap rounding and classical dynamic graph data structures. To obtain fast running time, in each rounding step, instead of swapping a generic pair, we choose a pair involving a leaf of the spanning tree.

1.3 Basic definitions and notation

Submodular functions. Let $f : 2^V \to \mathbb{R}_+$ be a set function on a finite ground set V of size $n := |V|$. The function is *submodular* if $f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$ for all subsets $A, B \subseteq V$. The function is *monotone* if $f(A) \leq f(B)$ for all subsets $A \subseteq B \subseteq V$. We assume

that the function f is given as a value oracle that takes as input any set $S \subseteq V$ and returns $f(S)$. We let $F : [0,1]^V \to \mathbb{R}_+$ denote the multilinear extension f. For every $x \in [0,1]^V$, we have

$$F(x) = \sum_{S \subseteq V} f(S) \prod_{e \in S} x_e \prod_{e \notin S} (1 - x_e) = \mathbb{E}[R(x)],$$

where $R(x)$ is a random set that includes each element $e \in V$ independently with probability x_e.

Matroids. A matroid $\mathcal{M} = (V, \mathcal{I})$ on a ground set V is a collection $\mathcal{I}$ of subsets of V, called independent sets, that satisfy certain properties (see e.g., [22], Chapter 39). In this paper, we consider matroids that are given to the input to the algorithm. Of particular interest are the partition and graphic matroids. A generalized partition matroid is defined as follows. We are given a partition $V_1, V_2, \ldots, V_h$ of V into disjoint subsets and budgets $k_1, k_2, \ldots, k_h$. A set S is independent ($S \in \mathcal{I}$) if $|S \cap V_i| \leq k_i$ for all $i \in [h]$. We let $k = \sum_{i=1}^{h} k_i$ denote the rank of the matroid. A graphic matroid is defined as follows. We are given a connected graph on $k+1$ vertices and n edges. The independent sets of the matroid are the forests of this graph.

Additional notation. Given a set $S \in \mathcal{I}$, we let f_S denote the function $f_S : 2^{V \setminus S} \to \mathbb{R}_{\geq 0}$ such that $f_S(S') = f(S' \cup S) - f(S)$ for all $S' \subseteq V \setminus S$. We let $\mathcal{M}/S = (V \setminus S, \mathcal{I}')$ denote the matroid obtained by contracting S in $\mathcal{M}$, i.e., $S' \in \mathcal{I}'$ iff $S' \cup S \in \mathcal{I}$. We let $P(\mathcal{M})$ denote the matroid polytope of $\mathcal{M}$: $P(\mathcal{M})$ is the convex hull of the indicator vectors of the bases of $\mathcal{M}$, where a base is an independent set of maximum size.

Constant factor approximation to $f(\mathrm{OPT})$. Our algorithm needs a $O(1)$ approximation to $f(\mathrm{OPT})$. Such an approximation can be computed very efficiently (see e.g. [5], Lemma 3.2).

1.4 Paper organization

In Section 2, we describe our algorithm for the continuous optimization problem of maximizing the multilinear extension subject to a general matroid constraint, with the properties stated in Theorem 1. As discussed in the introduction, our algorithm uses certain data structures to achieve a fast running time. In the full version of the paper, we show how to obtain these data structures for partition and graphic matroids. By combining these data structures with the results of Section 2, we obtain nearly-linear time algorithms for the continuous problem of maximizing the multilinear extension subject to a partition and graphic matroid constraint. To obtain a fast algorithm for the discrete problem, we also need a fast algorithm to round the fractional solution. Buchbinder *et al.* [5] give a nearly-linear time rounding algorithm for a partition matroid. In the full version of the paper, we give a nearly-linear time rounding algorithm for a graphic matroid, and prove Theorem 6. These results together give Theorems 2 and 3.

2 The algorithm for the continuous optimization problem

In this section, we describe and analyze our algorithm for the problem $\max_{x \in P(\mathcal{M})} F(x)$ for a general matroid $\mathcal{M}$, and prove Theorem 1. The algorithm is given in Algorithm 1 and it combines the continuous Greedy algorithm of [3] with a discrete Greedy algorithm that we provide in this paper, building on [5].

Algorithm 1 Algorithm for the continuous problem $\max_{x \in P(\mathcal{M})} F(x)$.

1: **procedure** ContinuousMatroid($f, \mathcal{M}, \epsilon$)
2: $c' = \Theta(1/\epsilon)$, where the Θ hides a sufficiently large absolute constant
3: $S = \text{LazySamplingGreedy}(f, \mathcal{M}, \epsilon)$
4: $x = \text{ContinuousGreedy}(f_S, \mathcal{M}/S, c', \epsilon)$
5: **return** $\mathbf{1}_S \vee x$ $\langle\langle$ $x \vee y$ `is the vector` $(x \vee y)_i = \max\{x_i, y_i\}$ `for all` i $\rangle\rangle$
6: **end procedure**

The continuous Greedy algorithm. The algorithm used on line 4 is the algorithm of [3]. To obtain a fast running time, we use an independent set data structure to maintain the independent sets constructed by the algorithm. The data structure needs to support two operations: add an element to the independent set, and check whether an element can be added to the independent set while maintaining independence. For a partition matroid, such a data structure with $O(1)$ time per operation is trivial to obtain. For a graphic matroid, we can use a union-find data structure [13, 23] with a $O(\log^* k)$ amortized time per operation.

▶ **Lemma 7** (Corollary 3.1 in [5]; [3]). *When run with values c and δ as input,* ContinuousGreedy *uses $O(n \ln(n/\delta)/\delta^2)$ independent set data structure operations, and $O(cn \ln^2(n/\delta)/\delta^4)$ queries to the value oracle of f and additional arithmetic operations. Moreover, if $\max_{S \in \mathcal{I}} \sum_{e \in S} f(e) \leq c \cdot f(\text{OPT})$, where $\text{OPT} \in \text{argmax}_{S \in \mathcal{I}} f(S)$, the solution x returned by the algorithm satisfies $F(x) \geq (1 - \frac{1}{e} - \delta) f(\text{OPT})$.*

The discrete Greedy algorithm is given in Algorithm 2. The algorithm works for any matroid constraint for which we can provide a fast data structure for maintaining a maximum weight base (note that the base is only an approximate maximum weight base, and we drop the word approximate for simplicity). We now describe the properties we require from this data structure. As discussed in the introduction, we give such data structures for a graphic matroid and a partition matroid in the full version of this paper.

The dynamic maximum weight base data structure. Algorithm 2 makes use of a data structure for maintaining the maximum weight base in the matroid, where each element has a weight and the weights are updated through a sequence of updates that can only decrease the weights. The data structure needs to support the following operation: UpdateBase decreases the weight of an element and it updates the base to a maximum weight base for the updated weights. The data structures that we provide in the full version of the paper for a graphic and a partition matroid support this operation in $O(\text{poly}(\log k))$ amortized time.

We note here that the data structure maintains a maximum weight base of the original matroid $\mathcal{M}$, and not the contracted matroid $\mathcal{M}/S$ obtained after picking a set S of elements. This suffices for us, since the discrete Greedy algorithm that we use will not change the weight of an element after it was added to the solution S. Due to this invariant, we can show that the maximum weight base B of $\mathcal{M}$ that the data structure maintains has the property that $S \subseteq B$ at all times, and $B \setminus S$ is a maximum weight base in $\mathcal{M}/S$. This follows from the observation that, if an element e is in the maximum weight base B and the only changes to the weights are such that the weight of e remains unchanged and the weights of elements other than e are decreased, then e remains in the new maximum weight base.

The discrete Greedy algorithm. The algorithm (Algorithm 2) is based on the random residual Greedy algorithm of [4]. The latter algorithm constructs a solution S over k iterations.

Algorithm 2 LazySamplingGreedy$(f, \mathcal{M}, \epsilon)$.

$M = \Theta(f(\mathrm{OPT}))$, $c = \Theta(1/\epsilon)$, $N = 2\ln(k/\epsilon)/\epsilon$
⟨⟨ `maintain cached (rounded) marginal values` ⟩⟩
For each $e \in V$, let $w(e) = (1-\epsilon)^N M$ if $f(\{e\}) \leq (1-\epsilon)^N M$ and $w(e) = (1-\epsilon)^{j-1} M$ if $f(\{e\}) \in ((1-\epsilon)^j M, (1-\epsilon)^{j-1} M]$
⟨⟨ `maintain a base` B `of maximum` $w(\cdot)$ `value in a data structure that supports the` UpdateBase `operation` ⟩⟩
$B = \mathrm{argmax}_{S \in \mathcal{I}} \sum_{e \in S} w(e)$
⟨⟨ `maintain a partition of` B `into buckets` ⟩⟩
$B^{(j)} = \{e \in B \colon w(e) = (1-\epsilon)^{j-1} M\}$ for each $j \in [N]$
$W = \sum_{e \in B} w(e)$
⟨⟨ `main loop` ⟩⟩
$S = \emptyset$
for $t = 1, 2, \ldots, k$ **do**
    Call RefreshValues
    if $W \leq 4cM$ **then**
        return S
    end if
    Sample an element e uniformly at random from B
    $S \leftarrow S \cup \{e\}$
    Remove e from the buckets of B for refreshing purpose so that $w(e)$ is now fixed
end for

procedure RefreshValues ⟨⟨ `Spot check and update values` ⟩⟩
    for $j = 1$ to N **do**
        $T = 0$
        while $T < 4\log_2 n$ **do**
            if $B^{(j)}$ is empty **then**
                Exit the while loop and continue to iteration $j+1$
            end if
            Sample e uniformly at random from $B^{(j)}$
            Let $v(e) = f(S \cup \{e\}) - f(S)$ be the current marginal value of e
            if $v(e) < (1-\epsilon)^j M$ **then**
                $T = 0$
                UpdateBase$(e, j, v(e))$
            else
                $T \leftarrow T + 1$
            end if
        end while
    end for
end procedure

In each iteration, the algorithm assigns a linear weight to each element that is equal to the marginal gain $f(S \cup \{e\}) - f(S)$ on top of the current solution, and it finds a maximum weight base B in $\mathcal{M}/S$. The algorithm then samples an element of B uniformly at random and adds it to the solution. As discussed in Section 1.2, the key difficulty for obtaining a fast running time is maintaining the maximum weight base. Our algorithm uses the following approach for maintaining an approximate maximum weight base. The algorithm maintains

the marginal value of each element (rounded to the next highest power of $(1-\epsilon)$), and it updates it in a lazy manner; at every point, $w(e)$ denotes the cached (rounded) marginal value of the element, and it may be stale.

The algorithm maintains the base B using the data structure discussed above that supports the UpdateBase operation. Additionally, the elements of $B \setminus S$ are stored into buckets corresponding to geometrically decreasing marginal values. More precisely, there are $N = O(\log(k/\epsilon)/\epsilon)$ buckets $B^{(1)}, B^{(2)}, \ldots, B^{(N)}$. The j-th bucket $B^{(j)}$ contains all of the elements of B with marginal values in the range $((1-\epsilon)^j M, (1-\epsilon)^{j-1} M]$, where M is a value such that $f(\mathrm{OPT}) \leq M \leq O(1) f(\mathrm{OPT})$ (we assume that the algorithm knows such a value M, as it can be obtained in nearly-linear time, see e.g. Lemma 3.2 in [5]). The remaining elements of B that do not appear in any of the N buckets have marginal values at most $(1-\epsilon)^N M$; these elements have negligible total marginal gain, and they can be safely ignored.

In order to achieve a fast running time, after each Greedy step, the algorithm uses sampling to (partially) update the base B, the cached marginal values, and the buckets. This is achieved by the procedure RefreshValues, which works as follows. RefreshValues considers each of the buckets in turn. The algorithm updates the bucket $B^{(j)}$ by spot checking $O(\log n)$ elements sampled uniformly at random from the bucket. For each sampled element e, the algorithm computes its current marginal value and, if it has decreased below the range of its bucket, it moves the element to the correct buckets and call UpdateBase to maintain the invariant that B is a maximum weight base.

When the algorithm finds an element whose bucket has changed, it resets to 0 the count for the number of samples taken from the bucket. Thus the algorithm keeps sampling from the bucket until $\Theta(\log n)$ consecutive sampled elements do not change their bucket. The sampling step ensures that, with high probability, in each bucket at least half of the elements are in the correct bucket. (We remark that, instead of resetting the sample count to 0, it suffices to decrease the count by 1, i.e., the count is the total number of samples whose bucket was correct minus the number of samples whose bucket was incorrect. The algorithm then stops when this tally reaches $\Theta(\log n)$. This leads to an improvement in the running time, but we omit it in favor of a simpler analysis.)

After running RefreshValues, the algorithm samples an element e uniformly at random from $B \setminus S$ and adds it to S. The algorithm then removes e from the buckets; this ensures that the weight of e will remain unchanged for the remainder of the algorithm.

2.1 Analysis of the approximation guarantee

Here we show that Algorithm 1 achieves a $1 - 1/e - \epsilon$ approximation. We first analyze the LazySamplingGreedy algorithm. We start with some convenient definitions. Consider some point in the execution of the LazySamplingGreedy algorithm. Consider a bucket $B^{(j)}$. At this point, each element $e \in B^{(j)}$ is in the correct bucket iff its current marginal value $f(S \cup \{e\}) - f(S)$ lies in the interval $((1-\epsilon)^j M, (1-\epsilon)^{j-1} M]$ (its cached marginal value $w(e)$ lies in that interval, but it might be stale). We say that the bucket $B^{(j)}$ is *good* if at least half of the elements in $B^{(j)}$ are in the correct bucket, and we say that the bucket is *bad* otherwise.

The following lemma shows that, with high probability over the random choices of RefreshValues, each run of RefreshValues ensures that every bucket $B^{(j)}$ with $j \in [N]$ is good.

▶ **Lemma 8.** *Consider an iteration in which* LazySamplingGreedy *calls* RefreshValues. *When* RefreshValues *terminates, the probability that the buckets* $\{B^{(j)} : j \in [N]\}$ *are all good is at least* $1 - 1/n^2$.

Proof. We will show that the probability that a given bucket is bad is at most $5 \log n/n^3$; the claim then follows by the union bound, since there are $N \leq n/(5 \log n)$ buckets. Consider a bucket $B^{(j)}$, where $j \in [N]$, and suppose that the bucket is bad at the end of RefreshValues. We analyze the probability the bucket is bad because the algorithm runs until iteration t, which is the last time the algorithm finds an element in $B^{(j)}$ in the wrong bucket, and for $4 \log n$ iterations after t, it always find elements in the right bucket even though only $1/2$ of $B^{(j)}$ are in the right bucket. Since at most half of the elements of $B^{(j)}$ are in the correct bucket and the samples are independent, this event happens with probability at most $(1/2)^{4 \log_2 n} = 1/n^4$. By the union bound over all choices of $t = 1, 2, \ldots, 5n \log n$, the failure probability for bucket $B^{(j)}$ is at most $5 \log n/n^3$. ◀

Since LazySamplingGreedy performs at most $k \leq n$ iterations, it follows by the union bound that all of the buckets $\{B^{(j)} \colon j \in [N]\}$ are all good throughout the algorithm with probability at least $1 - 1/n$. For the remainder of the analysis, we condition on this event. Additionally, we fix an event specifying the random choices made by RefreshValues and we implicitly condition all probabilities and expectations on this event.

Let us now show that B is a suitable approximation for the maximum weight base in $\mathcal{M}/S$ with weights given by the current marginal values $f(S \cup \{e\}) - f(S)$.

▶ **Lemma 9.** *Suppose that every bucket of B is good throughout the algorithm. Let $v(e) = f(S \cup \{e\}) - f(S)$ denote the current marginal values. We have*
(1) *$w(S') \geq v(S')$ for every $S' \subseteq V$;*
(2) *$w(B) \geq w(S')$ for every $S' \subseteq V$;*
(3) *$v(B) \geq \frac{1-\epsilon}{2} \cdot w(B) - \frac{\epsilon^2}{k} \cdot M$.*

Proof. The first property follows from the fact that, by submodularity, the weights $w(\cdot)$ are upper bounds on the marginal values.

The second property follows from the fact that the algorithm maintains the invariant that B is the maximum weight base in $\mathcal{M}/S$ with respect to the weights $w(\cdot)$.

Let us now show the third property. Consider the following partition of B into sets B_1, B_2, and B_3, where: B_1 is the set of all elements $e \in B$ such that e is in one of the buckets $\{B^{(j)} \colon j \in [N]\}$ and moreover e is in the correct bucket (note that $(1-\epsilon)w(e) \leq v(e) \leq w(e)$ for every $e \in B_1$); B_2 is the set of all elements $e \in B$ such that e is in one of the buckets $\{B^{(j)} \colon j \in [N]\}$ but e is not in the correct bucket (i.e., $(1-\epsilon)^N M < w(e)$ but $v(e) < (1-\epsilon)w(e)$); B_3 is the set of all elements $e \in B$ such that $w(e) < (1-\epsilon)^N M \leq (\epsilon/k)^2 M$.

Since $|B_3| \leq k$, we have $w(B_3) \leq |B_3| \cdot \left(\frac{\epsilon}{k}\right)^2 M \leq \frac{\epsilon^2}{k} M$.

Since all of the buckets are good, it follows that the total $w(\cdot)$ weight of the elements that are in the correct bucket is at least $\frac{1}{2} \cdot w(B \setminus B_3)$. Indeed, we have

$$w(B_1) = \sum_{j=1}^{N} w(B_1 \cap B^{(j)}) = \sum_{j=1}^{N} (1-\epsilon)^{j-1} M |B_1 \cap B^{(j)}| \geq \sum_{j=1}^{N} (1-\epsilon)^{j-1} M \frac{|B^{(j)}|}{2} = \frac{w(B \setminus B_3)}{2}.$$

Finally, since $v(e) \geq (1-\epsilon)w(e)$ for every $e \in B_1$, we have

$$v(B) \geq v(B_1) \geq (1-\epsilon)w(B_1) \geq \frac{1-\epsilon}{2} w(B \setminus B_3) \geq \frac{1-\epsilon}{2} w(B) - \frac{\epsilon^2}{k} M. \quad ◀$$

Now we turn to the analysis of the main for loop of LazySamplingGreedy (lines 11–19). Let Z be a random variable equal to the number of iterations where the algorithm executes line 17. We define sets $\{S_t \colon t \in \{0, 1, \ldots, k\}\}$ and $\{\mathrm{OPT}_t \colon t \in \{0, 1, \ldots, k\}\}$ as follows. Let $S_0 = \emptyset$ and $\mathrm{OPT}_0 = \mathrm{OPT}$. Consider an iteration $t \leq Z$ and suppose that

S_{t-1} and OPT_{t-1} have already been defined and they satisfy $S_{t-1} \cup \mathrm{OPT}_{t-1} \in \mathcal{I}$ and $|S_{t-1}| + |\mathrm{OPT}_{t-1}| = k$. Consider a bijection $\pi : B \to \mathrm{OPT}_{t-1}$ so that $\mathrm{OPT}_{t-1} \setminus \{\pi(e)\} \cup \{e\}$ is a base of $\mathcal{M}/S_{t-1}$ for all $e \in B$ (such a bijection always exists, see e.g., Corollary 39.12a in [22]). Let e_t be the element sampled on line 17 and $o_t = \pi(e_t)$. We define $S_t = S_{t-1} \cup \{e_t\}$ and $\mathrm{OPT}_t = \mathrm{OPT}_{t-1} \setminus \{o_t\}$. Note that $S_t \cup \mathrm{OPT}_t \in \mathcal{I}$. For each $t > Z$, we define $S_t = S_Z$ and $\mathrm{OPT}_t = \mathrm{OPT}_Z$.

In each iteration t, the gain in the Greedy solution value is $f(S_t) - f(S_{t-1})$, and the loss in the optimal solution value is $f(\mathrm{OPT}_{t-1}) - f(\mathrm{OPT}_t)$ (when we add an element to S_{t-1}, we remove an element from OPT_{t-1} so that $S_t \cup \mathrm{OPT}_t$ remains a feasible solution). The following lemma relates the two values in expectation.

▶ **Lemma 10.** *For every $t \in [k]$, if all of the buckets $B^{(j)}$ are good, we have* $\mathbb{E}[f(S_t) - f(S_{t-1})] \geq c \cdot \mathbb{E}[f(\mathrm{OPT}_{t-1}) - f(\mathrm{OPT}_t)]$.

Proof. Consider $t \in [k]$. Recall that Z is the number of iterations where the algorithm executes line 17. If $t > Z$, the inequality is trivially satisfied, since both expectations are equal to 0. Therefore we may assume that $t \leq Z$ and thus $S_t = S_{t-1} \cup \{e_t\}$ and $\mathrm{OPT}_t = \mathrm{OPT}_{t-1} \setminus \{o_t\}$.

Let us now fix an event R_{t-1} specifying the random choices for the first $t-1$ iterations, i.e., the random elements $e_1, \ldots, e_{t-1}$ and $o_1, \ldots, o_{t-1}$. In the following, all the probabilities and expectations are implicitly conditioned on R_{t-1}. Note that, once R_{t-1} is fixed, S_{t-1} and OPT_{t-1} are deterministic.

Let us first lower bound $\mathbb{E}[f(S_{t-1} \cup \{e_t\}) - f(S_{t-1})]$. Let w_t, B_t, and W_t denote w, B, and W right after executing RefreshValues in iteration t. Note that, since R_{t-1} and the random choices of RefreshValues are fixed, w_t, B_t, and W_t are deterministic.

Recall that all of the buckets of B_t are good, i.e., at least half of the elements of $B_t^{(j)}$ are in the correct bucket, for every $j \in [N]$. Let B'_t be the subset of B_t consisting of all of the elements that are in the correct bucket, and let B''_t be the subset of B_t consisting of all of the elements that are not in any bucket.

For every $e \in B'_t$, we have $f(S_{t-1} \cup \{e\}) - f(S_{t-1}) \geq (1-\epsilon) w_t(e)$. For every $e \in B''_t$, we have $f(S_{t-1} \cup \{e\}) - f(S_{t-1}) \leq w_t(e) = (1-\epsilon)^N M \leq (\epsilon/k)^2 M$, and therefore $w_t(B''_t) \leq \frac{\epsilon^2}{k} M$. Since all of the buckets are good and $W_t > 4cM$ (the algorithm did not terminate on line 14), we have

$$\begin{aligned}\sum_{e \in B'_t} w_t(e) &= \sum_{j=1}^{N} w_t(B'_t \cap B_t^{(j)}) = \sum_{j=1}^{N} (1-\epsilon)^{j-1} M |B'_t \cap B_t^{(j)}| \geq \sum_{j=1}^{N} (1-\epsilon)^{j-1} M \frac{|B_t^{(j)}|}{2} \\ &= \frac{w_t(B_t \setminus B''_t)}{2} \geq \frac{W_t}{2} - \frac{\epsilon^2}{2k} M \geq \left(2c - \frac{\epsilon^2}{2k}\right) M \geq \left(2c - \frac{\epsilon^2}{2k}\right) f(\mathrm{OPT}).\end{aligned}$$

By combining these observations, we obtain

$$\begin{aligned}&\mathbb{E}[f(S_{t-1} \cup \{e_t\}) - f(S_{t-1})] \geq \mathbb{E}[f(S_{t-1} \cup \{e_t\}) - f(S_{t-1}) | e_t \in B'_t] \Pr[e_t \in B'_t] \\ &= \mathbb{E}[f(S_{t-1} \cup \{e_t\}) - f(S_{t-1}) | e_t \in B'_t] \cdot \frac{|B'_t|}{|B_t|} \geq (1-\epsilon) \mathbb{E}[w_t(e_t) | e_t \in B'_t] \cdot \frac{|B'_t|}{|B_t|} \\ &= (1-\epsilon) w_t(B'_t) \cdot \frac{1}{|B'_t|} \cdot \frac{|B'_t|}{|B_t|} \geq \frac{(1-\epsilon)\left(2c - \frac{\epsilon^2}{2k}\right)}{|B_t|} f(\mathrm{OPT}) \geq \frac{c}{|B_t|} f(\mathrm{OPT})\end{aligned}$$

Let us now upper bound $\mathbb{E}[f(\mathrm{OPT}_{t-1}) - f(\mathrm{OPT}_t)]$. Recall that e_t is chosen randomly from

B and thus, o_t is chosen uniformly at random from OPT_{t-1} (since π is a bijection). Hence

$$\mathbb{E}[f(\mathrm{OPT}_{t-1}) - f(\mathrm{OPT}_{t-1} \setminus \{o_t\})] = \sum_{o \in \mathrm{OPT}_{t-1}} (f(\mathrm{OPT}_{t-1}) - f(\mathrm{OPT}_{t-1} \setminus \{o\})) \cdot \frac{1}{|\mathrm{OPT}_{t-1}|}$$

By submodularity, we have

$$f(\mathrm{OPT}_{t-1}) \geq \sum_{j=1}^{m} (f(\mathrm{OPT}_{t-1}) - f(\mathrm{OPT}_{t-1} \setminus \{o_j\}).$$

Therefore

$$\mathbb{E}[f(\mathrm{OPT}_{t-1}) - f(\mathrm{OPT}_{t-1} \setminus \{o_t\})] \leq \frac{f(\mathrm{OPT}_{t-1})}{|\mathrm{OPT}_{t-1}|} \leq \frac{f(\mathrm{OPT})}{|\mathrm{OPT}_{t-1}|}.$$

To recap, we have shown that:

$$\mathbb{E}[f(S_{t-1} \cup \{e_t\}) - f(S_{t-1})] \geq \frac{c \cdot f(\mathrm{OPT})}{|B_t|},$$
$$\mathbb{E}[f(\mathrm{OPT}_{t-1}) - f(\mathrm{OPT}_{t-1} \setminus \{o_t\})] \leq \frac{f(\mathrm{OPT})}{|\mathrm{OPT}_{t-1}|}.$$

Since $|B_t| = |\mathrm{OPT}_{t-1}|$, we have

$$\mathbb{E}[f(S_{t-1} \cup \{e_t\}) - f(S_{t-1})] \geq c \cdot \mathbb{E}[f(\mathrm{OPT}_{t-1}) - f(\mathrm{OPT}_{t-1} \setminus \{o_t\})].$$

Since the above inequality holds conditioned on every given event R_{t-1}, it holds unconditionally, and the lemma follows. ◀

The following lemma follows from Lemmas 9 and 10.

▶ **Lemma 11.** *If all of the buckets $B^{(j)}$ are good, the* LazySamplingGreedy *algorithm (Algorithm 2) returns a set $S \in \mathcal{I}$ with the following properties.*

(1) $\max_{S' \colon S' \cup S \in \mathcal{I}} \sum_{e \in S'} f_S(e) \leq 4cM = O(1/\epsilon) f(\mathrm{OPT})$.

(2) *There is a random subset $\mathrm{OPT}' \subseteq \mathrm{OPT}$ depending on S with the following properties: $S \cup \mathrm{OPT}' \in \mathcal{I}$ and $\mathbb{E}[f(\mathrm{OPT}')] \geq f(\mathrm{OPT}) - \frac{1}{c} \cdot \mathbb{E}[f(S)] \geq \left(1 - \frac{1}{c}\right) f(\mathrm{OPT})$.*

By combining Lemmas 7 and 11, we obtain:

▶ **Lemma 12.** *The* ContinuousMatroid *algorithm (Algorithm 1) returns a solution $\mathbf{1}_S \vee x \in P(\mathcal{M})$ such that $F(\mathbf{1}_S \vee x) \geq (1 - 1/e - O(\epsilon)) f(\mathrm{OPT})$ with constant probability.*

Proof. Note that, in order to apply Lemma 7, we need the following condition to hold: $\max_{S' \colon S' \cup S \in \mathcal{I}} \sum_{e \in S'} f_S(e) \leq c' f_S(\mathrm{OPT}'')$, where $\mathrm{OPT}'' \in \operatorname{argmax}_{S' \colon S' \cup S \in \mathcal{I}} f_S(S')$.

Using Lemma 11, we can show that the above condition holds with constant probability as follows. Let OPT' be the set guaranteed by Lemma 11. We have $f_S(\mathrm{OPT}') \leq f_S(\mathrm{OPT}'')$ and $f(S \cup \mathrm{OPT}') \geq f(\mathrm{OPT}')$. Therefore $f_S(\mathrm{OPT}'') \geq f_S(\mathrm{OPT}') \geq f(\mathrm{OPT}') - f(S)$.

By Lemma 11, we have $\mathbb{E}[f(\mathrm{OPT}) - f(\mathrm{OPT}')] \leq f(\mathrm{OPT})/c$. Therefore, by the Markov inequality, with probability at least $2/3$, we have $f(\mathrm{OPT}) - f(\mathrm{OPT}') \leq 3f(\mathrm{OPT})/c$. Consider two cases. First, if $f(S) \geq (1 - 1/e) f(\mathrm{OPT})$ then the algorithm can simply return S. Second, if $f(S) < (1 - 1/e) f(\mathrm{OPT})$ then $f_S(\mathrm{OPT}'') \geq f(\mathrm{OPT}') - f(S) \geq (1/e - 3/c) f(\mathrm{OPT})$. Therefore, $\max_{S' \colon S' \cup S \in \mathcal{I}} \sum_{e \in S'} f_S(e) \leq O(c f_S(\mathrm{OPT}'')) \leq c' f_S(\mathrm{OPT}'')$. Thus the conditions of Lemma 7 are satisfied and thus the continuous Greedy algorithm returns a solution $x \in P(\mathcal{M}/S)$ such that

$$F(\mathbf{1}_S \vee x) - f(S) \geq \left(1 - \frac{1}{e} - \epsilon\right) (f(\mathrm{OPT}') - f(S))$$
$$\geq \left(1 - \frac{1}{e} - \epsilon\right) \left(1 - \frac{3}{c}\right) f(\mathrm{OPT}) - f(S) \geq \left(1 - \frac{1}{e} - 2\epsilon\right) f(\mathrm{OPT}) - f(S).$$ ◀

References

1. Alexander Ageev and Maxim Sviridenko. Pipage Rounding: A New Method of Constructing Algorithms with Proven Performance Guarantee. *Journal of Combinatorial Optimization*, 8(3):307–328, 2004.
2. Yossi Azar and Iftah Gamzu. Efficient Submodular Function Maximization under Linear Packing Constraints. In *International Colloquium on Automata, Languages and Programming (ICALP)*, 2012.
3. Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2014.
4. Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular Maximization with Cardinality Constraints. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2014.
5. Niv Buchbinder, Moran Feldman, and Roy Schwartz. Comparing Apples and Oranges: Query Trade-off in Submodular Maximization. *Math. Oper. Res.*, 42(2):308–329, 2017.
6. Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a Submodular Set Function Subject to a Matroid Constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.
7. Chandra Chekuri, T. S. Jayram, and Jan Vondrák. On Multiplicative Weight Updates for Concave and Submodular Function Maximization. In *Conference on Innovations in Theoretical Computer Science (ITCS)*, 2015. doi:10.1145/2688073.2688086.
8. Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Dependent Randomized Rounding via Exchange Properties of Combinatorial Structures. In *IEEE Foundations of Computer Science (FOCS)*, pages 575–584. IEEE Computer Society, 2010.
9. Shaddin Dughmi, Tim Roughgarden, and Mukund Sundararajan. Revenue Submodularity. *Theory of Computing*, 8(1):95–119, 2012.
10. Uriel Feige. A threshold of ln n for approximating set cover. *jacm*, 45:634–652, 1998.
11. Yuval Filmus and Justin Ward. Monotone Submodular Maximization over a Matroid via Non-Oblivious Local Search. *SIAM Journal on Computing*, 43(2):514–542, 2014.
12. M L Fisher, G L Nemhauser, and L A Wolsey. An analysis of approximations for maximizing submodular set functions—II. *Mathematical Programming Studies*, 8:73–87, 1978.
13. Bernard A Galler and Michael J Fisher. An improved equivalence algorithm. *Communications of the ACM*, 7(5):301–303, 1964.
14. Ryan Gomes and Andreas Krause. Budgeted Nonparametric Learning from Data Streams. In *International Conference on Machine Learning (ICML)*, pages 391–398, 2010.
15. Jacob Holm, Kristian De Lichtenberg, and Mikkel Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *Journal of the ACM (JACM)*, 48(4):723–760, 2001.
16. David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 137–146, 2003.
17. Andreas Krause, Ajit Paul Singh, and Carlos Guestrin. Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. *Journal of Machine Learning Research*, 9:235–284, 2008.
18. Hui Lin and Jeff A. Bilmes. Multi-document Summarization via Budgeted Maximization of Submodular Functions. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, pages 912–920, 2010.
19. Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier Than Lazy Greedy. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2015.
20. G L Nemhauser and L A Wolsey. Best Algorithms for Approximating the Maximum of a Submodular Set Function. *Mathematics of Operations Research*, 3(3):177–188, 1978.

21 G L Nemhauser, L A Wolsey, and M L Fisher. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1):265–294, 1978.

22 Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, 2003.

23 Robert Endre Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM (JACM)*, 22(2):215–225, 1975.

24 Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *ACM Symposium on Theory of Computing (STOC)*, 2008.

On the Complexity of String Matching for Graphs

Massimo Equi
Department of Computer Science, University of Helsinki, Finland
massimo.equi@helsinki.fi

Roberto Grossi
Dipartimento di Informatica, Università di Pisa, Italy
grossi@di.unipi.it

Veli Mäkinen
Department of Computer Science, University of Helsinki, Finland
veli.makinen@helsinki.fi

Alexandru I. Tomescu
Department of Computer Science, University of Helsinki, Finland
alexandru.tomescu@helsinki.fi

Abstract

Exact string matching in labeled graphs is the problem of searching paths of a graph $G = (V, E)$ such that the concatenation of their node labels is equal to the given pattern string $P[1..m]$. This basic problem can be found at the heart of more complex operations on variation graphs in computational biology, of query operations in graph databases, and of analysis operations in heterogeneous networks.

We prove a conditional lower bound stating that, for any constant $\epsilon > 0$, an $O(|E|^{1-\epsilon}\, m)$-time, or an $O(|E|\, m^{1-\epsilon})$-time algorithm for exact string matching in graphs, with node labels and patterns drawn from a binary alphabet, cannot be achieved unless the Strong Exponential Time Hypothesis (SETH) is false. This holds even if restricted to undirected graphs with maximum node degree two, i.e. to *zig-zag matching in bidirectional strings*, or to *deterministic* directed acyclic graphs whose nodes have maximum sum of indegree and outdegree three. These restricted cases make the lower bound stricter than what can be directly derived from related bounds on regular expression matching (Backurs and Indyk, FOCS'16). In fact, our bounds are tight in the sense that lowering the degree or the alphabet size yields linear-time solvable problems.

An interesting corollary is that exact and approximate matching are equally hard (quadratic time) in graphs under SETH. In comparison, the same problems restricted to strings have linear-time vs quadratic-time solutions, respectively (approximate pattern matching having also a matching SETH lower bound (Backurs and Indyk, STOC'15)).

2012 ACM Subject Classification Theory of computation → Pattern matching

Keywords and phrases exact pattern matching, graph query, graph search, labeled graphs, string matching, string search, strong exponential time hypothesis, heterogeneous networks, variation graphs

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.55

Category Track A: Algorithms, Complexity and Games

Related Version This paper is based on our two technical reports [14, 15] available at `https://arxiv.org/abs/1901.05264` and `https://arxiv.org/abs/1902.03560`.

Funding This work has been partially supported by Academy of Finland (grant 309048).

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 55; pp. 55:1–55:15

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

1 Introduction

String matching is the classical problem of finding the occurrences of a pattern string as a substring of a text string [22]. As most of today's data is linked, it is natural to investigate string matching in labeled graphs. Indeed, large-scale labeled graphs are becoming ubiquitous in several areas, such as graph databases [6, 16, 30, 27], graph mining [19, 12], and computational biology [11]. Applications require sophisticated operations on these graphs, and often rely on primitives that locate paths whose nodes have labels or types matching a pattern given at query time. The most basic pattern is a string and, as we will see, this already poses a challenge when performing string matching in graphs.

Problem Definition

Given an alphabet Σ of symbols, consider a labeled graph $G = (V, E, L)$, where (V, E) represents a directed or undirected graph and $L : V \to \Sigma$ is a function that defines which symbol from Σ is assigned to each node as label.[1] A node labeled with $\sigma \in \Sigma$ is called a *σ-node*, and an edge whose endpoints are labeled σ_1 and σ_2, respectively, is called a *$\sigma_1\sigma_2$-edge*. If G is a directed graph, we say that G is *deterministic* if, for any two out-neighbors of the same node, their labels are different. In the following, we introduce the acronym *3-DDAG* to indicate a deterministic directed acyclic graph (DAG) such that its nodes are labeled with a binary alphabet and the sum of indegree and outdegree of each node is at most 3.

Given a pattern string $P[1..m]$ over Σ, we say that P has a *match* in G if there is a path $u_1, \ldots, u_k$ such that $P = L(u_1) \cdots L(u_k)$ (we also say that P *occurs* in G, and that $u_1, \ldots, u_k$ is an *occurrence* of P).

▶ **Problem 1** (String Matching in Labeled Graphs (SMLG)).
INPUT*: A labeled graph $G = (V, E, L)$ and a pattern string P, both over an alphabet Σ.*
OUTPUT*:* True *if and only if there is at least one occurrence of P in G.*

Results

We give conditional bounds for the SMLG problem using the Orthogonal Vectors (OV) hypothesis [34]. The latter states that for any constant $\epsilon > 0$, no algorithm can solve in $O(n^{2-\epsilon}\text{poly}(d))$ time the OV problem: given two sets $X, Y \subseteq \{0,1\}^d$ such that $|X| = |Y| = n$ and $d = \omega(\log n)$, decide whether there exist $x \in X$ and $y \in Y$ such that x and y are orthogonal, namely, $x \cdot y = 0$. We observe that it is common practice to use the Strong Exponential Time Hypothesis (SETH) [20] but, since SETH implies the OV hypotesis [34], it suffices to use the OV hypothesis in the bounds, as they hold also for SETH.

First, we consider the SMLG problem on directed graphs. Their weakest form is a 3-DDAG, for which we prove in Section 2 that subquadratic time for exact string matching cannot be achieved unless the OV hypothesis is false.

▶ **Theorem 1.** *For any constant $\epsilon > 0$, the String Matching in Labeled Graphs (SMLG) problem for a binary alphabet and a labeled* deterministic *directed acyclic graph (DAG) cannot be solved in either $O(|E|^{1-\epsilon}\, m)$ or $O(|E|\, m^{1-\epsilon})$ time unless the OV hypothesis fails. This holds even if it is restricted to graphs in which the sum of outdegree and indegree of any node is at most three (i.e, 3-DDAGs).*

[1] Note that we can also define the node labels as nonempty strings, but it suffices to use single symbols to show that string matching in graphs is challenging.

Table 1 Legend: V = set of nodes, E = set of edges, occ = number of matches for the pattern in the graph, m = pattern length, N = total length of text in all nodes, *(1)* errors only in the pattern, *(2)* errors in the graph, *(3)* matches span only one edge. The two rows highlighted in gray report the best known bounds for exact and approximate string matching, respectively.

State of the art for SMLG				
Year	**Authors**	**Graph**	**Exact/ Approximate**	**Time**
1992	Manber, Wu [24]	DAG	approximate$^{(1)}$	$O(m\|E\| + occ \lg\lg m)$
1993	Akutsu [2]	tree	exact	$O(N)$
1995	Park, Kim [26]	DAG	exact$^{(3)}$	$O(N + m\|E\|)$
1997	Amir et al. [5]	general	exact	$O(N + m\|E\|)$
1997	Amir et al. [5]	general	approximate$^{(2)}$	NP-Hard
1997	Amir et al. [5]	general	approximate$^{(1)}$	$O(Nm \lg N + m\|E\|)$
1998	Navarro [25]	general	approximate$^{(1)}$	$O(Nm + m\|E\|)$
2017	Rautiainen, Marschall [29]	general	approximate$^{(1)}$	$O(N + m\|E\|)$
2019	Jain et al. [21]	general binary alphabet	approximate$^{(2)}$	NP-Hard

Next, we consider the SMLG problem on undirected graphs and introduce the *zig-zag* pattern matching problem in strings, which models searching a string P along a path of an undirected graph. While an exact occurrence of P in a text string is found by scanning the text forward for increasing positions in P, a *zig-zag* occurrence of P can be found by partially scanning forward and backward adjacent text positions, as many times as needed (e.g. for an edge $\{u, v\}$ with $L(u) = \mathtt{a}$ and $L(v) = \mathtt{b}$, all patterns of the form $\mathtt{a}, \mathtt{ab}, \mathtt{aba}, \mathtt{abab}, \ldots$ occur starting from u). We prove in Section 3 the following result.

▶ **Theorem 2.** *The conditional lower bound stated in Theorem 1 holds even if it is restricted to undirected graphs whose nodes have degree at most 2, where the pattern and the node labels are drawn from a binary alphabet.*

Our results can cover arbitrary graphs in this way. Interpreting the graphs from Theorem 2 as directed, we observe that they have nodes with both indegree and outdegree 2. Looking at Theorem 1, we observe that it involves directed graphs with both nodes of indegree at most 1 and outdegree 2, and nodes with outdegree at most 1 and indegree 2. Thus, the only uncovered case is that of directed graphs with only nodes of indegree at most 1, or directed graphs with only nodes of outdegree at most 1. For such graphs, we observe that their edges can be decomposed into forests of directed trees (arborescences), whose roots may be connected in a directed cycle (at most one cycle per forest). In the extended version of this work we will show that the Knuth-Morris-Pratt algorithm [22] can be easily extended to solve exact string matching for these special directed graphs in linear time, thus completing the full picture.

History and Implications

The idea of extending the problem of string matching to graphs, as given in SMLG, is not new. If the nodes $u_1, \ldots, u_k$ are required to be distinct (i.e., to be a *simple* path), this problem is NP-hard as it solves the well-known Hamiltonian Path problem, so this

requirement is removed for this reason. The SMLG problem was studied over 25 years ago as a search problem for hypertext by Manber and Wu [24]. The history of key contributions is given in Table 1, where a common feature of the reported bounds is the appearance of the quadratic term $m\,|E|$ (except for some special cases). The quadratic cost of the approximate matching in graphs is asymptotically optimal under the Strong Exponential Time Hypothesis (SETH) [20] as (i) it solves the approximate string matching as a special case, since a graph consisting of just one directed path of $|E|+1$ nodes and $|E|$ edges is a text string of length $n = |E|+1$, and (ii) it has been recently proved that the edit distance of two strings of length n cannot be computed in $O(n^{2-\epsilon})$ time, for any constant $\epsilon > 0$, unless SETH is false [7]. This conditional lower bound explains why the $O(m|E|)$ barrier has been difficult to cross in the approximate case. Specifically, Amir et al. [4, 5], gave a quadratic-time solution for exact string matching in $O(N + m \cdot |E|)$ time, where $N = \sum_{u \in V} |L(u)|$. Rautiainen and Marschall [29] and Jain et al. [21] recently gave the best bound for errors in pattern only, $O(N + m \cdot |E|)$ time, same as the exact string matching. The two best results for exact and approximate pattern matching, both taking quadratic time in the worst case, are highlighted in Table 1. As allowing errors in the graph makes the problem NP-hard [5], we consider here errors in the pattern only.

In this scenario and the application domains mentioned at the beginning, our results have a number of implications discussed below.

- While we can explain the complexity of *approximate* string matching in graphs, not much is known on the complexity of *exact* string matching in graphs. The classical exact string matching can be solved in linear time [22], so one could expect the corresponding problem on graphs to be easier than approximate string matching. A lower bound (i.e., NP-hard, as mentioned above) exists only in the case when the pattern is restricted to match only simple paths in the graph. Extensions of this type of matching for special graph classes have been studied in [23]. Here we study the general case, where paths can pass through nodes multiple times. Somewhat surprisingly Theorems 1 and 2 imply that *exact and approximate pattern matching are equally hard in graphs*, even if they are 3-DDAGs.
- Our results imply that the algorithm for directed graphs by Amir et al. [4, 5] is essentially the best we can hope for asymptotic bounds unless the OV hyptothesis is false. This also applies to the case of undirected graphs by the simple transformation so that each edge $\{u, v\}$ is transformed into a pair of arcs (u, v) and (v, u). Note that we need also Theorem 2 to explicitly state that this is the best possible also for undirected graphs of maximum degree 2. To complete the picture, we show how to get linear time for the above special case of directed graphs where each node has indegree at most 1, or directed graphs whose nodes have outdegree at most 1.
- Our results also explain why it has been difficult to find indexing schemes for fast exact string matching in graphs, with other than best-case or average-case guarantees [31, 17], except for limited search scenarios [32]. They complement recent findings about *Wheeler graphs* [17, 18, 3]. Wheeler graphs are a class of graphs admitting an index structure that can be constructed in linear time and that supports linear-time exact pattern matching. Gibney and Thankachan [18] claim that it is NP-complete to recognize whether a (non-deterministic) DAG is a Wheeler graph. Alanko et al. [3] claim a linear-time algorithm for recognizing whether a deterministic DAG is a Wheeler graph. Theorem 1 shows that converting an arbitrary deterministic DAG into an equivalent Wheeler graph should take at least quadratic time unless the OV hypothesis is false. In particular, the 3-DDAG obtained in the reduction from OV in the proof of Theorem 1 is not a Wheeler graph.

- We observe that, for any given pattern P, the 3-DDAG obtained by our reduction admits at most one occurrence per node, as it is deterministic and acyclic. When both P and a node u are given, it takes linear time to search P starting from u. Interestingly, if P alone is given, we observe that our results imply that an algorithm reporting whether there exists a node u from which an occurrence of P starts cannot take subquadratic time unless the OV hypothesis is false. Indeed, this hypothetical algorithm would be able to solve the SMLG problem also on that 3-DDAG.
- In the extended version of this work we will describe a simple transformation so that we can see our 3-DDAG and the pattern P as two DFAs, so that our SMLG problem reduces to the emptiness intersection for the string sets recognized by these two DFAs. In this way we give a quadratic conditional lower bound for the latter problem using OV. This adds to the results known in the literature for DFAs (tree automata) under SETH [33] and 3SUM [13]. It is worth noting that our SMLG problem on tree automata takes instead linear time, as will be discussed in the extended version of this work, and this seems to suggest that SMLG could be easier than emptiness intersection for two DFAs under SETH, even though both problems have conditional quadratic lower bounds.

Our reductions share some similarities with those for string problems [7, 10, 1, 8, 9]. The closest connection is with a conditional hardness of several forms of regular expression matching [8]. Especially, one could start with a *non-deterministic finite automaton (NFA)* derived from the regular expression matching of type $|\cdot|$, and add universal "jolly" gadgets (see our reduction) to come up with an OV lower bound for exact pattern matching in directed non-deterministic graphs. (For the interested reader, this is what we have done in an early version of this work [14].) However, to cover the deterministic and bounded degree cases, we build our reduction using a different strategy. This strategy yields a graph of small degree and enables local merging of non-deterministic subgraphs into deterministic counterparts. This locality feature of our reduction is crucial, since converting an NFA into a *deterministic finite automaton (DFA)* can take exponential time [28]. Finally, while this reduction works also for undirected graphs of small degree, it does not cover undirected graphs of degree two. For this case (zig-zag matching in a bidirectional string), we need a more intricate reduction as the underlying graph has less structure.

2 Deterministic Directed Acyclic Graphs

In this section we reduce the OV problem to the SMLG problem for the restricted case of 3-DDAGs. Since any SMLG algorithm for arbitrary directed graphs can be applied also to 3-DDAGs in the same complexity, we will show that a subquadratic-time SMLG algorithm would make the OV hypothesis false. In this scenario, 3-DDAGs are the most restricted case, as otherwise the SMLG problem can be solved in linear time.

Given an OV instance with sets $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_n\}$ of d-dimensional binary vectors, we show how to build a pattern P and a 3-DDAG G such that P will have a match in G if and only if there exists a vector in X orthogonal to one in Y. We first describe how to build P and how to obtain a directed graph whose nodes are labeled with a constant-sized alphabet. Then we discuss how to turn such graph into the 3-DDAG G.

2.1 Pattern

Pattern P is over the alphabet $\Sigma = \{\mathtt{b}, \mathtt{e}, \mathtt{0}, \mathtt{1}\}$, has length $|P| = O(nd)$, and can be built in $O(nd)$ time from the first set of vectors $X = \{x_1, \ldots, x_n\}$. Namely, we define $P = \mathtt{bb}P_{x_1}\mathtt{e}\,\mathtt{b}P_{x_2}\mathtt{e}\ldots\mathtt{b}P_{x_n}\mathtt{ee}$ where P_{x_i} is a string of length d that is associated with $x_i \in X$,

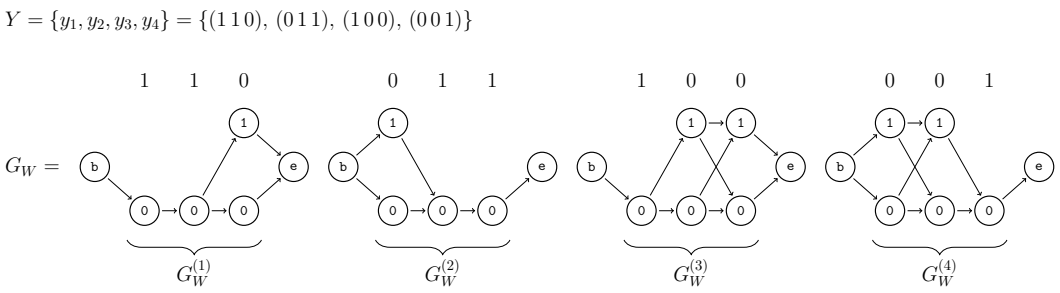


Figure 1 Gadget G_W.

for $1 \leq i \leq n$. The h-th symbol of P_{x_i} is either `0` or `1`, for each $h \in \{1, \ldots, d\}$, such that $P_{x_i}[h] = \texttt{1}$ if and only if $x_i[h] = 1$.[2] We thus view the vectors in X as subpatterns P_{x_i}s which are concatenated by placing separator characters `eb`. Note that P starts with `bb` and ends with `ee`: such strings are found nowhere else in P, marking thus its beginning and its end.

2.2 Directed Graph

The gadget implementing the main logic of the reduction is a directed graph $G_W = (V_W, E_W, L_W)$, illustrated in Figure 1. Starting from the second set of vectors Y, set V_W can be seen as n disjoint *groups* of nodes $V_W^{(1)}, V_W^{(2)}, \ldots, V_W^{(n)}$ (plus some extra nodes), where the nodes in $V_W^{(j)}$ are uniquely associated with vector $y_j \in Y$, for $1 \leq j \leq n$. The corresponding induced subgraph $G_W^{(j)} = (V_W^{(j)}, E_W^{(j)})$ will contain an occurrence of a subpattern P_{x_i} if and only if $x_i \cdot y_j = 0$. We give more details below.

The nodes in $V_W^{(j)}$ are defined as follows. For $1 \leq h \leq d$, we consider entry $y_j[h]$ of vector $y_j \in Y$. If $y_j[h] = 1$, we place just a `0`-node w_{jh}^0 to indicate that we only accept $P_{x_i}[h] = \texttt{0}$ for this h coordinate. Instead, if $y_j[h] = 0$, we place both a `0`-node w_{jh}^0 and a `1`-node w_{jh}^1 to indicate that the value of $P_{x_i}[h]$ does not matter. The nodes in $V_W^{(j)}$ are preceded by a special begin `b`-node $b_W^{(j)}$ and succeeded by a special end `e`-node $e_W^{(j)}$. The overall nodes are thus $V_W = \bigcup_{1 \leq j \leq n} (V_W^{(j)} \cup \{b_W^{(j)}, e_W^{(j)}\})$, and it holds that $|V_W| = O(nd)$.

As for the edges in $E_W^{(j)}$, they properly connect the nodes inside each group $V_W^{(j)}$. Specifically, node $b_W^{(j)}$ is connected to w_{j1}^0 and, if it exists, to w_{j1}^1. Also, we place edges connecting both nodes w_{jd}^0 and w_{jd}^1 (if this exists) to node $e_W^{(j)}$. Moreover, there is an edge for every pair of nodes that are consecutive in terms of h coordinate, for $1 \leq h < d$ (e.g., w_{jh}^1 is connected to $w_{j\,h+1}^0$). The overall edges are thus $E_W = \bigcup_{1 \leq j \leq n} E_W^{(j)}$, where $|E_W| = O(nd)$.

In this way we define the directed graph $G_W = (V_W, E_W, L_W)$, which can be built in $O(nd)$ time from set Y and consists of n connected components $G_W^{(j)}$, one for each vector $y_j \in Y$.

We observe that pattern occurrences in G_W have some useful combinatorial properties. The lemma below is an immediate observation, which follows from the fact that each $G_W^{(j)}$ is acyclic and not connected to any other $G_W^{(j')}$.

▶ **Lemma 3.** *If subpattern* $\texttt{b}P_{x_i}\texttt{e}$ *has a match in* G_W *then the nodes matching* P_{x_i} *share the same* j *coordinate and have distinct and consecutive* h *coordinates.*

Lemma 4 (whose proof will be provided in the extended version of this work) relates the occurrence of a subpattern to the OV problem.

[2] Note that `1` is a symbol of Σ while 1 is the truth value in x_i.

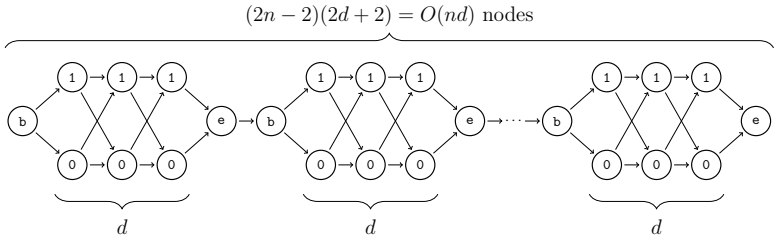


Figure 2 Gadget G_U.

▶ **Lemma 4.** *Subpattern* $\texttt{b}P_{x_i}\texttt{e}$ *has a match in* G_W *if and only if there exist* $y_j \in Y$ *such that* $x_i \cdot y_j = 0$.

In the following we will also use gadget $G_U = (V_U, E_U, L_U)$, the degenerate case of G_W with $2n-2$ (instead of just n) connected components $G_U^{(j)}$ where, for all $1 \leq j \leq 2n-2$ and $1 \leq h \leq d$, we place both a `0`-node and a `1`-node: we call these two nodes u_{jh}^0 and u_{jh}^1, respectively, to distinguish them from those in G_W. Moreover, every `e`-node of this gadget is connected with the next `b`-node, in terms of j coordinate (see Figure 2). As it can be seen, any subpattern P_{x_i} occurs in G_U, so it can be used as a "jolly" gadget.

2.3 Non-deterministic Graph

A possible approach is based on suitably combining one instance of gadget G_W and two instances of gadgets G_U, named G_{U1} and G_{U2}. The idea is that, when $x_i \cdot y_j = 0$, we want P to occur in G, so that the three conditions below hold.

- Instance G_{U1}: P_{x_1} occurs in $G_{U1}^{(n-1+j-(i-1))}$, …, $P_{x_{i-1}}$ occurs in $G_{U1}^{(n-1+j-1)}$.
- Instance G_W: P_{x_i} occurs in $G_W^{(j)}$.
- Instance G_{U2}: $P_{x_{i+1}}$ occurs in $G_{U2}^{(j)}$, …, P_{x_n} occurs in $G_{U2}^{(j+n-i-1)}$.

On the other hand, when $x_i \cdot y_j \neq 0$, we do not want P_{x_i} to occur in $G_W^{(j)}$. We can suitably link the instances G_W, G_{U1} and G_{U2} so that we get the above conditions: we connect the `e`-nodes in G_{U1} to `b`-nodes in G_W, the `e`-nodes in G_W to `b`-nodes in G_{U2} and we place additional starting `b`-nodes and additional ending `e`-nodes, to properly match the `bb` and `ee` prefix and suffix of P, respectively. However, even if G_W, G_{U1} and G_{U2} are deterministic, their resulting composition is not so, because of the out-neighbours of the `e`-nodes.[3] We show below how to obtain a deterministic graph by suitably merging G_W with portions of G_U.

2.4 Deterministic Graph

In order to obtain a *deterministic* DAG, we need to suitably combine one instance of gadget G_W with the two instances G_{U1} and G_{U2} (recall that both G_{U1} and G_{U2} have instances of gadget $G_U^{(j)}$, for all $1 \leq j \leq 2n-2$). While G_{U2} will be used as is, G_{U1} needs to be partially merged with G_W to obtain determinism. We start building our final graph G from G_W by adding parts of G_{U1} when needed, obtaining a deterministic graph called G_{U1W}, as shown in Figure 3. Consider subgraph $G_W^{(j)}$ and assume that the first position in which the `1`-node is lacking is h. We place a partial version of subgraph $G_{U1}^{(j')}, j' := n-1+j$, by adding to the graph the nodes and edges of $G_{U1}^{(j')}$ that are located between position $h+1$ and node

[3] An `e`-node can have two `b`-nodes as out-neighbors when linking G_{U1} to G_W, see [15].

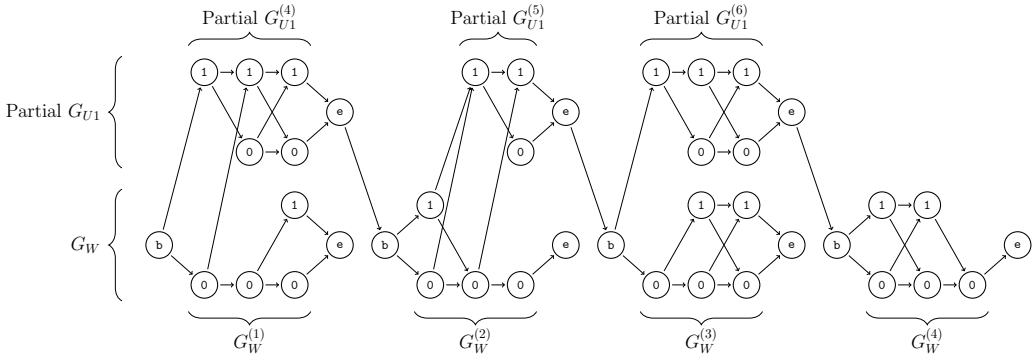


Figure 3 Graph G_{U1W} after merging G_{U1} (from Figure 2) with G_W (from Figure 1).

$e_{U1}^{(j')}$ (included). If $h = d$ we place only node $e_{U1}^{(j')}$. We also place 1-node u_{jh}^1 and we connect the 0-node and the 1-node (if any) of $G_W^{(j)}$ in position $h-1$ to it (if $h > 1$), or we connect $b_W^{(j)}$ to it (if $h = 1$). Moreover, we connect node u_{jh}^1 to the first 0- and 1-node of partial $G_{U1}^{(j')}$. If $h = d$ we connect u_{jh}^1 to $e_{U1}^{(j')}$. Then we scan $G_W^{(j)}$ from left to right looking for those positions h', $h \leq h' < d$, such that there is no 1-node in position $h'+1$. We connect the 0-node and the 1-node (if any) of $G_W^{(j)}$ in position h' to the 1-node of $G_{U1}^{(j')}$ in position $h'+1$. Finally, we place edge $(e_{U1}^{(j')}, b_W^{(j+1)})$. To complete the merging task, we apply the above modification to all $G_W^{(j)}$, for $1 \leq j \leq n-1$, and thus obtain gadget G_{U1W}.

At this point, we place gadget G_{U2} and we connect G_{U1W} to it by placing edges $(e_W^{(j)}, b_{U2}^{(j)})$, for all $1 \leq j \leq n$. Also, for every b-node of G_{U1W} we place an additional b-node as in-neighbor. We do the same for every e-node of G_{U2}, placing an e-node as out-neighbor. Adding subgraphs $G_{U1}^{(1)}, \ldots, G_{U1}^{(n-1)}$ with one additional b-node as in-neighbor of their b-nodes, and connecting the e-node of $G_{U1}^{(n-1)}$ to the b-node of $G_W^{(1)}$, completes the transformation into the wanted deterministic directed acyclic graph, which we call G. Figure 4 gives an overall picture of G.

It is easy to verify that every b- and e-node in G can have no more than two out-neighbours and, in such case, they have different labels. This shows that graph G is deterministic.

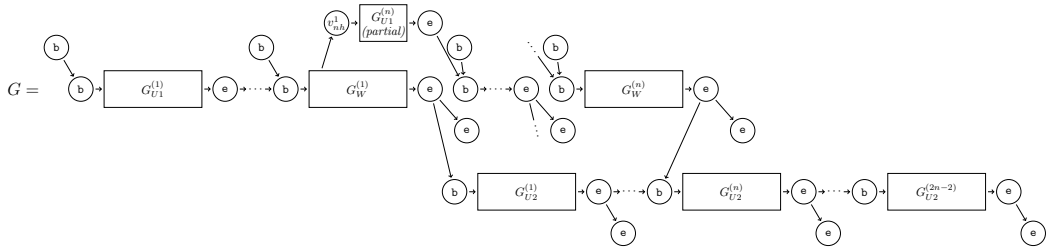


Figure 4 Final deterministic DAG G. In such graph there are $n-1$ instances of $G_{U1}^{(j)}$, $n-1$ partial instances of $G_{U1}^{(j)}$, n instances of $G_W^{(j)}$ and $2n-2$ instances of $G_{U2}^{(j)}$.

The deterministic DAG G has a crucial property which, combined with Lemma 3 and Lemma 4, is key to ensure the correctness of our reduction.

▶ **Lemma 5.** *Pattern P has a match in G if and only if a subpattern $\mathtt{b}P_{x_i}\mathtt{e}$ of P has a match in the underlying subgraph G_W of G_{U1W}.*

Proof. For the ($\Rightarrow$) implication, because of the directed eb-edges, each distinct subpattern $\mathtt{b}P_{x_i}\mathtt{e}$ matches a path from either a distinct portion of G_{U1W} (or from the $G_{U1}^{(j)}$ subgraphs, $1 \leq j \leq n-1$, before it) or G_{U2}. Moreover, each occurrence of P must begin with bb and end with ee. String bb can be matched only in G_{U1W} (or in the $G_{U1}^{(j)}$ subgraphs before it), hence the match must start here. On the other hand, string ee is found either in G_{U1W} or in G_{U2}. Observe that, by construction, once a match for pattern P is started in G_{U1W} (or in the $G_{U1}^{(j)}$ subgraphs before it), the only way to successfully conclude it is either by matching ee within G_{U1W}, or by matching also a portion of G_{U2} and then ee. Because of the structure of the graph, in both cases a subpattern $\mathtt{b}P_{x_i}\mathtt{e}$ of P must match one of the subgraphs $G_W^{(j)}$ that are present in G_{U1W}.

The ($\Leftarrow$) implication is trivial. In fact, if $\mathtt{b}P_{x_i}\mathtt{e}$ has a match in one subgraph $G_W^{(j)}$, then by construction we can match $\mathtt{b}P_{x_1}\mathtt{e}\ldots\mathtt{b}P_{x_{i-1}}\mathtt{e}$ possibly in the $G_{U1}^{(j)}$ subgraphs before G_{U1W}, then possibly in the partial $G_{U1}^{(j)}$ subgraphs of G_{U1W}. We can then match $\mathtt{b}P_{x_{i+1}}\mathtt{e}\ldots\mathtt{b}P_{x_n}\mathtt{e}$ in G_{U2}, and thus have a full match for P in G. ◀

We are now ready to prove our main result.

Proof of Theorem 1. First, we prove that the reduction is correct. Then we analyze its cost and show how a subquadratic-time algorithm for SMLG would contradict the OV hypothesis. Then, we explain how the graph can be modified to become a 3-DDAG using a binary alphabet.

Correctness. We need to ensure that pattern P has a match in G if and only if there exist vectors $x_i \in X$ and $y_j \in Y$ which are orthogonal. This follows from Lemma 5, which guarantees that P has a match in G if and only if a subpattern P_{x_i} has a match in G_W, and the fact that, by Lemma 4, this holds if and only if $x_i \cdot y_j = 0$.

Cost. As observed during the construction in Section 2.1 and Section 2.2, both pattern P and graph G have size $O(nd)$. Indeed, for each one of the n vectors $x_i \in X$ we place in P characters b and e plus d characters that can be either 0 or 1. In graph G, the size of each subgraph is proportional to the dimension d of the vectors and we place $O(n)$ of them.

Using the OV hypothesis. The last step is to show that any $O(|E|^{1-\epsilon}\, m)$-time or $O(|E|\, m^{1-\epsilon})$-time algorithm A for SMLG contradicts the OV hypothesis. Given two sets of vectors X and Y, we can perform our reduction obtaining pattern P and graph G in $O(nd)$ time, by observing that $|E| = O(nd)$ and $m = O(nd)$. No matter whether A has $O(|E|^{1-\epsilon}m)$ or $O(|E|\, m^{1-\epsilon})$ time complexity, we will end up with an algorithm deciding if there exists a pair of orthogonal vectors between X and Y in $O(nd \cdot (nd)^{1-\epsilon}) = O(n^{2-\epsilon}\mathrm{poly}(d))$ time, which contradicts the OV hypothesis.

Maximum sum of indegree and outdegree 3. Observe that every node in G can have at most 2 in-neighbours and 2 out-neighbours. An emblematic case is that of four nodes, say v, w, v', and w', with edges (v,w), (v,w'),(v',w), and (v',w'). To reduce to 1 the outdegree of v and v', and the indegree of w and w', the idea is to add two dummy nodes $\bar{v}$ and $\bar{w}$ connected by an edge $(\bar{v},\bar{w})$, and then replace the four edges above with $(v,\bar{v})$, $(v',\bar{v})$, $(\bar{w},w)$, and $(\bar{w},w')$. The dummy nodes can be labeled e.g. with 0 and then one can do a symmetric modification in the pattern. One needs to apply such transformations between any two consecutive columns of G.

Alphabet size. Our alphabet is of size 4. One can reduce the alphabet size to binary using the encoding $\alpha(\mathtt{0}) = \mathtt{0000}$, $\alpha(\mathtt{1}) = \mathtt{1111}$, $\alpha(\mathtt{b}) = \mathtt{10}$, and $\alpha(\mathtt{e}) = \mathtt{01}$ for both the pattern and the graph. (That is, we replace each σ-node with a path of as many nodes as characters in $\alpha(\sigma)$.) Observe that, after small adjustments that do not weaken our results, there is a bijection from matches before and after applying the encoding. ◀

As discussed in the Introduction, G has both nodes of indegree at most 1 (and outdegree more than 1), and nodes of outdegree at most 1 (and indegree more than 1). In the extended version of this work we will give a linear-time algorithm for directed graphs with nodes of only one such type.

3 Undirected Graphs: Zig-zag Matching

The lower bound given for the SMLG problem can cover the special case of an undirected graph with maximum degree 2. To this end, we need to modify the reduction defining a new alphabet, pattern and graph. The original alphabet $\Sigma = \{\mathtt{b}, \mathtt{e}, \mathtt{0}, \mathtt{1}\}$ is replaced with $\Sigma' = \{\mathtt{b}, \mathtt{e}, \mathtt{A}, \mathtt{B}, \mathtt{s}, \mathtt{t}\}$. Characters `1` and `0` are encoded in the following manner:

$\mathtt{1} = \mathtt{ABA}$ and $\mathtt{0} = \mathtt{ABABABA}$.

When such encoding is applied, character `s` will be used as a separator marking the beginning and the end of the old characters. As an example, the subpattern

$P_{x_i} = \mathtt{1\ 0\ 1}$ will be encoded as $P'_{x_i} = \mathtt{s\ ABA\ s\ ABABABA\ s\ ABA\ s}$.

A new pattern P' is built applying this encoding to each one of the subpatterns P_{x_i}, thus obtaining new subpatterns P'_{x_i}. We then concatenate all the subpatterns P'_{x_i} by placing the new character `t` to separate them, instead of `eb`. Finally, we place characters `bt` at the beginning of the new pattern, and `te` at the end. Here follows an example:

$P = \mathtt{bb\ 100\ e\ b\ 101\ ee}$

$$P' = \mathtt{b\ \ t\ s\ \ \underset{1}{ABA}\ \ s\ \ \underset{0}{ABABABA}\ \ s\ \ \underset{0}{ABABABA}\ \ s\ \ t\ s\ \ \underset{1}{ABA}\ \ s\ \ \underset{0}{ABABABA}\ \ s\ \ \underset{1}{ABA}\ \ s\ \ t\ e}$$

Note that for each subpattern we are introducing a constant number of new characters, hence the size of the entire pattern P' still is $O(nd)$.

An analogous encoding will be applied to the graph. The strategy is to encode G_W in an undirected path by concatenating subpaths representing each $G_W^{(j)}$, one after another.

The positions h in which both a `0`- and a `1`-node are present in $G_W^{(j)}$ are replaced by a path that can be matched both by $\mathtt{0} = \mathtt{ABABABA}$ and $\mathtt{1} = \mathtt{ABA}$. Positions h with only a `0`-node and no `1`-node are encoded instead with a path that can be matched only by $\mathtt{0} = \mathtt{ABABABA}$ (see Figure 5). We use `s`-nodes to separate these paths. We denote by $LG_W^{(j)}$ (*Linear* $G_W^{(j)}$) this linearized version of $G_W^{(j)}$. Moreover, given subgraph $G_W^{(j)}$, two new `t`-nodes will mark the beginning and the ending of its encoding. Figure 6 illustrates this transformation for $G_W^{(j)}$.

In a similar manner, G_U is also encoded as a path. We do not need to encode all its $2n-2$ subgraphs: since the matching path can go through nodes more than once, we only need to encode one of these subgraphs, in the same manner as done for $G_W^{(j)}$. Let LG_U be the linearized version of only one of the "jolly" gadgets that were composing the original G_U.

Then, for each $1 \leq j \leq n$, we build structure $LG^{(j)}$ by placing `t`-nodes, LG_U instances, $LG_W^{(j)}$, a `b`-node on the left and an `e`-node on the right, as in Figure 7. In such structure the `b`-node and the `e`-node delimit the beginning and the end of a viable match for a pattern. The `t`-nodes are separating the LG_U structures from $LG_W^{(j)}$ and, in general, they are marking the beginning and the end of a match for a subpattern P'_{x_i}. The idea behind $LG^{(j)}$ is that a match of P can traverse LG_U from beginning to end, backwards and forwards as many

(a) (b)

Figure 5 New substructures. (a) The old substructure is replaced by an undirected path that can match either `sABAs` (which represents `1`) by going forward only, or `sABABABAs` (which represents `0`), by going forward, backward, and forward again. (b) The an undirected path replacing a `0`-node can match only the string `sABABABAs`.

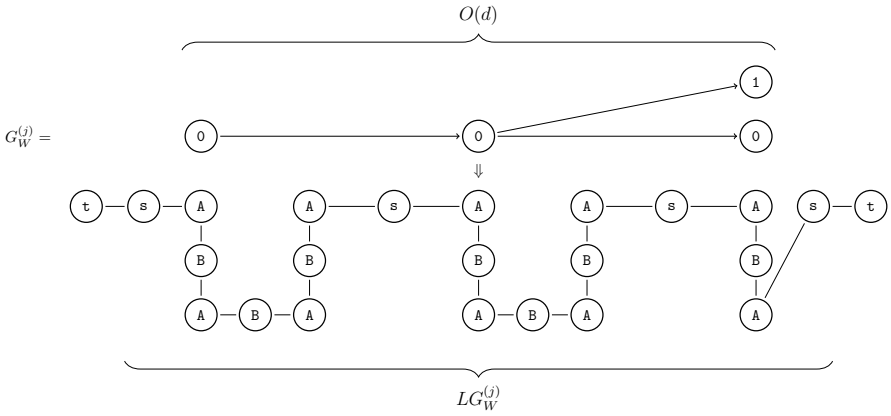


Figure 6 A subgraph $G_W^{(j)}$ is converted into a linear structure $LG_W^{(j)}$ using `s` as separator.

times as needed, before starting a match of some subpattern P'_{x_i} inside $LG_W^{(j)}$. Notice also that this allows only subpatterns on even positions i to match inside $LG_W^{(j)}$. We will address this minor issue at the end (see page 12).

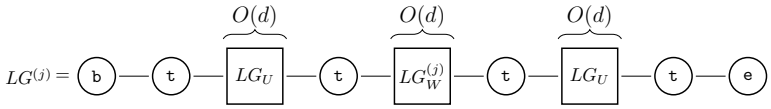


Figure 7 The $LG_W^{(j)}$ structure surrounded by two instances of LG_U. The `t`-nodes establish the beginning and the end of a match for a subpattern $\texttt{t}P'_{x_i}\texttt{t}$ while the `b`- and `e`-nodes are the starting and ending point for a match of the whole pattern P'.

In order to construct the final graph LG we concatenate all $LG^{(1)}$, $LG^{(2)}$, ..., $LG^{(n)}$ into a single undirected path. Figure 8 gives a picture of the end result.

No issues arise regarding the size of the graph, since we are replacing every `0`-node, or every pair of a `0`-node and a `1`-node, with a constant number of new nodes. By construction, the two gadgets LG_U and LG_W both have size $O(d)$, since for each one of the d entries of a vector we place one of the two possible encodings. In LG there are n instances of $LG_W^{(j)}$, each one surrounded by two LG_U instances. Hence the total size of the graph remains $O(nd)$.

In order to prove the correctness of the reduction, we will show some properties on LG by introducing the following lemmas, whose formal proofs will be available in the extended version of this work. We use $t_l LG_W^{(j)} t_r$ to refer to $LG_W^{(j)}$ extended with the `t`-nodes on its left and on its right. When referring to the k-th `s`-character in P'_{x_i} we mean the k-th `s`-character found scanning P'_{x_i} from left to right; in the same manner we refer to the k-th `s`-node in $LG_W^{(j)}$.

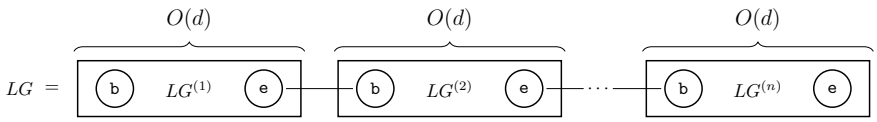


Figure 8 The final graph LG.

▶ **Lemma 6.** *If subpattern* $\mathtt{t}P'_{x_i}\mathtt{t}$ *has a match in* $t_l LG_W^{(j)} t_r$ *starting at* t_l *and ending at* t_r*, then the* k*-th* `s`*-character in* P'_{x_i} *matches the* k*-th* `s`*-node in* $LG_W^{(j)}$*, for all* $1 \leq k \leq d+1$*.*

▶ **Lemma 7.** *Subpattern* $\mathtt{t}P'_{x_i}\mathtt{t}$ *has a match in* $t_l LG_W^{(j)} t_r$ *starting at* t_l *and ending at* t_r *if and only if there exist* $y_j \in Y$ *such that* $x_i \cdot y_j = 0$*.*

The main difference with the original proof resides in assuming that a match for P'_{x_i} starts at t_l and ends at t_r. This feature is crucial for the correctness of the reduction and can be safely exploited since, as shown in the following, the `b`- and `e`-nodes guarantee that in case of a match for P' we will cross the $LG_W^{(j)}$ gadget from left to right at least once.

▶ **Lemma 8.** *Pattern* P' *has a match in* LG *if and only if there exist* i *and* j *such that* i *is even and subpattern* $\mathtt{t}P'_{x_i}\mathtt{t}$ *has a match in* $t_l LG_W^{(j)} t_r$ *starting at* t_l *and ending at* t_r*.*

Proof. For the ($\Rightarrow$) implication, first observe that the `b`- and `e`-nodes in LG are forcing a direction to follow. Let $LG_{Ul}^{(j)}$ and $LG_{Ur}^{(j)}$ be the LG_U gadgets to the left and to the right of $LG_W^{(j)}$, respectively. Since pattern P' starts with a `b` and ends with an `e`, a match can only start at the `b`-node on the left of $LG_{Ul}^{(j)}$ and end at the `e`-node on the right of $LG_{Ur}^{(j)}$, for some j. Hence $LG_W^{(j)}$ needs to be crossed by a match from left to right at least once. Thus, there must exist a subpattern $\mathtt{t}P'_{x_i}\mathtt{t}$ that has a match starting at t_l and ending at t_r. For such a pattern Lemma 7 applies. Moreover, because of our construction, only a subpattern on even position can achieve such a match.

The ($\Leftarrow$) implication is immediate since given a subpattern $\mathtt{t}P'_{x_i}\mathtt{t}$ which has a match in $t_l LG_U^{(j)} t_r$ one can match $\mathtt{bt}P'_{x_1}\mathtt{t}\ldots\mathtt{t}P'_{x_{i-1}}\mathtt{t}$ in $LG_{Ul}^{(j)}$ and $\mathtt{t}P'_{x_{i+1}}\mathtt{t}\ldots\mathtt{t}P'_{x_n}\mathtt{te}$ in $LG_{Ur}^{(j)}$ and have a full match for P' in LG. ◀

Since Lemma 8 gives us a property which holds only if a subpattern is in even position, we need to tweak pattern P' to make the reduction work. Indeed, we define two patterns. The first pattern $P'^{(1)}$ is P' itself; the second pattern $P'^{(2)}$ is obtained by swapping the subpatterns P'_{x_i} on odd position with the next subpatterns $P'_{x_{i+1}}$ on even position, for every $i = 1, 3, \ldots$. For example, if n is even, we will have:

$$P'^{(1)} = \mathtt{bt}\ P'_{x_1}\ \mathtt{t}\ P'_{x_2}\ \mathtt{t}\ P'_{x_3}\ \mathtt{t}\ P'_{x_4}\ \mathtt{t}\ \ldots\ \mathtt{t}\ P'_{x_{n-1}}\ \mathtt{t}\ P'_{x_n}\ \mathtt{te} = P'$$
$$P'^{(2)} = \mathtt{bt}\ P'_{x_2}\ \mathtt{t}\ P'_{x_1}\ \mathtt{t}\ P'_{x_4}\ \mathtt{t}\ P'_{x_3}\ \mathtt{t}\ \ldots\ \mathtt{t}\ P'_{x_n}\ \mathtt{t}\ P'_{x_{n-1}}\ \mathtt{te}$$

While $P'^{(1)}$ checks the even positions of P', $P'^{(2)}$ checks the odd ones. If n is even then the last subpattern would not have the chance to be matched against any $G_W^{(j)}$. In such case we can simply add a dummy subpattern $\bar{P} = \mathtt{s\ ABA\ s\ ABA\ s}\ldots\mathtt{s\ ABA\ s}$ (with d repetitions of `ABA`) at the end of P as it were its last subpattern, so that the number of subpatterns becomes odd. Indeed, observe that $\bar{P}$ corresponds to vector $\bar{x} = (11\ldots1)$, which has null product only with vector $\bar{y} = (00\ldots0)$. Hence if $\bar{y} \notin Y$ then $\bar{P}$ does not have a match in any $LG^{(j)}$, while if $\bar{y} \in Y$ every subpattern P'_{x_i} has a match in the $LG^{(j)}$ built on top of $\bar{y}$. This means that $\bar{P}$ does not disrupt our reduction.

Now we are ready to present the end result.

▶ **Lemma 9.** *Either $P'^{(1)}$ or $P'^{(2)}$ has a match in LG if and only if there exist vectors $x_i \in X$ and $y_j \in Y$ which are orthogonal.*

Proof. For ($\Rightarrow$) we assume that either $P'^{(1)}$ or $P'^{(2)}$ have a match in LG. By Lemma 8 this means that there exists a subpattern $P'^{(q)}_{x_i}$, $q \in \{1, 2\}$ which has a match in $LG^{(j)}_W$, for some j. Lemma 7 then ensures that $x_i \cdot y_j = 0$, thus x_i and y_j are orthogonal. For the other implication ($\Leftarrow$) we assume that there exists two orthogonal vectors $x_i \in X$ and $y_j \in Y$. Thanks to Lemma 7 we find a subpattern P'_{x_i} matching $LG^{(j)}_W$. By construction, P'_{x_i} has to be in even position either in $P'^{(1)}$ or in $P'^{(2)}$. By Lemma 8 this means that either $P'^{(1)}$ or $P'^{(2)}$ has a match in LG. ◀

Theorem 2 follows directly from the correctness of these constructions, except for the alphabet size reduction to binary, which will be covered in the extended version of this work.

References

1. Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight Hardness Results for LCS and Other Sequence Similarity Measures. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 59–78, 2015.
2. Tatsuya Akutsu. A linear time pattern matching algorithm between a string and a tree. In *4th Symposium on Combinatorial Pattern Matching, Padova, Italy*, pages 1–10, 1993.
3. Jarno Alanko, Alberto Policriti, and Nicola Prezza. On Prefix-Sorting Finite Automata. *arXiv e-prints*, page arXiv:1902.01088, February 2019. arXiv:1902.01088.
4. Amihood Amir, Moshe Lewenstein, and Noa Lewenstein. Pattern matching in hypertext. In *WADS'97, Halifax, LNCS 1272*, pages 160–173, 1997.
5. Amihood Amir, Moshe Lewenstein, and Noa Lewenstein. Pattern Matching in Hypertext. *J. Algorithms*, 35(1):82–99, 2000.
6. Renzo Angles and Claudio Gutierrez. Survey of Graph Database Models. *ACM Comput. Surv.*, 40(1):1:1–1:39, February 2008. doi:10.1145/1322432.1322433.
7. Arturs Backurs and Piotr Indyk. Edit Distance Cannot Be Computed in Strongly Subquadratic Time (Unless SETH is False). In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 51–58, New York, NY, USA, 2015. ACM. doi:10.1145/2746539.2746612.
8. Arturs Backurs and Piotr Indyk. Which Regular Expression Patterns Are Hard to Match? In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 457–466, 2016.
9. Arturs Backurs and Christos Tzamos. Improving Viterbi is Hard: Better Runtimes Imply Faster Clique Algorithms. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70, pages 311–321. PMLR, 2017.
10. Karl Bringmann and Marvin Kunnemann. Quadratic Conditional Lower Bounds for String Problems and Dynamic Time Warping. In *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, FOCS '15, pages 79–97, Washington, DC, USA, 2015. IEEE Computer Society. doi:10.1109/FOCS.2015.15.
11. The Computational Pan-Genomics Consortium. Computational pan-genomics: status, promises and challenges. *Briefings in Bioinformatics*, 19(1):118–135, 2018. doi:10.1093/bib/bbw089.
12. Alessio Conte, Gaspare Ferraro, Roberto Grossi, Andrea Marino, Kunihiko Sadakane, and Takeaki Uno. Node Similarity with q -Grams for Real-World Labeled Networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 1282–1291, 2018. doi:10.1145/3219819.3220085.

13 Mateus de Oliveira Oliveira and Michael Wehar. Intersection Non-emptiness and Hardness Within Polynomial Time. In *DLT*, volume 11088 of *Lecture Notes in Computer Science*, pages 282–290. Springer, 2018.

14 Massimo Equi, Roberto Grossi, and Veli Mäkinen. On the Complexity of Exact Pattern Matching in Graphs: Binary Strings and Bounded Degree. *arXiv e-prints*, page arXiv:1901.05264, January 2019. arXiv:1901.05264.

15 Massimo Equi, Roberto Grossi, Alexandru I. Tomescu, and Veli Mäkinen. On the Complexity of Exact Pattern Matching in Graphs: Determinism and Zig-Zag Matching. *arXiv e-prints*, page arXiv:1902.03560, February 2019. arXiv:1902.03560.

16 Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. Cypher: An Evolving Query Language for Property Graphs. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 1433–1445, 2018. doi:10.1145/3183713.3190657.

17 Travis Gagie, Giovanni Manzini, and Jouni Sirén. Wheeler graphs: A framework for BWT-based data structures. *Theor. Comput. Sci.*, 698:67–78, 2017. doi:10.1016/j.tcs.2017.06.016.

18 Daniel Gibney and Sharma V. Thankachan. On the Hardness and Inapproximability of Recognizing Wheeler Graphs. *arXiv e-prints*, page arXiv:1902.01960, February 2019. arXiv:1902.01960.

19 Shohei Hido and Hisashi Kashima. A Linear-Time Graph Kernel. In Wei Wang 0010, Hillol Kargupta, Sanjay Ranka, Philip S. Yu, and Xindong Wu, editors, *ICDM 2009, The Ninth IEEE International Conference on Data Mining, Miami, Florida, USA, 6-9 December 2009*, pages 179–188. IEEE Computer Society, 2009.

20 Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

21 Chirag Jain, Haowen Zhang, Yu Gao, and Srinivas Aluru. On the Complexity of Sequence to Graph Alignment. *bioRxiv*, 2019. To appear in RECOMB 2019. doi:10.1101/522912.

22 Donald E. Knuth, James H. Morris, and Vaughan R. Pratt. Fast Pattern Matching in Strings. *SIAM Journal on Computing*, 6(2):323–350, March 1977.

23 Antoine Limasset, Bastien Cazaux, Eric Rivals, and Pierre Peterlongo. Read mapping on de Bruijn graphs. *BMC Bioinformatics*, 17:237, 2016. doi:10.1186/s12859-016-1103-9.

24 U. Manber and S. Wu. Approximate string matching with arbitrary costs for text and hypertext. In *IAPR Workshop on Structural and Syntactic Pattern Recognition, Bern, Switzerland*, pages 22–33, 1992.

25 Gonzalo Navarro. Improved approximate pattern matching on hypertext. *Theoretical Computer Science*, 237(1-2):455–463, 2000.

26 K. Park and D. Kim. String matching in hypertext. In *6th Symposium on Combinatorial Pattern Matching, Espoo, Finland*, page 318, 1995.

27 Eric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF. World Wide Web Consortium, Recommendation REC-rdf-sparql-query-20080115, January 2008.

28 M. O. Rabin and D. Scott. Finite Automata and Their Decision Problems. *IBM Journal of Research and Development*, 3(2):114–125, April 1959. doi:10.1147/rd.32.0114.

29 Mikko Rautiainen and Tobias Marschall. Aligning sequences to general graphs in $O(V + mE)$ time. *bioRxiv*, pages 216–127, 2017.

30 Marko A. Rodriguez. The Gremlin graph traversal machine and language (invited talk). In *Proceedings of the 15th Symposium on Database Programming Languages, Pittsburgh, PA, USA, October 25-30, 2015*, pages 1–10, 2015. doi:10.1145/2815072.2815073.

31 Jouni Sirén, Niko Välimäki, and Veli Mäkinen. Indexing Graphs for Path Queries with Applications in Genome Research. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 11(2):375–388, March 2014. doi:10.1109/TCBB.2013.2297101.

32 Chris Thachuk. Indexing hypertext. *Journal of Discrete Algorithms*, 18:113–122, 2013. Selected papers from the 18th International Symposium on String Processing and Information Retrieval (SPIRE 2011). doi:10.1016/j.jda.2012.10.001.

33 Michael Wehar. Hardness Results for Intersection Non-Emptiness. In *ICALP (2)*, volume 8573 of *Lecture Notes in Computer Science*, pages 354–362. Springer, 2014.

34 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2):357–365, 2005. doi:10.1016/j.tcs.2005.09.023.

Unique End of Potential Line

John Fearnley
University of Liverpool, UK

Spencer Gordon
California Institute of Technology, Pasadena, CA, USA

Ruta Mehta
University of Illinois at Urbana-Champaign, IL, USA

Rahul Savani
University of Liverpool, UK

Abstract

The complexity class CLS was proposed by Daskalakis and Papadimitriou in 2011 to understand the complexity of important NP search problems that admit both path following and potential optimizing algorithms. Here we identify a subclass of CLS – called UniqueEOPL – that applies a more specific combinatorial principle that guarantees unique solutions. We show that UniqueEOPL contains several important problems such as the P-matrix Linear Complementarity Problem, finding Fixed Point of Contraction Maps, and solving Unique Sink Orientations (USOs). UniqueEOPL seems to a proper subclass of CLS and looks more likely to be the right class for the problems of interest. We identify a problem – closely related to solving contraction maps and USOs – that is complete for UniqueEOPL. Our results also give the fastest randomised algorithm for P-matrix LCP.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases P-matrix linear complementarity problem, unique sink orientation, contraction map, TFNP, total search problems, continuous local search

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.56

Category Track A: Algorithms, Complexity and Games

Related Version This paper substantially revises and extends the work described in our previous preprint "End of Potential Line" `arXiv:1804.03450` [21]. The full version of the paper is available at `https://arxiv.org/abs/1811.03841`.

Funding *John Fearnley*: Research supported by EPSRC grant EP/P020909/1.
Ruta Mehta: Research supported by NSF grant CCF-1750436.

Acknowledgements We thank Aviad Rubinstein and Kousha Etessami for alerting us to the work in [51], and we thank Rasmus Ibsen Jensen for helpful comments on a preprint of this paper.

1 Introduction

The complexity class TFNP contains search problems that are guaranteed to have a solution, and whose solutions can be verified in polynomial time [44]. While it is a semantically defined complexity class and thus unlikely to contain complete problems, a number of syntactically defined subclasses of TFNP have proven very successful at capturing the complexity of total search problems. In this paper, we focus on two in particular, PPAD and PLS. The class PPAD was introduced in [49] to capture the difficulty of problems that are guaranteed total by a parity argument. It has attracted intense attention in the past decade, culminating in a series of papers showing that the problem of computing a Nash-equilibrium in two-player games is PPAD-complete [10, 13], and more recently a conditional lower bound that rules out a PTAS for the problem [52]. No polynomial-time algorithms for PPAD-complete problems

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 56; pp. 56:1–56:15


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

are known, and recent work suggests that no such algorithms are likely to exist [4, 25]. PLS is the class of problems that can be solved by local search algorithms (in perhaps exponentially-many steps). It has also attracted much interest since it was introduced in [38], and looks similarly unlikely to have polynomial-time algorithms. Examples of PLS-complete problems include computing: a pure Nash equilibrium in a congestion game [19], a locally optimal max cut [53], or a stable outcome in a hedonic game [24].

If a problem lies in PPAD and PLS then it is unlikely to be complete for either class, since this would imply an extremely surprising containment of one class in the other. Daskalakis and Papadimitriou [14] observed that several prominent total function problems for which no polynomial-time algorithms are known lie in PPAD ∩ PLS. Motivated by this they introduced CLS, a syntactically defined subclass of PPAD ∩ PLS, that captures optimization problems over a continuous domain in which a continuous potential function is being minimized, with access to a polynomial-time continuous improvement function. They showed that many well-studied problems are in CLS, including the problem of solving a simple stochastic game, the more general problems of solving a P-matrix Linear Complementarity Problem, finding an approximate fixpoint to a contraction map, finding an approximate stationary point of a multivariate polynomial, and finding a mixed Nash equilibrium of a congestion game. In this paper we study an interesting subset of CLS consisting of problems with *unique* solutions.

Contraction. In this problem we are given a function $f : \mathbb{R}^d \to \mathbb{R}^d$ that is purported to be c-contracting, meaning that for all points $x, y \in [0, 1]^n$ we have $d(f(x), f(y)) \leq c \cdot d(x, y)$, where c is a constant satisfying $0 < c < 1$, and d is a distance metric. Banach's fixpoint theorem states that if f is contracting, then it has a unique *fixpoint* [3], meaning that there is a unique point $x \in \mathbb{R}^d$ such that $f(x) = x$.

P-LCP. The *P-matrix linear complementarity problem* (P-LCP) is a variant of the linear complementarity problem in which the input matrix is a P-matrix [12]. An interesting property of this problem is that, if the input matrix actually is a P-matrix, then the problem is guaranteed to have a unique solution [12]. Designing a polynomial-time algorithm for P-LCP has been open for decades, at least since the 1978 paper of Murty [47] that provided exponential-time examples for *Lemke's algorithm* [42] for P-LCPs.

USO. A *unique sink orientation* (USO) is an orientation of the edges of an n-dimensional hypercube such that every face of the cube has a unique sink. Since the entire cube is a face of itself, this means that there is a unique vertex of the cube that is a sink, meaning that all edges are oriented inwards. The USO problem is to find this *unique* sink.

All of these problems are most naturally stated as *promise* problems, since we have no way of efficiently verifying whether a function is contracting, whether a matrix is a P-matrix, or whether an orientation is a USO. Hence, it makes sense, for example, to study the contraction problem where it is promised that the function f is contracting, and likewise for the other two. However, each of these problems can be turned into non-promise problems that lie in TFNP. In the case of Contraction, if the function f is not contracting, then there exists a short certificate of this fact. Specifically, any pair of points $x, y \in \mathbb{R}^d$ such that $d(f(x), f(y)) > c \cdot d(x, y)$ give an explicit proof that the function f is not contracting. We call these *violations*, since they witness a violation of the promise inherent in the problem.

So, Contraction can be formulated as the non-promise problem of either finding a solution or finding a violation. This problem is in TFNP because in the case where there is not a unique solution, there must exist a violation of the promise. The P-LCP and USO problems also have violations that can be witnessed by short certificates, and so they can be turned into non-promise problems in the same way, and these problems also lie in TFNP. For Contraction and P-LCP we actually know that both are in CLS [14]. Prior to this work USO was not known to lie in any non-trivial subclass of TFNP, and placing USO into a non-trivial subclass of TFNP was identified as an interesting open problem by Kalai [39, Problem 6].

We remark that not every problem in CLS has the uniqueness properties that we identify above. For example, the KKT problem [14] lies in CLS, but has no apparent notion of having a unique solution. The problems that we study share the special property that there is a natural promise version of the problem, and that promise problem has a unique solution.

Our contributions. We define the complexity classes PromiseUEOPL and UniqueEOPL to capture problems in CLS that have unique solutions. We argue that UniqueEOPL is likely to be a strict subset of CLS. We introduce the notion of *promise-preserving reductions*, which allow us to simultaneously obtain results for the promise and non-promise versions of problems. We show that all of our motivating problems – USO, P-LCP, and finding a fixpoint of a Piecewise-Linear Contraction under an ℓ_p-norm – are contained in UniqueEOPL (PromiseUEOPL for the promise versions) via promise-preserving reductions. Thus, we resolve the open problem of Kalai mentioned above, by showing that USO is in UniqueEOPL and thus also CLS, PPAD and PLS. Our results also imply that parity, mean-payoff, discounted, and simple-stochastic games lie in UniqueEOPL. We also provide a complete problem for UniqueEOPL, called One-Permutation Discrete Contraction (OPDC). It is motivated by a discretized version of contraction, but it is also closely related to USO, and we consider its hardness to be a substantial step towards showing hardness for contraction and USO.

The new techniques used in our reductions also lead to new algorithmic results. We obtain direct polynomial-time algorithms for finding fixpoints of contraction maps in fixed dimension for any ℓ_p norm, where previously such algorithms relied on a reduction to the Tarski fixpoint problem [51]. Our reduction for P-LCP allows a technique of Aldous [2] to be applied, which in turn gives the fastest-known randomized algorithm for P-LCP.

A main message of our paper is that several important problems lie in UniqueEOPL *and* that UniqueEOPL is likely to be a proper subset of CLS.

Related work. Hubáček and Yogev [36] proved lower bounds for CLS. They introduced a problem known as EndOfMeteredLine which they showed was in CLS, and for which they proved a query complexity lower bound of $\Omega(2^{n/2}/\sqrt{n})$ and hardness under the assumption that there were one-way permutations and indistinguishability obfuscators for problems in $\mathsf{P}_{/\mathsf{poly}}$. Recently, two variants of ContractionMap have been shown to be CLS-complete. Whereas in the original definition of ContractionMap it is assumed that an ℓ_p or ℓ_∞ norm is fixed, and the contraction property is measured w.r.t. the induced metric, in these two complete variants, a metric [15] and meta-metric [20] are given as input to the problem.

Papadimitriou showed that P-LCP, the problem of solving the LCP or returning a violation of the P-matrix property, is in PPAD [49] using Lemke's algorithm. The relationship between Lemke's algorithm and PPAD has been studied by Adler and Verma [1]. Later, Daskalakis and Papadimitrou showed that P-LCP is in CLS [14], using the potential reduction method in [41]. Many algorithms for P-LCP have been studied, e.g., [47, 46, 40]. However, no polynomial-time algorithms are known for P-LCP. The best-known algorithms for P-LCP are based on a reduction to Unique Sink Orientations (USOs) of cubes [59]. For an P-matrix LCP of size n, the USO algorithms of [60] apply, and give a deterministic algorithm that runs in time $O(1.61^n)$ and a randomized algorithm with expected running time $O(1.43^n)$. The application of Aldous' algorithm [2] to the UniqueEOPL instance that we produce from a P-matrix LCP takes expected time $2^{n/2} \cdot \mathrm{poly}(n) = O(1.4143^n)$ in the worst case.

We study USOs of cubes, a problem that was first studied by Stickney and Watson [59] in the context of P-matrix LCPs. Motivated by Linear Programming, *acylic* USOs (AUSOs) have also been studied, both for cubes and general polytopes [33, 28]. Recently Gärtner

and Thomas studied the computational complexity of recognizing USOs and AUSOs [29]. A series of papers provide upper and lower bounds for approaches for solving (A)USOs, including [60, 31, 22, 43, 55, 23, 61, 26, 54]. To the best of our knowledge, we are first to study the general problem of solving a USO from a complexity-theoretic point of view.

The problem of computing a fixpoint of a continuous map $f : \mathcal{D} \to \mathcal{D}$ with Lipschitz constant c has been extensively studied, in both continuous and discrete variants [9, 8, 16]. For arbitrary maps with $c > 1$, exponential bounds on the query complexity are known [32, 7]. In [6, 35, 58], algorithms for computing fixpoints of weakly ($c = 1$) and strictly ($c < 1$) contracting maps are studied.

A number of algorithms are known for contractions w.r.t. the ℓ_2 norm [48, 34, 57]. There is an exponential lower bound for absolute approximation with $c = 1$ [57]. For relative approximation ($||x - f(x)|| \leq \epsilon$) in dimension d, an $O(d \cdot \log 1/\epsilon)$ time algorithm is known [34]. For absolute approximation ($||x - x^*|| \leq \epsilon$ where x^* is an exact fixpoint) with $c < 1$, an ellipsoid-based algorithm with time complexity $O(d \cdot [\log(1/\epsilon) + \log(1/(1-c))])$ is known [34]. For the ℓ_∞ norm, [56] gave an algorithm to find an ϵ-relative approximation in time $O(\log(1/\epsilon)^d)$ which is polynomial for constant d. A polynomial time algorithm for finding an approximate fixpoint of a contraction map in constant dimension can be obtained through a reduction to the Tarski fixpoint problem [51].

2 Unique End of Potential Line

We define two new complexity classes called EOPL and UniqueEOPL. EOPL combines EndOfLine and SinkOfDag, the canonical complete problems for PPAD and PLS [49].

▶ **Definition 1** (EndOfPotentialLine). *Given Boolean circuits $S, P : \{0,1\}^n \to \{0,1\}^n$ such that $P(0^n) = 0^n \neq S(0^n)$ and a Boolean circuit $V : \{0,1\}^n \to \{0,1,\ldots,2^m - 1\}$ such that $V(0^n) = 0$ find one of the following:*
(R1) *A point $x \in \{0,1\}^n$ such that $S(P(x)) \neq x \neq 0^n$ or $P(S(x)) \neq x$.*
(R2) *A point $x \in \{0,1\}^n$ such that $x \neq S(x)$, $P(S(x)) = x$, and $V(S(x)) - V(x) \leq 0$.*
This problem defines an exponentially large graph where each vertex has in-degree and out-degree at most one (as in EndOfLine) that is also a DAG (as in SinkOfDag). An edge exists from x to y if and only if $S(x) = y$, $P(y) = x$, and $V(x) < V(y)$. Only some bit-strings encode vertices. Specifically, if $S(x) = x$ for some bit-string x, then x does *not* encode a vertex. The problem consists of a single instance that is simultaneously an instance of EndOfLine and an instance of SinkOfDag. To solve the problem, it suffices to solve *either* of these problems. Solutions of type 1 are ends of lines, and solutions of type 2 are points where the potential does not increase along an edge.

We define the complexity class EOPL to consist of all problems that can be reduced in polynomial time to EndOfPotentialLine. We show the following containment.

▶ **Theorem 2.** EOPL $\subseteq$ CLS.

To prove this, we reduce EndOfPotentialLine to the EndOfMeteredLine, which was defined and shown to be in CLS by Hubáček and Yogev [36]. The difference between the two problems is that EndOfMeteredLine requires that the potential increases by *exactly* one along each edge. Our reduction inserts new vertices into the instance to satisfy this property.

2.1 Promise problems with unique solutions

Each of the problems that we study have a *promise*, and if the promise is satisfied the problem has a *unique* solution. For example, in the contraction problem, we are given a function as a circuit but cannot efficiently check whether the function is actually contracting.

If the function is contracting, then Banach's fixpoint theorem states that it has a unique fixpoint [3]. If it is not contracting, there exist *violations* that can be witnessed by short certificates. We can use violations to formulate the problem as a non-promise problem that lies in TFNP: we ask for either a fixpoint or a violation of contraction.

When we place this type of problem in EOPL, we obtain an instance with extra properties. Specifically, if the original problem has no violations, i.e., the promise is satisfied, then the EndOfPotentialLine instance will contain a *single* line that starts at 0^n, and ends at the unique solution. So, if we ever find two distinct lines, we immediately know that the instance fails to satisfy the promise. We define the following problem to capture these properties.

▶ **Definition 3** (UniqueEOPL). *Given Boolean circuits $S, P : \{0,1\}^n \to \{0,1\}^n$ such that $P(0^n) = 0^n \neq S(0^n)$ and a Boolean circuit $V : \{0,1\}^n \to \{0,1,\ldots,2^m-1\}$ such that $V(0^n) = 0$ find one of the following:*

(U1) *A point $x \in \{0,1\}^n$ such that $P(S(x)) \neq x$.*

(UV1) *A point $x \in \{0,1\}^n$ such that $x \neq S(x)$, $P(S(x)) = x$, and $V(S(x)) - V(x) \leq 0$.*

(UV2) *A point $x \in \{0,1\}^n$ such that $S(P(x)) \neq x \neq 0^n$.*

(UV3) *Two points $x, y \in \{0,1\}^n$, such that $x \neq y$, $x \neq S(x)$, $y \neq S(y)$, and either $V(x) = V(y)$ or $V(x) < V(y) < V(S(x))$.*

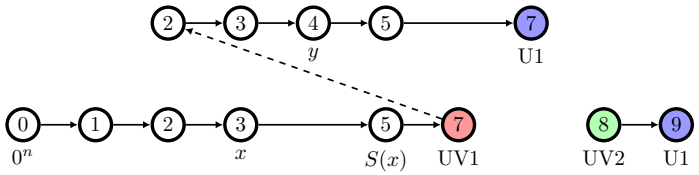


Figure 1 UniqueEOPL instance with 3 lines. The *main line* starts at 0^n and ends with a UV1 solution. There is a final line of length one to the bottom right, whose start vertex is a UV2 solution. The ranges of potential values for the main line and top line intersect, so they contribute many UV3 solutions. We highlight one on the diagram with x, $S(x)$, and y, such that $V(x) < V(y) < V(S(x))$.

We split solutions into two types: proper solutions and violations. Solutions of type 1 encode the end of a line, which are the proper solutions. 1 violations are vertices at which the potential fails to increase. 2 violations are the start of any line other than 0^n. 3 violations are a different witness that there are more than one line, namely a pair of vertices x and y, with either $V(x) = V(y)$, or such that $V(y)$ lies between $V(x)$ and $V(S(x))$, so x and y cannot lie on the same line. See Figure 1 for an illustration.

We remark that 1 and 2 violations already capture the property that "there is a unique line", since if we exclude them, then a second line cannot exist. However, with only these two violations, we may find two vertices on two different lines, but both may be exponentially many steps away from the start of their respective lines. 3 violations make any pair of vertices that are provably on two different lines a violation. All of the problems in UniqueEOPL have the property that if we ever find a 3 violation, the problem can be solved immediately.

We define UniqueEOPL to be the class of problems that can be reduced in polynomial time to UniqueEOPL[1]. For each of our problems, it is also interesting to consider the promise variant, in which it is guaranteed via a promise that no violations exist. We define

[1] We remark that Hubáček and Yogev [36] mention that their lower bound results for CLS may also apply to such problems, but they did not investigate a corresponding complexity class.

PromiseUniqueEOPL to be the promise version of UniqueEOPL in which it is promised that 0^n is the only start of a line, and PromiseUEOPL to be the class of promise problems that can be reduced in polynomial time to PromiseUniqueEOPL.

The problem UniqueEOPL has the interesting property that, if it is promised that there are no violation solutions, then there must be a unique solution. All of the problems that we study in this paper share this property, and indeed when we reduce them to UniqueEOPL, the resulting instance will have a unique line whenever the original problem has no violation solutions. We formalise this by defining the concept of a *promise-preserving* reduction. This is a reduction between two problems A and B, both of which have proper solutions and violation solutions. The reduction is promise-preserving if, when it is promised that A has no violations, then the resulting instance of B also has no violations. Hence, if we reduce a problem to UniqueEOPL via a chain of promise-preserving reductions, and we know that there are no violations in the original problem, then there is a unique line ending at the unique proper solution in the instance. So, if we show that a problem is in UniqueEOPL (or UniqueEOPL-complete) via a chain of promise-preserving reductions, then we automatically get that the promise version of that problem, where it is promised that there are no violations, lies in PromiseUEOPL (or PromiseUEOPL-complete).

3 One-Permutation Discrete Contraction (OPDC)

OPDC plays a crucial role in our results. We show that it lies in UniqueEOPL, and the we reduce both PL-Contraction and Unique-Sink-Orientation to it, thereby showing that those problems also lie in UniqueEOPL. We also show that UniqueEOPL can be reduced to OPDC, making it the first example of a non-trivial UniqueEOPL-complete problem.

Direction functions. OPDC can be seen as a discrete variant of the continuous contraction problem. A contraction map is a function $f : [0,1]^n \to [0,1]^d$ that is contracting under a metric d, i.e., $d(f(x), f(y)) \leq c \cdot f(x,y)$ for all $x, y \in [0,1]^d$ and some constant c satisfying $0 < c < 1$. We discretize this by overlaying a grid of points on the $[0,1]^d$ cube. Let $[k]$ denote the set $\{0, 1, \ldots, k\}$. Given a tuple of grid widths $(k_1, k_2, \ldots, k_d)$, we define the set $P(k_1, k_2, \ldots, k_d)$ as $[k_1] \times [k_2] \times \cdots \times [k_d]$. We sometimes refer to $P(k_1, k_2, \ldots, k_d)$ simply as P. Note that each point $p \in P$ is a tuple $(p_1, p_2, \ldots, p_d)$, where p_i is an integer between 0 and k_i, and this point maps onto the point $(p_1/k_1, p_2/k_2, \ldots, p_d/k_d) \in [0,1]^d$.

Instead of a single function f, in the discretized problem we use a family of *direction functions* over the grid P. For each dimension $i \leq d$, we have function $D_i : P \to \{\mathsf{up}, \mathsf{down}, \mathsf{zero}\}$. Intuitively, the natural reduction from a contraction map f to a family of direction functions would, for each point $p \in P$ and each dimension $i \leq d$ set: $D_i(p) = \mathsf{up}$ whenever $f(p)_i > p_i$, $D_i(p) = \mathsf{down}$ whenever $f(p)_i < p_i$, and $D_i(p) = \mathsf{zero}$ whenever $f(p)_i = p_i$. In other words, the function D_i simply outputs whether $f(p)$ moves up, down, or not at all in dimension i. So a point $p \in P$ with $D_i(p) = \mathsf{zero}$ for all i would correspond to the fixpoint of f.

A 2d example. To illustrate this definition, consider the 2d instance given in the two leftmost parts of Figure 2, which we use as a running example. These are two direction functions: the left one shows a direction function for the up-down dimension, which we will call dimension 1 and illustrate in blue. The right one shows the left-right dimension, which we will call dimension 2 and illustrate in red. Each square represents a point in the discretized space, and the value of the direction function is shown inside the box. Note that there is exactly one point p where $D_1(p) = D_2(p) = \mathsf{zero}$, which is the fixpoint that we seek.

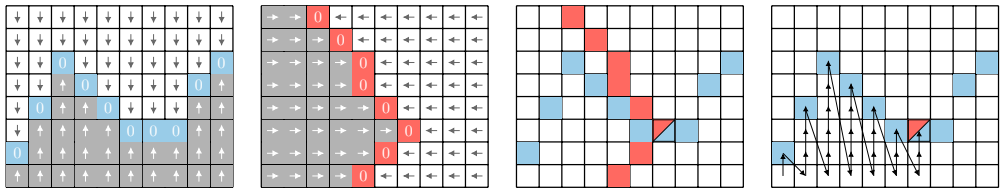


Figure 2 This figure should be viewed in color. From left to right: A direction function for the up/down dimension; a direction function for the left/right dimension; the red and blue surfaces; the path that we follow.

Slices. We will frequently refer to subsets of P in which some dimensions have been fixed. A *slice* is represented as a tuple $(s_1, s_2, \ldots, s_d)$, where each s_i is either: a number in $[k_i]$, which indicates that dimension i should be fixed to s_i; or the special symbol $*$, which indicates that dimension i is free to vary. We define Slice_d to be the set of all possible slices in dimension d. Given a slice $s \in \mathsf{Slice}_d$, we define $P_s \subseteq P$ to be the set of points in that slice, i.e., P_s contains every point $p \in P$ such that $p_i = s_i$ whenever $s_i \neq *$. We say that a slice $s' \in \mathsf{Slice}_d$ is a sub-slice of a slice $s \in \mathsf{Slice}_d$ if $s_j \neq * \implies s'_j = s_j$ for all $j \in [d]$. An *i-slice* is a slice s for which $s_j = *$ for all $j \leq i$, and $s_j \neq *$ for all $j > i$. In other words, all dimensions up to and including dimension i are allowed to vary, while all other dimensions are fixed.

In our 2d example, there are three types of i-slices. There is one 2-slice: the slice $(*, *)$ that contains every point. For each x, there is a 1-slice $(*, x)$, which restricts the left/right dimension to the value x. For each pair x, y there is a 0-slice (y, x), which contains only the exact point corresponding to x and y.

The OPDC problem. Let P be a grid of points in dimension d and $\mathcal{D} = (D_i)_{i=1,\ldots,d}$ a family of direction functions over P. We say that a point $p \in P_s$ in some slice s is a *fixpoint* of s if $D_i(p) = \mathsf{zero}$ for all dimensions i where $s_i = *$. The promise version of OPDC promises that for every i-slice s, the following conditions hold.

1. There is a unique fixpoint of s.
2. Let $s' \in \mathsf{Slice}_d$ be a sub-slice of s where some coordinate i for which $s_i = *$ has been fixed. If q is the unique fixpoint of s, and p is the unique fixpoint of s', then $p_i < q_i$ implies $D_i(p) = \mathsf{up}$, and $p_i > q_i$ implies $D_i(p) = \mathsf{down}$.

Since the slice $(*, *, \ldots, *)$ is an i-slice, the first condition implies that all i-slices including the full problem have a unique fixpoint. Intuitively, the second condition just ensures that the D_i behave as direction functions. It says that if we have found the unique fixpoint p of the $(i+1)$-slice s', and if it is not the unique fixpoint of the i-slice s, then $D_i(p)$ tells us which way to walk to find the fixpoint of s. This is a crucial property used in our reduction from OPDC to UniqueEOPL, and in our algorithms for contraction maps.

In our 2d example, the first condition requires that every slice $(*, x)$ has a unique fixpoint, i.e., in every column there is a unique blue zero. The second condition says that, if we are at some blue zero, then the red direction function at that point tells us the direction of the overall fixpoint. Our example satisfies both conditions. We next define a total variant of OPDC that uses violations to cover the cases where $\mathcal{D}$ fails to satisfy these two conditions.

▶ **Definition 4** (OPDC). *Given a tuple $(k_1, k_2, \ldots, k_d)$ and circuits $(D_i(p))_{i=1,\ldots,d}$, where each circuit $D_i : P(k_1, k_2, \ldots, k_d) \to \{\mathsf{up}, \mathsf{down}, \mathsf{zero}\}$, find one of the following*

(O1) *A point $p \in P$ such that $D_i(p) = \mathsf{zero}$ for all i.*

(OV1) *An i-slice s and points $p, q \in P_s$ with $p \neq q$ s.t. $D_j(p) = D_j(q) = \mathsf{zero}$ for all $j \leq i$.*

(OV2) *An i-slice s and points $p, q \in P_s$ s.t. $D_j(p) = D_j(q) = \mathsf{zero}$ for all $j < i$, $p_i = q_i + 1$, and $D_i(p) = \mathsf{down}$ and $D_i(q) = \mathsf{up}$.*

(OV3) *An i-slice s and a point $p \in P_s$ s.t. $D_j(p) = D_j(q) = \mathsf{zero}$ for all $j < i$, and either $p_i = 0$ and $D_i(p) = \mathsf{down}$, or $p_i = k_i$ and $D_i(p) = \mathsf{up}$.*

Solution type 1 encodes a fixpoint, which is the proper solution of OPDC. Solution type 1 witnesses a violation of the fact that each i-slice should have a unique fixpoint, by giving two different points p and q that are both fixpoints of the same i-slice. Solutions of type 2 witness violations of the first and second properties. In these solutions we have two points p and q that are both fixpoints of their respective $(i-1)$-slices and are directly adjacent in an i-slice s. If there is a fixpoint r of the slice s, then this witnesses a violation of the fact that $D_i(p)$ and $D_i(q)$ should both point towards r, since clearly one of them does not. On the other hand, if slice s has no fixpoint, then p and q also witness this fact, since the fixpoint should be in-between p and q, which is not possible. Solutions of type 3 consist of a point p that is a fixpoint of its $(i-1)$-slice but $D_i(p)$ points outside the boundary of the grid. These are violations because $D_i(p)$ should point towards the fixpoint of the i-slice containing p, but that fixpoint cannot be outside the grid.

It is perhaps not immediately obvious that OPDC is a total problem. Our promise-preserving reduction from OPDC to UniqueEOPL proves totality, and shows that if the OPDC instance has no violations then it has a unique solution. The prefix *One-Permutation* was chosen to emphasize that our solution conditions only consider i-slices. In the continuous contraction map problem with an ℓ_p metric, *every* slice has a unique fixpoint, and our reduction from contraction maps to OPDC works for any permutation of the dimensions.

3.1 One-Permutation Discrete Contraction is UniqueEOPL-complete

To show that OPDC lies in UniqueEOPL under promise-preserving reductions, we make use of an intermediate problem that we call UniqueForwardEOPL, which is a version of UniqueEOPL in which we only have a successor circuit S, meaning that no predecessor circuit P is given. Without this circuit, there is no way to tell if a vertex is the start of a line, so the only solutions to this problem are the end of a line, a vertex at which the potential fails to increase, or the analogue of 3. Although we no longer have a predecessor circuit, it is still the case that if the problem is promised to have no violations, then it must contain a single line ending at the unique proper solution. We reduce OPDC to UniqueForwardEOPL, and then reduce UniqueForwardEOPL to UniqueEOPL.

An illustration of the reduction. We illustrate using the 2d example shown in Figure 2. The reduction uses the notion of a *surface*. The surface of a direction function D_i is exactly the set of points $p \in P$ such that $D_i(p) = \mathsf{zero}$. In the third part of Figure 2, we have overlaid the surfaces of the two direction functions from Figure 2. The fixpoint p that we seek has $D_i(p) = \mathsf{zero}$ for all dimensions i, and so it lies at the intersection of these surfaces.

To reach the overall fixpoint, we walk along a path starting from the bottom-left corner, which is shown on the rightmost part of Figure 2. The path begins by walking upwards until it finds the blue surface. Once it has found the blue surface, it then there are two possibilities: either we have found the overall fixpoint, in which case the line ends, or we have not found the overall fixpoint, and the red direction function tells us that the direction of the overall fixpoint is to the right. If we have not found the overall fixpoint, then we move one step to the right, go back to the bottom of the diagram, and start walking upwards again. We keep repeating this until we find the overall fixpoint.

How do we define a potential? Observe that the dimension-two coordinates of the points on the line are weakly monotone, i.e., the line never moves to the left. Furthermore, for any dimension-two slice (any slice in which the left/right coordinate is fixed), the dimension-one coordinate is increasing. So, if $p = (p_1, p_2)$ denotes any point on the line, if k denotes the maximum coordinate in either dimension, then the function $V(p_1, p_2) = k \cdot p_2 + p_1$ is a function that monotonically increases along the line, which we can use as a potential function.

Uniqueness. For a promise-preserving reduction, the line must be unique whenever the OPDC instance has no violations. To do this we must be careful that only points that are to the left of the fixpoint are actually on the line, and that no "false" line exists to the right of the fixpoint. Here we rely on the following fact: if the line visits a point with coordinate x in dimension 2, then it must have visited the point p on the blue surface in the slice defined by $x - 1$. Moreover, for that point p we must have $D_2(p) = \mathsf{up}$, which means that it is to the left of the overall fixpoint. Using this fact, each vertex on our line will be a pair (p, q), where p is the current point that we are visiting, and q is either the symbol $-$, indicating that we are still in the first column of points, and we have never visited a point on the blue surface, or a point q that is on the blue surface that satisfies $q_2 = p_2 - 1$ and $D_2(q) = \mathsf{up}$. Hence q is always the last point that we visited on the blue surface, which provides a witness that we have not yet walked past the overall fixpoint. When we finish walking up a column, and find the point on the blue surface, we overwrite q with the new point. This step is not easily reversible, since to determine the predecessor of a vertex we would need to recover the value that was overwritten. So we create a UniqueForwardEOPL instance, and our onwards reduction to UniqueEOPL will produce a predecessor circuit.

Violations. Our 2d example does not contain any violations, but our reduction can still handle all possible violations in the OPDC instance. At a high level, there are two possible ways in which the reduction can go wrong if there are violations.

1. It is possible, that as we walk upwards in some column, we do not find a fixpoint, and our line will get stuck. In our 2d example, this case corresponds to a column of points in which there is no point on the blue surface. However, if there is no point on the blue surface, then we will either: find two adjacent points p and q in that column with $D_1(p) = \mathsf{up}$ and $D_2(p) = \mathsf{down}$, which is a solution of type 2, or find a point p at the top of the column with $D_1(p) = \mathsf{up}$, or a point q at the bottom of the column with $D_1(q) = \mathsf{down}$. Both of these are solutions of type 3. There is also the similar case where we walk all the way to the right without finding an overall fixpoint, in which case we will find a point p on the right-hand boundary that satisfies $D_1(p) = \mathsf{zero}$ and $D_2(p) = \mathsf{up}$, which is a solution of type 3.
2. The other possibility is that there may be more than one point on the blue surface in some columns. This inevitably leads to multiple lines, since if q and q' are both on the blue surface in some column, and p is in the column to the right of p and q, then (p, q) and (p, q') will both be valid vertices on two different lines. We map these violations back to solutions of type 1. Specifically, the points p and q, which are given as part of the two vertices, are both fixpoints of the same slice, which is exactly what 1 asks for.

Our reduction is promise-preserving because violations in the UFEOPL instance are never mapped back to proper solutions of the OPDC instance.

Generalizing to d dimensions. The full reduction from OPDC to UniqueForwardEOPL generalizes the approach given above to d dimensions. We say that a point $p \in P$ is on the *i-surface* if $D_j(p) = \mathsf{zero}$ for all $j \leq i$. In our 2d example we followed a line of points

on the 1-surface, in order to find a point on the 2-surface. In between any two points on the 1-surface, we followed a line of points on the 0-surface (every point is trivially on the 0-surface). Our line will visit a sequence of points on the $(d-1)$-surface in order to find the point on the d-surface, which is the fixpoint. Between any two points on the $(d-1)$-surface the line visits a sequence of points on the $(d-2)$-surface, between any two points on the $(d-2)$-surface the line visits a sequence of points on the $(d-3)$-surface, and so on. Every time we find a point on the i-surface, we remember it, increment our position in dimension i by 1, and reset our coordinates back to 0 for all dimensions $j < i$. Hence, a vertex will be a tuple $(p_0, p_1, \ldots, p_d)$, where each p_i is either the symbol $-$, indicating that we have not yet encountered the i-surface, or the most recent point on the i-surface that we have visited.

The potential is likewise generalized so that the potential of a point p is proportional to $\sum_{i=1}^{d} k^i p_i$, where k is some constant that is larger than the grid size. Thus progress in dimension i dominates progress in dimension j whenever $j < i$, and the potential monotonically increase along the line. We are also able to deal with all possible violations.

Completing the reduction to UniqueEOPL. The final step of the reduction uses SinkOfVerifiableLine, a problem introduced by Bitansky et al [4]. SinkOfVerifiableLine is intuitively similar to UniqueForwardEOPL. It was shown by Hubáček and Yogev [36] that SinkOfVerifiableLine can be reduced to an EndOfMeteredLine instance with a unique line, and hence to UniqueEOPL. However, [36] only deals with the promise problem. Our contribution is to deal with violations. In doing so, we complete our chain of promise-preserving reductions from OPDC to UniqueEOPL. It is worth pointing out that this step of the reduction implies that the cryptographic hardness results shown by Bitansky et al. for SinkOfVerifiableLine [5] also apply to UniqueEOPL.

Hardness of OPDC. We show that OPDC is UniqueEOPL-hard by giving a polynomial-time promise-preserving reduction from UniqueEOPL to OPDC. Our reduction produces an OPDC instances where the set of points P is the Boolean hypercube $\{0,1\}^n$. In the case where the UniqueEOPL instance has no violations, meaning that it contains a single line, the reduction embeds this line into the hypercube. To do this, it splits the line in half. The second half is embedded into a particular sub-cube, while the first half is embedded into all other sub-cubes. This process is recursive, so each half of the line is again split in half, and further embedded into sub-cubes. The reduction ensures that the only fixpoint of the instance corresponds to the end of the line. If the UniqueEOPL instance does have violations, this embedding may fail, but we are then able to produce a violation for the original UniqueEOPL instance. We remark that this reduction makes significant progress towards showing hardness for Contraction and USO, since OPDC is a discrete variant of Contraction, and when the set of points is a hypercube, the problem is also very similar to USO. The key difference is that OPDC insists that only i-slices have a unique fixpoint, whereas Contraction and USO insist that *all* slices have unique fixpoints.

▶ **Theorem 5.** OPDC *is* UniqueEOPL*-complete under promise-preserving reductions, even when the set of points P is a hypercube.*

4 UniqueEOPL containment results

We show that USO, P-LCP, and a variant of Contraction all lie in UniqueEOPL. For each of these three problems, we provide a reduction to OPDC, shown to be in UniqueEOPL in the previous section.

A USO instance naturally gives rise to an OPDC instance where the underlying grid of points is actually a cube, and there is an easy reduction that shows the following.

▶ **Theorem 6.** *USO is in* UniqueEOPL *under promise-preserving reductions.*

This result substantially advances our knowledge about USO, since prior to this work, it was only known to lie in TFNP, and Kalai [39, Problem 6] had posed the challenge to place it in some non-trivial subclass of TFNP. We place it in *all* of the standard subclasses of TFNP[2].

We provide two reductions from P-LCP to UniqueEOPL. One is via the known reduction from P-LCP to USO [59]. The other uses Lemke's algorithm and produces a UEOPL instance with size linear in that of the P-LCP, which we require for our algorithmic result in Section 5. Lemke's algorithm is a PPAD-style path-following algorithm. The potential comes from a parameter used in Lemke's algorithm that changes monotonically on an P-matrix LCP. The complication is to deal with violations when the input matrix is not a P-matrix.

The reason for two reductions is that each produces different types of violation. We emphasize that all violations used are well-known and natural, and perhaps one can convert between them in polynomial time. Moreover, for the promise problem the choice of violations is irrelevant: each reduction independently shows that promise P-LCP lies in PromiseUEOPL.

▶ **Theorem 7.** *P-LCP* $\in$ UniqueEOPL *under promise-preserving reductions.*

For contraction, we study maps specified by *piecewise linear* functions. This differs from [14], where the map is given by an arbitrary arithmetic circuit. Although every contraction map has a unique fixpoint, for an arbitrary arithmetic circuit, the unique *exact* fixpoint may be irrational, and finding it is not known to be in FNP. Prior work instead asked for an *approximate* fixpoint [14]. However, given our interest in uniqueness of solutions we need to consider exact fixpoints, and thus study the problem with LinearFIXP arithmetic circuits [18], where multiplication of two variables is disallowed, and when the function is contracting, there is a unique rational fixpoint. This is still an interesting class of contraction maps, since it is powerful enough to represent the well-studied simple-stochastic games [18, 11]. We place this problem in UniqueEOPL via a promise-preserving reduction to OPDC. When the promise is not satisfied, the reduction either produces the standard violation, a pair of points at which the function is not contracting, or a different, more technical, violation.

OPDC was inspired by the continuous contraction problem, and our reduction from contraction is to OPDC. The most complicated part of the reduction is picking a suitable set of points for the OPDC instance that is small enough, but also is guaranteed to contain the unique fixed point of the contraction instance. To do this, we formulate the fixpoint problem for a LinearFIXP circuit as an LCP and reason about the bit-length of solutions to this LCP.

▶ **Theorem 8.** *Finding the fixpoint of a piecewise linear contraction map in the* ℓ_p *norm is in* UniqueEOPL *under promise-preserving reductions for any* $p \in \mathbb{N} \cup \{\infty\}$.

Finally, we note that our results imply that several other problems lie in UniqueEOPL. The simple-stochastic game (SSG) problem is known to reduce to piecewise-linear Contraction [18] and P-LCP [30]. Discounted games are known to reduce to SSGs [62], mean-payoff games to discounted games [62], and parity games to mean-payoff games [50]. So all these problems lie in UniqueEOPL too. [27] noted that ARRIVAL [17] lies in EOPL; since their EndOfPotentialLine instance contains only one line, ARRIVAL also lies in UniqueEOPL. However, none of these are promise-problems. Each can be formulated so as to *unconditionally* have a unique solution. Hence, they seem to be easier than other problems captured by UniqueEOPL.

[2] However, we do not place the problem in the recently defined class PWPP [37]

▶ **Theorem 9.** *The following problems are in* UniqueEOPL*: Solving a parity game; mean-payoff game; discounted game; simple-stochastic game; the ARRIVAL problem.*

5 New algorithms

The insights provided by our containment results give two algorithmic results. Firstly, we obtain simple polynomial-time algorithms for finding the fixpoint of a contraction map in fixed dimension for any ℓ_p norm. This result was already known via a reduction to the problem of finding a Tarski fixpoint [51], but our algorithm utilises the structural properties of contraction that arise from our reduction to OPDC, and is arguably simpler.

Secondly, as noted in [27], one of our reductions for P-LCP allows a technique of Aldous [2] to be applied, giving the fastest known randomized algorithm for P-LCP.

6 Conjectures and conclusions

▶ **Conjecture 1.** *USO is hard for* UniqueEOPL.

Among our three motivating problems, USO seems the most likely to be UniqueEOPL-complete. Our hardness proof for OPDC already goes some way towards proving this, since it applies even on a hypercube. Going further, could we even show the stronger result of hardness for P-LCP, which would imply hardness of USO? The complexity of these two problems has been open for decades.

▶ **Conjecture 2.** *Piecewise-Linear Contraction in an ℓ_p norm is hard for* UniqueEOPL.

For this result, in addition to the i-slice vs. all slice issue, we would also need to convert the discrete OPDC problem to the continuous contraction problem. Converting discrete problems to continuous fixpoint problems has been well-studied in the context of PPAD-hardness reductions [13, 45], but here we must additionally maintain the contraction property.

Aside from hardness, we also think that the relationship between Contraction and USO should be explored further, since OPDC exposes significant, previously unrecognised, similarities between the two problems.

▶ **Conjecture 3.** UniqueEOPL $\subset$ EOPL $=$ CLS.

The question of EOPL vs CLS is unresolved, and we actually think it could go either way. One could show that EOPL $=$ CLS by placing either of the two known CLS-complete Contraction variants into EOPL [15, 20]. If the two classes are actually distinct, then it is interesting to ask which of the problems in CLS are also in EOPL.

On the other hand, we believe that UniqueEOPL is a strict subset of EOPL. The evidence for this is that the extra violation in UniqueEOPL that does not appear in EndOfPotentialLine changes the problem significantly. It introduces many new solutions whenever there are multiple lines in the instance, and so it is unlikely, in our view, that one could reduce EndOfPotentialLine to UniqueEOPL. We also believe it very unlikely that other problems in CLS, such as the KKT problem of finding an approximate stationary point of a multivariate polynomial, are in UniqueEOPL. Of course, there is little hope to unconditionally prove that UniqueEOPL $\subset$ EOPL, but we can ask for further evidence, such as oracle separations, to support the idea.

References

1 Ilan Adler and Sushil Verma. The Linear Complementarity Problem, Lemke Algorithm, Perturbation, and the Complexity Class PPAD. Technical report, Manuscript, Depatrment of IEOR, University of California, Berkeley, CA 94720, 2011.

2 David Aldous. Minimization algorithms and random walk on the d-cube. *The Annals of Probability*, pages 403–413, 1983.

3 Stefan Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae*, 3(1):133–181, 1922.

4 Nir Bitansky, Omer Paneth, and Alon Rosen. On the Cryptographic Hardness of Finding a Nash Equilibrium. In *Proc. of FOCS*, pages 1480–1498, 2015.

5 Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a Nash equilibrium. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 1480–1498. IEEE, 2015.

6 Ch. Boonyasiriwat, Kris Sikorski, and Ch. Xiong. A note on two fixed point problems. *J. Complexity*, 23(4-6):952–961, 2007.

7 Xi Chen and Xiaotie Deng. On algorithms for discrete and approximate Brouwer fixed points. In *Proc. of STOC*, pages 323–330, 2005.

8 Xi Chen and Xiaotie Deng. Matching algorithmic bounds for finding a Brouwer fixed point. *J. ACM*, 55(3):13:1–13:26, 2008.

9 Xi Chen and Xiaotie Deng. On the complexity of 2D discrete fixed point problem. *Theor. Comput. Sci.*, 410(44):4448–4456, 2009.

10 Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibria. *J. ACM*, 56(3):14, 2009.

11 Anne Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, 1992.

12 Richard W Cottle, Jong-Shi Pang, and Richard E Stone. *The Linear Complementarity Problem*. SIAM, 2009.

13 Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009.

14 Constantinos Daskalakis and Christos Papadimitriou. Continuous Local Search. In *Proc. of SODA*, pages 790–804, 2011.

15 Constantinos Daskalakis, Christos Tzamos, and Manolis Zampetakis. A Converse to Banach's Fixed Point Theorem and its CLS Completeness. In *Proc. of STOC*, 2018.

16 Xiaotie Deng, Qi Qi, Amin Saberi, and Jie Zhang. Discrete Fixed Points: Models, Complexities, and Applications. *Math. Oper. Res.*, 36(4):636–652, 2011.

17 Jérôme Dohrau, Bernd Gärtner, Manuel Kohler, Jiří Matoušek, and Emo Welzl. ARRIVAL: a zero-player graph game in $\mathsf{NP} \cap \mathsf{coNP}$. In *A journey through discrete mathematics*, pages 367–374. Springer, Cham, 2017.

18 Kousha Etessami and Mihalis Yannakakis. On the Complexity of Nash Equilibria and Other Fixed Points. *SIAM J. Comput.*, 39(6):2531–2597, 2010.

19 Alex Fabrikant, Christos Papadimitriou, and Kunal Talwar. The complexity of pure Nash equilibria. In *Proc. of STOC*, pages 604–612. ACM, 2004.

20 John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. CLS: new problems and completeness. *CoRR*, abs/1702.06017, 2017. `arXiv:1702.06017`.

21 John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. End of Potential Line. *CoRR*, abs/1804.03450, 2018. `arXiv:1804.03450`.

22 John Fearnley and Rahul Savani. The complexity of all-switches strategy improvement. In *Proc. of SODA*, pages 130–139, 2016.

23 Oliver Friedmann, Thomas Dueholm Hansen, and Uri Zwick. A subexponential lower bound for the Random Facet algorithm for Parity Games. In *Proc. of SODA*, pages 202–216, 2011.

24 Martin Gairing and Rahul Savani. Computing Stable Outcomes in Hedonic Games. In *Proc. of SAGT*, pages 174–185, 2010.

25 Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a Nash equilibrium. In *Annual Cryptology Conference*, pages 579–604. Springer, 2016.

26 Bernd Gärtner. The Random-Facet simplex algorithm on combinatorial cubes. *Random Struct. Algorithms*, 20(3):353–381, 2002.

27 Bernd Gärtner, Thomas Dueholm Hansen, Pavel Hubáček, Karel Král, Hagar Mosaad, and Veronika Slívová. ARRIVAL: next stop in CLS. In *Proc. of ICALP*, pages 60:1–60:13, 2018.

28 Bernd Gärtner and Ingo Schurr. Linear programming and unique sink orientations. In *Proc. of SODA*, pages 749–757, 2006. URL: `http://dl.acm.org/citation.cfm?id=1109557.1109639`.

29 Bernd Gärtner and Antonis Thomas. The Complexity of Recognizing Unique Sink Orientations. In *Proc. of STACS*, pages 341–353, 2015.

30 Thomas Dueholm Hansen and Rasmus Ibsen-Jensen. The complexity of interior point methods for solving discounted turn-based stochastic games. In *Conference on Computability in Europe*, pages 252–262, 2013.

31 Thomas Dueholm Hansen, Mike Paterson, and Uri Zwick. Improved upper bounds for Random-Edge and Random-Jump on abstract cubes. In *Proc. of SODA*, pages 874–881, 2014.

32 Michael D. Hirsch, Christos H. Papadimitriou, and Stephen A. Vavasis. Exponential lower bounds for finding Brouwer fix points. *J. Complexity*, 5(4):379–416, 1989.

33 Kathy Williamson Hoke. Completely unimodal numberings of a simple polytope. *Discrete Applied Mathematics*, 20(1):69–81, 1988.

34 Z. Huang, Leonid Khachiyan, and Krzysztof Sikorski. Approximating fixed points of weakly contracting mappings. *J. Complexity*, 15:200–213, 1999.

35 Z. Huang, Leonid G. Khachiyan, and Christopher (Krzysztof) Sikorski. Approximating Fixed Points of Weakly Contracting Mappings. *J. Complexity*, 15(2):200–213, 1999.

36 Pavel Hubáček and Eylon Yogev. Hardness of continuous local search: Query complexity and cryptographic lower bounds. In *Proc. of SODA*, pages 1352–1371, 2017.

37 Emil Jeřábek. Integer factoring and modular square roots. *Journal of Computer and System Sciences*, 82(2):380–394, 2016.

38 David S Johnson, Christos H Papadimitriou, and Mihalis Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100, 1988.

39 Gil Kalai. Three Puzzles on Mathematics, Computation, and Games. *CoRR*, abs/1801.02602, 2018. `arXiv:1801.02602`.

40 Masakazu Kojima, Nimrod Megiddo, Toshihito Noma, and Akiko Yoshise. *A unified approach to interior point algorithms for linear complementarity problems*, volume 538. Springer Science & Business Media, 1991.

41 Masakazu Kojima, Nimrod Megiddo, and Yinyu Ye. An interior point potential reduction algorithm for the linear complementarity problem. *Mathematical Programming*, 54(1-3):267–279, 1992.

42 Carlton E Lemke. Bimatrix equilibrium points and mathematical programming. *Management science*, 11(7):681–689, 1965.

43 Jiří Matoušek and Tibor Szabó. Random Edge Can Be Exponential on Abstract Cubes. In *Proc. of FOCS*, pages 92–100, 2004.

44 Nimrod Megiddo and Christos H Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, 1991.

45 Ruta Mehta. Constant rank bimatrix games are PPAD-hard. In *Proc. of STOC*, pages 545–554, 2014.

46 Walter D Morris Jr. Randomized pivot algorithms for P-matrix linear complementarity problems. *Mathematical programming*, 92(2):285–296, 2002.

47 Katta G Murty. Computational complexity of complementary pivot methods. In *Complementarity and fixed point problems*, pages 61–73. Springer, 1978.

48 A. Nemirovsky and D. B. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley, New York, 1983.

49 Christos H Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994.

50 Anuj Puri. *Theory of hybrid systems and discrete event systems*. PhD thesis, University of California at Berkeley, 1996.

51 Qi Qi. *Computational efficiency in internet economics and resource allocation*. PhD thesis, Stanford University, 2012.

52 Aviad Rubinstein. Settling the Complexity of Computing Approximate Two-Player Nash Equilibria. In *Proc. of FOCS*, pages 258–265, 2016.

53 Alejandro A Schäffer and Mihalis Yannakakis. Simple local search problems that are hard to solve. *SIAM journal on Computing*, 20(1):56–87, 1991.

54 Ingo Schurr and Tibor Szabó. Finding the Sink Takes Some Time: An Almost Quadratic Lower Bound for Finding the Sink of Unique Sink Oriented Cubes. *Discrete & Computational Geometry*, 31(4):627–642, 2004.

55 Ingo Schurr and Tibor Szabó. Jumping Doesn't Help in Abstract Cubes. In *Proc. of IPCO*, pages 225–235, 2005.

56 Spencer Shellman and Krzysztof Sikorski. A recursive algorithm for the infinity-norm fixed point problem. *Journal of Complexity*, 19(6):799–834, 2003.

57 Krzysztof Sikorski. *Optimal solution of Nonlinear Equations*. Oxford Press, New York, 200.

58 Krzysztof Sikorski. Computational complexity of fixed points. *Journal of Fixed Point Theory and Applications*, 6(2):249–283, 2009.

59 Alan Stickney and Layne Watson. Digraph models of Bard-type algorithms for the linear complementarity problem. *Mathematics of Operations Research*, 3(4):322–333, 1978.

60 Tibor Szabó and Emo Welzl. Unique Sink Orientations of Cubes. In *Proc. of FOCS*, pages 547–555, 2001.

61 Antonis Thomas. Exponential Lower Bounds for History-Based Simplex Pivot Rules on Abstract Cubes. In *Proc. of ESA*, pages 69:1–69:14, 2017.

62 Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1-2):343–359, 1996.

Dichotomy for Symmetric Boolean PCSPs

Miron Ficak
Theoretical Computer Science Department, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland
miron.ficak@student.uj.edu.pl

Marcin Kozik
Theoretical Computer Science Department, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland
marcin.kozik@uj.edu.pl

Miroslav Olšák
Department of Algebra, Charles University, Prague, Czech Republic
mirek@olsak.net

Szymon Stankiewicz
Theoretical Computer Science Department, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland
szymon.stankiewicz@student.uj.edu.pl

Abstract

In one of the most actively studied version of Constraint Satisfaction Problem, a CSP is defined by a relational structure called a template. In the decision version of the problem the goal is to determine whether a structure given on input admits a homomorphism into this template. Two recent independent results of Bulatov [FOCS'17] and Zhuk [FOCS'17] state that each finite template defines CSP which is tractable or NP-complete.

In a recent paper Brakensiek and Guruswami [SODA'18] proposed an extension of the CSP framework. This extension, called Promise Constraint Satisfaction Problem, includes many naturally occurring computational questions, e.g. approximate coloring, that cannot be cast as CSPs. A PCSP is a combination of two CSPs defined by two similar templates; the computational question is to distinguish a YES instance of the first one from a NO instance of the second.

The computational complexity of many PCSPs remains unknown. Even the case of Boolean templates (solved for CSP by Schaefer [STOC'78]) remains wide open. The main result of Brakensiek and Guruswami [SODA'18] shows that Boolean PCSPs exhibit a dichotomy (PTIME vs. NPC) when "all the clauses are symmetric and allow for negation of variables". In this paper we remove the "allow for negation of variables" assumption from the theorem. The "symmetric" assumption means that changing the order of variables in a constraint does not change its satisfiability. The "negation of variables" means that both of the templates share a relation which can be used to effectively negate Boolean variables.

The main result of this paper establishes dichotomy for all the symmetric boolean templates. The tractability case of our theorem and the theorem of Brakensiek and Guruswami are almost identical. The main difference, and the main contribution of this work, is the new reason for hardness and the reasoning proving the split.

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

Keywords and phrases promise constraint satisfaction problem, PCSP, algebraic approach

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.57

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is avaialable at `https://arxiv.org/abs/1904.12424`, [11] .

Funding Research was partially supported by National Science Centre, Poland grant no. 2014/2013/B/ST6/01812.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 57; pp. 57:1–57:12

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

1 Introduction

Constraint Satisfaction Problems have been studied in computer science in many forms. In the general approach an instance of the CSP consists of variables and constraints. In the decision version of the problem the objective is to verify whether there exists an evaluation of variables that meets all the constraints.

One particular type of CSPs received a lot of attention in the past years. In this approach constraints are relations taken from a fixed, finite relational structure called a template. The interest in this particular version was driven by a conjecture of Feder and Vardi [10] postulating that each finite template defines a CSP which is tractable or NP-complete.

A great variety of decision problems independently studied by computer scientists can be cast as CSPs. To name a few: 3-SAT, k-colorability, (generalized) unreachability in directed graphs or solving systems of linear equation over a finite field, are all CSPs defined by finite templates. The class of all the computational problems falling into the scope of the conjecture is very big and its verification was a gradual and lengthy process. Nevertheless, from the start, the claim was supported by strong evidence. In this context the classical result of Schaefer [15] showing that the dichotomy holds for templates over Boolean domain, is perhaps the most important.

The dichotomy for all the finite templates was recently confirmed by two, independent results of Bulatov [6] and Zhuk [16]. Both of them use the algebraic approach [13, 7], where the complexity of a template is studied via compatible operations called polymorphisms. The algebraic approach proved very successful not only in the decision version of the CSP: a number of important results in optimization [14], approximation [2] etc. of the CSP is based on some versions of polymorphisms.

A positive resolution of the dichotomy conjecture motivates the following question: is the class of CSPs unique, or maybe a part of a larger, natural class which also exhibits a dichotomy? Note that such a class should be amenable to some sort of the algebraic approach, as no other tools offer comparable power even in the case of the CSP. In the recent paper [5] Brakensiek and Guruswami proposed a candidate for such a class.

The Constraint Satisfaction Problem defined by a fixed language can be cast as a problem of finding homomorphism from a relational structure given on input to a fixed template. The class proposed by Brakensiek and Guruswami as an extension of CSP is called Promise Constraint Satisfaction Problems. A PCSP is based on two CSPs with similar templates and the question is to distinguish YES instances of the first CSP from NO instances of the second.

To provide a few examples: the CSP defined by an undirected clique (without loops) of size k as a template is just k-colorability. Defining PCSP by two cliques, say of sizes k and l satisfying $k < l$, we get the following problem: distinguish between the graphs with chromatic number $\leq k$ and those with chromatic number $> l$. These problems are studied independently [9, 12, 3, 8], but the characterization of complexities for all pairs (k, l) is either incomplete or done under additional assumptions.

Another example is a Boolean PCSP. A single ternary relation $\{(0,0,1),(0,1,0),(1,0,0)\}$ defines a CSP which is known as Monotone-1-in-3-SAT, and similarly the relation $\{0,1\}^3 \setminus \{(0,0,0),(1,1,1)\}$ gives rise to the CSP known as Monotone-NAE-SAT. Thus the question of distinguishing between instances which are satisfiable as Monotone-1-in-3-SAT instances and not satisfiable as Monotone-NAE-SAT instances is a PCSP. Surprisingly this problem is tractable even allowing for the negation of variables [1, 5].

Further examples of problems expressible as PCSPs can be found in [5]. Promise Constraint Satisfaction Problems generalize CSPs and include many additional, natural problems. The algebraic approach to the CSP can be adjusted to work in the case of

the PCSP. The first Galois correspondence between PCSPs and the polymorphisms was introduced in [5], and the more abstract algebraic approach was proposed in [8]. Despite all the interest, PCSPs lack a classification result that would play the role of Schaefer's theorem. This motivates a more systematic study of Boolean PCSPs.

The main result of Brakensiek and Guruswami, Theorem 2.1 in [5], establishes dichotomy for a certain class of Boolean PCSPs. A PCSP template falls into this class if all the relations in the templates are symmetric (i.e. invariant under permutations, or equivalently, determined by Hamming weights of the tuples) and additionally the template contains a relation which can be used to negate Boolean variables in both CSP templates. As the additional relation is binary and symmetric, the result concerns all the symmetric templates containing this particular relation. In this paper we remove the additional assumption and show that all symmetric Boolean templates exhibit a dichotomy.

Let us further compare the results. The algorithms required for the original and extended result are exactly the same: Gaussian elimination or linear programming relaxation depending on the polymorphisms of the template. The list of polymorphisms implying tractability differs slightly as we need to allow additional threshold functions (Boolean functions returning 0 if and only if the number of 1's is below a threshold). Unfortunately the condition which guarantees hardness in the original paper fails when the negating relation is absent. The new hardness condition and a more involved analysis of the minion of polymorphisms are required in the proof and constitute the main contribution of this paper.

The publication is organized as follows. The next section contains basic definitions commonly used in context of an algebraic approach to the CSP or the PCSP. Section 3 contains a list of polymorphisms that guarantee tractability, statement of the main theorem and a proof of the tractability case. In section 4 we introduce notation and nomenclature. Section 5 contains the algebraic condition implying hardness of PCSP and a proof of this implication. The main part of the reasoning behind the result is focused on showing that lack of polymorphisms from the tractability list implies, in our case, the condition for hardness. Section 6 contains an overview of this proof and a complete reasoning can be found in the full version of the paper.

2 Basic definitions

This section contains basic definitions and notions relevant to CSP and PCSP. A relation $R \subseteq A^n$ is an *n-ary relation* and the set A is its *universe*. A relation is *symmetric*, if for every permutation σ of $[n]$ (where $[n]$ is defined to be $\{1, \dots, n\}$) if $(a_1, \dots, a_n) \in R$ then also $(a_{\sigma(1)}, \dots, a_{\sigma(n)}) \in R$. A relation $R^m \subseteq (A^m)^n$ is a *Cartesian power* of $R \subseteq A^n$ if $(a^1, \dots, a^n) \in R^m$ if and only if $(a^1_i, \dots, a^n_i) \in R$ for every i (i.e. R^m is defined from R coordinate-wise).

A *relational structure* $\mathbf{A}$ is a tuple $(A; R_1, \dots R_n)$ where each R_i is a relation on A, and we call a relational structure *symmetric* if all its relations are. Two relational structures are *similar* if they have the same sequence of arities of their relations. E.g. a relational structure $(A; R_1, \dots R_n)$ and it's m-th power $(A^m; (R_1)^m, \dots (R_n)^m)$ are similar. For two similar structures say $\mathbf{A} = (A; R_1, \dots, R_n)$ and $\mathbf{B} = (B; S_1, \dots, S_n)$ a function $h : A \to B$ is *a homomorphism* if for every i and every tuple $(a_1, \dots, a_m) \in R_i$ the tuple $(h(a_1), \dots, h(a_m)) \in S_i$.

The Constraint Satisfaction Problem defined by a relational structure $\mathbf{B}$ (denoted by CSP($\mathbf{B}$)) is the following decision problem:

Input:	a relational structure $\mathbf{A}$ similar to $\mathbf{B}$
Question:	does there exists a homomorphism from $\mathbf{A}$ to $\mathbf{B}$?

The relational structure $\mathbf{B}$ is called a *template* of such a problem.

The Promise Constraint Satisfaction Problem is a promise problem defined by a pair of similar relational structures $(\mathbf{B}, \mathbf{C})$ such that there exists a homomorphism from $\mathbf{B}$ to $\mathbf{C}$. The PCSP$(\mathbf{B}, \mathbf{C})$ is:

Input:	a relational structure $\mathbf{A}$ similar to $\mathbf{B}$ and $\mathbf{C}$
Output YES:	if there exists a homomorphism from $\mathbf{A}$ to $\mathbf{B}$
Output NO:	if there is no homomorphism from $\mathbf{A}$ to $\mathbf{C}$.

Just like in the case of the CSP, the pair $(\mathbf{B}, \mathbf{C})$ is called a template. Clearly PCSP$(\mathbf{B}, \mathbf{B})$ is CSP$(\mathbf{B})$ and therefore the PCSP generalizes the CSP.

Both problems exhibit a Galois correspondence i.e. instead of studying the structure of the template one can choose to analyze the structure of template's polymorphisms [13, 7, 5, 8]. A *polymorphism* of a relational structure $\mathbf{B}$ is a homomorphism from a finite Cartesian power of $\mathbf{B}$ to $\mathbf{B}$. Similarly a polymorphism of a PCSP template $(\mathbf{B}, \mathbf{C})$ is a homomorphism from a finite Cartesian power of $\mathbf{B}$ to $\mathbf{C}$. We denote the set of all polymorphisms of $\mathbf{B}$ by Pol$(\mathbf{B})$, and the set of all polymorphisms of $(\mathbf{B}, \mathbf{C})$ by Pol$(\mathbf{B}, \mathbf{C})$.

For each relational structure $\mathbf{B}$ the set Pol$(\mathbf{B})$ is *clone* i.e. it contains projections and is closed under composition. Similarly for a pair $(\mathbf{B}, \mathbf{C})$ the set Pol$(\mathbf{B}, \mathbf{C})$ is a minion. *A minion* is a set of functions closed under taking *minors* i.e. creating functions by identifying variables, permuting variables and introducing dummy variables. If $f(x_1, \dots, x_n)$ is a function and $f'(x) = f(x, \dots, x)$ then $f'(x)$ is the unary minor of $f(x_1, \dots, x_n)$ and $f''(x, y) = f(x, y, \dots, y)$ is a binary minor of $f(x_1, \dots, x_n)$.

In some cases, instead of considering a PCSP template $((A; R_1, \dots, R_n), (B; S_1, \dots, S_n))$ we work with an equivalent concept of *a language* i.e. a sequence of pairs $[R_1, S_1], \dots, [R_n, S_n]$. We say that a pair $[S, T]$ is *compatible* with a minion $\mathcal{M}$, if every member of $\mathcal{M}$ maps an appropriate power of S to T (the exponent of the power is the arity of the operation).

A primitive positive formula (pp-formula) is a formula constructed using atomic formulas, conjunction and existential quantification. Such formulas play a special role in CSP and PCSP: if a relation R has a primitive positive definition in $\mathbf{B}$ then R is compatible with Pol$(\mathbf{B})$ and adding R to $\mathbf{B}$ does not change the computational complexity of the CSP$(\mathbf{B})$. Similarly, if a pair $[R, S]$ has a pp-definition in the language of $(\mathbf{A}, \mathbf{B})$ (pp-formula in $[R_i, S_i]$ defines such $[R, S]$ in the natural way) then $[R, S]$ is compatible with Pol$(\mathbf{A}, \mathbf{B})$ and adding it to the language/template does not change the complexity [5]. One more construction, called *strict relaxation*, plays an important role in the theory of PCSP: if $[R_i, S_i]$ is an element of the language $(\mathbf{B}, \mathbf{C})$ and $R \subseteq R_i$ while $S_i \subseteq S$ then $[R, S]$ is compatible with Pol$(\mathbf{A}, \mathbf{B})$ and adding it to the language/template does not change the complexity.

3 Main theorem and tractability

Focusing on the Boolean domain we present the main theorem of the paper and prove that the tractable cases are indeed solvable in P. In this part of the proof our paper does not deviate much from [5]; the polymorphisms which imply tractability are almost the same with an exception of the threshold case.

- A n-ary function is *a max* (*a min*) if it returns maximum (resp. minimum) of its arguments (in the natural order on $\{0, 1\}$).

- A function $f(x_1, \ldots, x_n)$ is *an alternating threshold* if $n = 2k + 1$ and

$$f(x_1, \ldots, x_k, x_{k+1}, \ldots, x_n) = \begin{cases} 0 & \text{if } \sum_{i=1}^{k} x_i \geq \sum_{i=k+1}^{n} x_i, \\ 1 & \text{if } \sum_{i=1}^{k} x_i < \sum_{i=k+1}^{n} x_i, \end{cases}$$

- A function $f(x_1, \ldots, x_n)$ is *an xor* if n is odd and $f(x_1, \ldots, x_n) = x_1 + \cdots + x_n \bmod 2$.
- A function $f(x_1, \ldots, x_n)$ is *a q-threshold* (where q is a rational between 0 and 1) if

$$f(x_1, \ldots, x_n) = \begin{cases} 0 & \text{if } \sum_{i=1}^{n} x_i < nq, \\ 1 & \text{if } \sum_{i=1}^{n} x_i > nq, \end{cases}$$

and nq is not an integer. Note that all the evaluations of the $f(x_1, \ldots, x_n)$ are determined. We denote the set of all max functions by MAX, all the min functions by MIN, all alternating thresholds by AT all xor by XOR and all q-thresholds by THR_q. For a set of functions F by $\overline{F}$ we denote $\{1 - f(x_1, \ldots, x_n) : f(x_1, \ldots, x_n) \in F\}$. We are ready to state the main result of the paper.

▶ **Theorem 1.** *Let $(\mathbf{A}, \mathbf{B})$ be a symmetric, Boolean PCSP language. If $Pol(\mathbf{A}, \mathbf{B})$ contains a constant or includes at least one of the sets MAX, MIN, AT, XOR , THR_q (for some q), $\overline{MAX}$, $\overline{MIN}$, $\overline{AT}$, $\overline{XOR}$ or $\overline{THR_q}$ (for some q) then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is tractable. Otherwise it is NP-complete.*

Comparing the statement of Theorem 1 and Theorem 2.1 of [5] we find two differences: the earlier paper additionally assumes that negated variables can appear in instances and it allows the authors to substitute "THR_q for some q" with $\mathrm{THR}_{1/2}$ in the list of conditions that force tractability.

In the remaining part of this section we will show the tractability case of Theorem 1. The reasoning differs very little from the one found in [5] and therefore we cover it quickly: If $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ contains a constant function $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is clearly tractable; if it includes MAX, MIN and XOR tractability follows from Lemma 3.1 of [5]. If $\mathrm{AT} \subseteq \mathrm{Pol}(\mathbf{A}, \mathbf{B})$ then Claim 2 of Section 3.2 [5] implies tractability. Finally the case of THR_q is a minor generalization of the argument in Claim 1 of Section 3.2 in the same paper, or a special case of Theorem 5.2 in [4].

The remaining cases reduce, just like in [5], to the ones from the previous paragraph: let relational structure $\mathbf{B}'$ be obtained from $\mathbf{B}$ by exchanging the roles of 0 and 1 (that is, in every relation in $\mathbf{B}$, in every tuple of this relation and at every position in this tuple we change x to $1 - x$). The YES instances of $\mathrm{PCSP}(\mathbf{A}, \mathbf{B}')$ and $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ are trivially the same and so are the NO instances. If $\overline{\mathrm{MIN}} \subseteq \mathrm{Pol}(\mathbf{A}, \mathbf{B})$ then $\mathrm{MIN} \subseteq \mathrm{Pol}(\mathbf{A}, \mathbf{B}')$ and, by the cases already established, $\mathrm{PCSP}(\mathbf{A}, \mathbf{B}')$ is tractable. Clearly $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is tractable as well and all the remaining tractable cases can be dealt with the same way.

4 The notation for symmetric Boolean PCSPs

In order to show NP-hardness in the remaining case of Theorem 1, we require a few definitions which allow us to work with symmetric Boolean relations and Boolean function concisely.

Every symmetric relation $R \subseteq \{0, 1\}^m$ is uniquely determined by the set $I \subseteq \{0, \ldots, m\}$ consisting of the Hamming weights of its elements. This fact allows us to use R and I interchangeably. Let $(\mathbf{B}, \mathbf{C})$ be a symmetric, Boolean PCSP template with language $[R_1, S_1], \ldots, [R_n, S_n]$ where the arities of the relations are $a_1, \ldots, a_n$. We will denote such a language by $[\, I_1 \mid J_1 \,]_{a_1}, \ldots, [\, I_n \mid J_n \,]_{a_n}$ where I_i (J_i) is a set of Hamming weights of elements

of R_i (S_i respectively). We will often use a flattened form of this notation: we will denote $[\,\{1\} \mid \{1,2\}\,]_2$ by $[\,1 \mid 1,2\,]_2$ and so on as well as $[n] = \{1,\dots,n\}$.

Focusing on compatibility; an operation $f(x_1,\dots,x_n)$ is compatible with $[\,0 \mid 0\,]_1$ if and only if $f(0,\dots,0) = 0$ and compatible with $[\,1 \mid 1\,]_1$ if and only if $f(1,\dots,1) = 1$. The pair $[\,1 \mid 1\,]_2$ defines negation in $\mathbf{A}$ and $\mathbf{B}$ and therefore the main result of [5] is a special case of Theorem 1; the additional assumption states that $[\,1 \mid 1\,]_2$ is in the language of PCSP.

We proceed to illustrate a number of pp-definitions and strict relaxations that appear repeatedly in the proofs. Using $[\,I \mid J\,]_n$ and $[\,0 \mid 0\,]_1$ one can define $[\,I \setminus \{n\} \mid J \setminus \{n\}\,]_{n-1}$ using the following pp-formula:

$$\exists x_1\ [\,0 \mid 0\,]_1(x_1) \wedge [\,I \mid J\,]_n(x_1,\dots,x_n).$$

Similarly

$$\exists x_1\ [\,1 \mid 1\,]_1(x_1) \wedge [\,I \mid J\,]_n(x_1,\dots,x_n)$$

defines $[\,I' \mid J'\,]_{n-1}$ where $I' = \{i-1 : i \in I \text{ and } i \neq 0\}$ and $J' = \{j-1 : j \in J \text{ and } j \neq 0\}$. The strict relaxations we use are straightforward: take $[\,I \mid J\,]_n$ with $i \in I$ while $j \notin J$ then, for example, $[\,i \mid \{0,\dots,n\} \setminus \{j\}\,]_n$ is a strict relaxation of $[\,I \mid J\,]_n$.

In the proof of tractability for $(\mathbf{B},\mathbf{C})$ (at the end of Section 3) we swapped the role of 0 and 1 in $\mathbf{C}$. In the new notation we change $[\,I_1 \mid J_1\,]_{a_1},\dots,[\,I_n \mid J_n\,]_{a_n}$ to $[\,I_1 \mid J_1'\,]_{a_1},\dots,[\,I_n \mid J_n'\,]_{a_n}$ where $J_k' = \{a_k - j : j \in J_k\}$. In some of the proofs we reuse this construction, although we usually swap for both $\mathbf{B}$ and $\mathbf{C}$ at the same time.

We define notation for Boolean functions next. A Boolean function $f(x_1,\dots,x_n)$ is *idempotent* if $f(0,\dots,0) = 0$ and $f(1,\dots,1) = 1$. By the discussion above a minion is idempotent (i.e. contains idempotent functions only) if it is compatible with $[\,0 \mid 0\,]_1$ and $[\,1 \mid 1\,]_1$. Moreover the idempotent part of $\mathrm{Pol}(\mathbf{B},\mathbf{C})$ can be obtained by adding these pairs to the language.

For a Boolean function $f(x_1,\dots,x_n)$ and a set $U \subseteq [n]$ the value $f(U)$ is defined as $f(x_1,\dots,x_n)$ where $\{i : x_i = 1\} = U$. When n is clear from the context we can write $\overline{U}$ instead of $[n] \setminus U$. Let $f(x_1,\dots,x_n)$ be a Boolean function $U \subseteq [n]$ then U is

- a 1-SET if $f(U) = 1$,
- a 0-SET if $f(\overline{U}) = 0$,
- a 1-FIXING-SET (0-FIXING-SET) if every $V \supseteq U$ is a 1-SET (resp. 0-SET).

Moreover we say that a minion has *small fixing sets*, if there exists a constant N such that every function from the minion has a 1-FIXING-SET smaller than N, or every function from the minion has a 0-FIXING-SET smaller than N. Finally we say that a minion has *bounded antichains*, if there exist a constant M such that no function in the minion has M pairwise disjoint 1-SETs, and no function in the minion has M pairwise disjoint 0-SETs.

5 The hardness proof

In order to satisfy the assumptions of Lemma 5.1, we need some structural properties of the minion $\mathrm{Pol}(\mathbf{A},\mathbf{B})$. The following theorem collects these properties and is a cornerstone of our classification.

▶ **Theorem 2.** *Let $\mathbf{A},\mathbf{B}$ be a symmetric PCSP language such that $Pol(\mathbf{A},\mathbf{B})$ is idempotent. If $Pol(\mathbf{A},\mathbf{B})$ does not include MAX, MIN, AT, XOR and THR_q (for any q), then $Pol(\mathbf{A},\mathbf{B})$ has small fixing sets and bounded antichains.*

The Brakensiek and Guruswami [5] version of Theorem 2 requires that $(\mathbf{A}, \mathbf{B})$ contains $[\,1 \mid 1\,]_2$ and concludes that there exists a constant M such that every member of $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ has a set of size at most M which is a 1-FIXING-SET and a 0-FIXING-SET at the same time. The following example illustrates that their condition fails in our case.

▶ **Example 3.** Consider PCSP defined by a language consisting of $[\,0 \mid 0\,]_1$, $[\,1 \mid 1\,]_1$, $[\,1 \mid 1,2\,]_3$ and $[\,1 \mid 1,2\,]_4$. It is easy to verify that it falls into the hardness case of Theorem 1. On the other hand for each odd n the function $f(x_1, \ldots, x_n)$ defined as maximum of x_1 and n-ary element of $\mathrm{THR}_{1/2}$ is compatible with all the relational pairs. These functions have no uniform bound on the size of minimal 0-FIXING-SETs.

In the reminder of this section we use Theorem 2 to finish the proof of Theorem 1. We begin by introducing the machinery developed in [8] (a direct proof is possible, but involves a bit more technical considerations). The paper [8] defines *minor identity* as a formal expression of the form

$$f(x_1, \ldots, x_n) \approx g(x_{\pi(1)}, \ldots, x_{\pi(m)})$$

where f and g are function symbols (of arity n and m, respectively), $x_1, \ldots, x_n$ are variables, and $\pi : [m] \to [n]$. A minor identity is satisfied in a minion $\mathcal{M}$ (of functions from A to B) if there exists an interpretation of the function symbols f and g in $\mathcal{M}$, say ζ, satisfying

$$\zeta(f)(a_1, \ldots, a_n) = \zeta(g)(a_{\pi(1)}, ..., a_{\pi(m)})$$

for all $a_1, \ldots, a_n \in A$.

A bipartite minor condition is a finite set of minor identities in which function symbols used on the right- and left-hand sides are disjoint. A minor condition is *satisfied in a minion*, if there exists an interpretation simultaneously satisfying all the identities. A minor condition is *trivial* if it is satisfied in every minion, in particular, in the minion consisting of all projections on a set A that contains at least two elements. Finally, still following [8], a bipartite minor condition Σ is ε-robust (for some $\varepsilon > 0$) if no ε-fraction of identities from Σ is trivial.

▶ **Lemma 5.1** (Corollary 5.8 from [8]). *If there exists an $\varepsilon > 0$ such that $Pol(\mathbf{A}, \mathbf{B})$ does not satisfy any ε-robust bipartite minor condition, then $PCSP(\mathbf{A}, \mathbf{B})$ is NP-hard.*

In order to apply Lemma 5.1 to $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ we need to ensure that $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ does not satisfy any ε-robust bipartite minor condition. Our first step is to prove it in the idempotent case.

▶ **Proposition 5.2.** *Let $\mathcal{M}$ be an idempotent minion with small fixing sets, and bounded antichains. Then $\mathcal{M}$ does not satisfy any ε-robust bipartite minor condition.*

Proof. The proof follows the same pattern as the proofs of Propositions 5.10 and 5.12 in [8] so we will use the notation from those Propositions in this proof. All we need to do is to find $\varepsilon > 0$ and a mapping assigning to each member of $\mathcal{M}$ a probability distribution on its variables. The probability distribution needs to satisfy the following condition: if $f, g \in \mathcal{M}$ and $f(x_1, \ldots, x_n) \approx g(x_{\pi(1)}, \ldots, x_{\pi(m)})$ then

- choosing a variable from the LHS according to the distribution for f and
- choosing a variable from the RHS according to the distribution for g,

with probability greater than ε we will choose the same variable.

In order to find such ε and the mapping for $\mathcal{M}$ we assume without loss of generality that small fixing sets in $\mathcal{M}$ are 1-FIXING-SETs and their size as well as a size of an antichain is bounded by constant M. We choose $\varepsilon < 1/M^4$ and define the probability distribution

as follows: fix $f \in \mathcal{M}$ and from the collection of 1-FIXING-SETs smaller than M choose a maximal subset of pairwise disjoint 1-FIXING-SETs. Let U_f be the set of numbers appearing in this subset and the probability distribution for f is the uniform distribution on U_f.

Take an identity as above; as $|U_f| \leq M^2$ and $|U_g| \leq M^2$ in order to prove the claim it suffices to show that $\pi(U_g) \cap U_f \neq \emptyset$. Let U be one of the 1-FIXING-SETs which defined U_g. The set $\pi(U)$ is a 1-FIXING-SET of f and its size is bounded by M. The maximality of the subset defining U_f implies that U_f and $\pi(U)$ intersect, which concludes the proof. ◀

We are now ready to finish the proof of Theorem 1 (modulo Theorem 2) following a reasoning similar to the one used in [5]. Let $(\mathbf{B}, \mathbf{C})$ be a PCSP language such that $\mathrm{Pol}(\mathbf{B}, \mathbf{C})$ doesn't contain constant functions and do not include any of MAX, MIN, AT, XOR, THR_q, $\overline{\mathrm{MAX}}$, $\overline{\mathrm{MIN}}$, $\overline{\mathrm{AT}}$, $\overline{\mathrm{XOR}}$, $\overline{\mathrm{THR}_q}$. Let $(\mathbf{B}_+, \mathbf{C}_+)$ be $(\mathbf{B}, \mathbf{C})$ with $[\,1 \mid 1\,]_1$ and $[\,0 \mid 0\,]_1$ added. By Theorem 2 and Proposition 5.2 $\mathrm{Pol}(\mathbf{B}_+, \mathbf{C}_+)$ does not satisfy any ε-robust minor condition (for some fixed ε). Note that $\mathrm{Pol}(\mathbf{B}_+, \mathbf{C}_+)$ consists of these elements of $\mathrm{Pol}(\mathbf{B}, \mathbf{C})$ which have identity as the unary minor. Thus $\mathrm{Pol}(\mathbf{B}, \mathbf{C}) \setminus \mathrm{Pol}(\mathbf{B}_+, \mathbf{C}_+)$ consists of elements of $\mathrm{Pol}(\mathbf{B}, \mathbf{C})$ which have $x \mapsto 1 - x$ as the unary minor.

Consider the set $\overline{\mathrm{Pol}(\mathbf{B}, \mathbf{C}) \setminus \mathrm{Pol}(\mathbf{B}_+, \mathbf{C}_+)}$. It is a minion and it is equal to $\mathrm{Pol}(\mathbf{B}_-, \mathbf{C}_-)$, where $(\mathbf{B}_-, \mathbf{C}_-)$ is obtained from $(\mathbf{B}, \mathbf{C})$ in two steps: first the roles of 0 and 1 are swapped in $\mathbf{C}$ (just like in the tractability proof) and then $[\,1 \mid 1\,]_1, [\,0 \mid 0\,]_1$ are added to the language. Applying Proposition 5.2 to $(\mathbf{B}_-, \mathbf{C}_-)$ we conclude that $\overline{\mathrm{Pol}(\mathbf{B}, \mathbf{C}) \setminus \mathrm{Pol}(\mathbf{B}_+, \mathbf{C}_+)}$ does not satisfy any ε-robust minor condition (for some ε). The same holds for $\mathrm{Pol}(\mathbf{B}, \mathbf{C}) \setminus \mathrm{Pol}(\mathbf{B}_+, \mathbf{C}_+)$ and therefore $\mathrm{Pol}(\mathbf{B}, \mathbf{C})$ is a disjoint union of two minions which, for some ε, do not satisfy any ε-robust minor conditions. It follows that $\mathrm{Pol}(\mathbf{B}, \mathbf{C})$ does not satisfy any ε-robust minor condition and by Lemma 5.1 the $\mathrm{PCSP}(\mathbf{B}, \mathbf{C})$ is NP-hard.

6 Proof overview

Our proof of Theorem 2 consists of the following four propositions.

▶ **Proposition 6.1.** *Let $(\mathbf{A}, \mathbf{B})$ be a symmetric language such that $\mathcal{M} = Pol(\mathbf{A}, \mathbf{B})$ is idempotent. If $\mathcal{M}$ does not include neither MAX nor MIN, then it is compatible with some relational pair $[\,a \mid 1, \ldots, a+1\,]_{a+1}$ and some relational pair $[\,1 \mid 0, \ldots, c\,]_{c+1}$.*

For the next proposition we need to specialize the notion of bounded antichains. We say that a minion has *bounded antichains of 1-SETs (0-SETs)* if there exists a uniform bound on the number of pairwise disjoint 1-SETs (0-SETs respectively) an element of the minion can have.

▶ **Proposition 6.2.** *Let $\mathcal{M}$ be a minion compatible with $[\,a \mid 1, \ldots, a+1\,]_{a+1}$ and $[\,1 \mid 0, \ldots, c\,]_{c+1}$. Then $\mathcal{M}$ has bounded antichains of 1-SETs if and only if $\mathcal{M}$ has bounded antichains of 0-SETs.*

▶ **Proposition 6.3.** *Let $(\mathbf{A}, \mathbf{B})$ be a symmetric language such that $\mathcal{M} = Pol(\mathbf{A}, \mathbf{B})$ is idempotent. If $\mathcal{M}$ is compatible with some $[\,a \mid 1, \ldots, a+1\,]_{a+1}$, some $[\,1 \mid 0, \ldots, c\,]_{c+1}$ and does not have bounded antichains then $\mathcal{M}$ includes XOR or AT.*

▶ **Proposition 6.4.** *Let $(\mathbf{A}, \mathbf{B})$ be a symmetric language such that $\mathcal{M} = Pol(\mathbf{A}, \mathbf{B})$ is idempotent. If $\mathcal{M}$ has bounded antichains and does not include any of THR_q then it has small fixing sets.*

The structure of the proof is as follows: if $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ has MIN or MAX we are in a tractable case. Otherwise we split the reasoning in two cases: either $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ fails the bounded antichain condition and by Proposition 6.3 we are tractable due to AT or XOR, or

we have bounded antichains and by Proposition 6.4 we are either tractable due to THR_q or have small fixing sets which implies hardness (by Proposition 5.2). Proposition 6.2 allows us to "flip" the template if necessary.

In this section, we prove Propositions 6.1 and 6.2. We also provide proof sketches of Propositions 6.3 and 6.4. Detailed proofs can be found in the full version of the paper.

Proof of Proposition 6.1. The proof splits into two parts:

- $\mathcal{M}$ does not have MIN then $\mathcal{M}$ is compatible with $[\,a \mid 1,\dots,a+1\,]_{a+1}$
- $\mathcal{M}$ does not have MAX then $\mathcal{M}$ is compatible with $[\,1 \mid 0,\dots,c\,]_{c+1}$

Proof of both cases is analogous, so we will only prove the first part. Let us assume that $\mathcal{M} = \mathrm{Pol}(\mathbf{A},\mathbf{B})$ and $\mathcal{M}$ does not have MIN. So there must be $[\,I \mid J\,]_n$ in the language of $(\mathbf{A},\mathbf{B})$ such that MIN is not compatible with it. This implies that there exists $b < a < n$ such that $a \in I$ and $b \notin J$. Now, using pp-definitions and strict relaxations from Section 4, we will show that $\mathcal{M}$ is compatible with $[\,a-b \mid 1,\dots,a-b+1\,]_{a-b+1}$:

- use strict relaxation of $[\,I \mid J\,]_n$ to obtain $[\,a \mid 0,\dots,b-1,b+1,\dots,n\,]_n$;
- from the last pair pp-define, using $[\,0 \mid 0\,]_1$, the pair $[\,a \mid 0,\dots,b-1,b+1,\dots,a+1\,]_{a+1}$,
- finally from the previous pair pp-define, this time using $[\,1 \mid 1\,]_1$, the required pair $[\,a-b \mid 1,\dots,a-b+1\,]_{a-b+1}$. ◀

The following lemma is used in the proof of Proposition 6.2.

▶ **Lemma 6.5.** *Let $\mathcal{M}$ be a minion. Then:*

- *if $\mathcal{M}$ is compatible with some $[\,a \mid 1,\dots,a+1\,]_{a+1}$, then for each f in $\mathcal{M}$ a union of a-many pairwise disjoint 0-SETs is a 1-SET.*
- *if $\mathcal{M}$ is compatible with some $[\,1 \mid 0,\dots,c\,]_{c+1}$, then for each f in $\mathcal{M}$ a union of c-many pairwise disjoint 1-SETs is a 0-SET.*

Proof. The proofs of the two cases are analogous, so we will only prove the second one. Let $U_1,\dots,U_c$ be disjoint 1-SETs of the n-ary function $f \in \mathcal{M}$ and $U = \bigcup_{i=1}^{c} U_i$. Since every coordinate i occurs in exactly one set of $U_1,\dots,U_c,\overline{U}$ and f is compatible with $[\,1 \mid 0,\dots,c\,]_{c+1}$, the tuple $(f(U_1),\dots,f(U_c),f(\overline{U}))$ cannot evaluate to $(1,\dots,1)$. Therefore $f(\overline{U}) = 0$ and U is a 0-SET. See Figure 1 for example.

1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0

Figure 1 Example of c disjoint 1-SETs creating a 0-SET with $c = 3$. The yellow column represents the result of an evaluation of function f on tuples represented by other columns. The columns are in $[\,1 \mid 0,1,2,3\,]_4$ and the grey cells are $U_1,\dots,U_c$ while the red cells are U. ◀

Proof of Proposition 6.2. By using Lemma 6.5 we conclude that:

- if f contains an antichain of 1-SETs of size n then it also contains an antichain of 0-SETs of size at least $\lfloor \frac{n}{c} \rfloor$
- if f contains an antichain of 0-SETs of size n then it also contains an antichain of 1-SETs of size at least $\lfloor \frac{n}{a} \rfloor$

so if one of the antichains of 0-SETs or 1-SETs is bounded then the other one has to be bounded as well. ◀

Proof sketch of Proposition 6.3. Since $\mathcal{M}$ has unbounded antichains, we can take a function from $\mathcal{M}$ with a arbitrarily large antichain of 1-SETs. By taking its minor, we obtain f satisfying

$$f(1,0,\dots,0) = f(0,1,0,\dots,0) = \cdots = f(0,\dots,0,1,0) = 1.$$

Notice that the last coordinate is exceptional, it does not have to form a 1-SET. By taking further minors of f we either get g, of arbitrarily large arity, that satisfies

$$g(1,0,\dots,0) = g(0,1,0,\dots,0) = \cdots = g(0,\dots,0,1) = 1,$$

or compatibility with AT (see the full version of the paper). We are left with the case when g's, of arbitrarily large arity, are in $\mathcal{M}$.

If $\mathcal{M}$ does not include AT, it is compatible (after possibly changing ones to zeros and zeros to ones) with $[\,1 \mid 0,\dots,n-2,n\,]_n$ or $[\,0,d \mid 0,\dots,n-1\,]_n$ for some $d < n$. We use these relational pairs for forcing further behavior of g, and finally obtain an xor of an arbitrarily large arity. This implies that XOR is a subset of $\mathcal{M}$. ◀

Proof sketch of Proposition 6.4. If a minion $\mathcal{M}$ has bounded antichains and does not have q-threshold for any q, we can find (skipping an easy case discussed in the full version of the paper) positive integers a, b, c, d such that $c/d < a/b < 1$ such that $\mathcal{M}$ is compatible with relational pairs

$$[\,a \mid 0,\dots,b-1\,]_b, \quad [\,c \mid 1,\dots,d\,]_d. \tag{1}$$

Notice that the converse, i.e. that these relational pairs prevent threshold, is clear since (1) disallows any q-threshold such that $q < a/b$ and any q-threshold such that $q > c/d$. It can be shown that these relational pairs are the general obstacle to a threshold polymorphism. We prove the proposition by induction on $a + b + c + d$.

For the reminder of the proof to work we are forced to work with weaker assumptions – instead of $\mathcal{M}$ being compatible with (1) we assume that $\mathcal{M}$ is "almost compatible" with the relational pairs. Nevertheless, the "almost compatibility" notion is rather technical, and we ignore it in this sketch. For a formal proof, see the full version of the paper. Here, let us simply assume that $\mathcal{M}$ is compatible with (1).

It turns out that the only interesting case is $c/d < a/b < 1/2$. All the other cases can be either resolved directly or reduced to this one. Now, consider a minimal (ordered by inclusion) 0-SET U and let f_U denote $|U|$-ary operation obtained from f by plugging zeros to every coordinate not contained in U. Since f is compatible with $[\,a \mid 0,\dots,b-1\,]_b$ and U is a 0-SET, f_U is compatible with $[\,c \mid 1,\dots,d-c\,]_{d-c}$. Every 1-SET in f_U is also a 1-SET in f, so f_U has bounded antichains of 1-SETs. (bounded across every $f \in \mathcal{M}$ and every U). Moreover, since U is minimal, the complement $\overline{U}$ of U is "almost" a 1-SET (every strict superset is). If $\overline{U}$ was a 1-SET, f_U would be compatible with $[\,a \mid 0,1,\dots,b-a-1\,]_{b-a}$ since f is compatible with $[\,a \mid 0,1,\dots,b-1\,]_b$. This is where the weaker notion of compatibility (the star-compatibility) is necessary in the full proof. However for the sake of simplicity, assume that f_U is compatible with $[\,a \mid 0,1,\dots,b-a-1\,]_{b-a}$. Since f_U has bounded antichains of 1-SETs and it is compatible with relational pairs

$$[\,a \mid 0,1,\dots,b-a-1\,]_{b-a}, \quad [\,c \mid 1,\dots,d-c\,]_{d-c}$$

where $c/(d-c) < a/(b-a)$, it has also bounded antichains of 0-SETs. Therefore, we can apply the induction hypothesis and obtain a small (bounded across every $f \in \mathcal{M}$ and every U) 1-FIXING-SET or 0-FIXING-SET V in f_U. For our purposes, we don't need to know

that the set is fixing, it suffices that it is a 0-SET or a 1-SET. Let $\mathcal{L}_f$ denote the set of all possible sets V above across all minimal 0-SETs U. From the induction hypothesis, we also get that either every $V \in \mathcal{L}_f$ is a 1-SET in the appropriate f_U, or every $V \in \mathcal{L}_f$ is a 0-SET in the appropriate f_U.

▷ Claim 4. The size of pairiwise disjoint subsystems of $\mathcal{L}_f$ is bounded by a number independent of the chosen $f \in \mathcal{M}$.

If every $V \in \mathcal{L}_f$ is a 1-SET in the appropriate f_U, then V is a 1-SET in f and the claim follows from $\mathcal{M}$ having bounded antichains. Let us prove the claim if every $V \in \mathcal{L}_f$ is a 0-SET in the appropriate f_U. Consider c disjoint elements $V_1, \ldots, V_c \in \mathcal{L}$, and let $U_1, \ldots, U_c$ be the appropriate minimal 0-SETs. Thus also every $\overline{U_i} \cup V_i$ is a 0-SET. Since

$$U_1, U_2, \ldots, U_c, \overline{U_1} \cup V_1, \overline{U_2} \cup V_2, \ldots, \overline{U_c} \cup V_c$$

are 0-SETs, $V_1 \cup \ldots \cup V_c$ is a 1-SET by compatibility with $[\, c \mid 1, 2, \ldots 2c+1 \,]_{2c+1}$. Let M be the bound on antichains of 1-SETs in $\mathcal{M}$, the size of antichains in $\mathcal{L}$ is bounded by cM.

Finally, we use the claim to find a small 1-FIXING-SET in f. Consider any maximal sequence $V_1, \ldots, V_n \in \mathcal{L}$ of disjoint sets and let

$$W = V_1 \cup V_2 \cup \ldots \cup V_n,$$

Every 0-SET contains a minimal 0-SET, every minimal 0-SET contains some $V \in \mathcal{L}$ and every $V \in \mathcal{L}$ intersects W. Therefore every 0-SET intersects W, so W is the desired 1-FIXING-SET. ◀

References

1. P. Austrin, J. Håstad, and V. Guruswami. (2 + epsilon)-Sat Is NP-Hard. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 1–10, October 2014. doi:10.1109/FOCS.2014.9.
2. Libor Barto and Marcin Kozik. Robust Satisfiability of Constraint Satisfaction Problems. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 931–940, New York, NY, USA, 2012. ACM. doi:10.1145/2213977.2214061.
3. Joshua Brakensiek and Venkatesan Guruswami. New Hardness Results for Graph and Hypergraph Colorings. In *Proceedings of the 31st Conference on Computational Complexity*, CCC '16, pages 14:1–14:27, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CCC.2016.14.
4. Joshua Brakensiek and Venkatesan Guruswami. An Algorithmic Blend of LPs and Ring Equations for Promise CSPs. *CoRR*, abs/1807.05194, 2018. arXiv:1807.05194.
5. Joshua Brakensiek and Venkatesan Guruswami. *Promise Constraint Satisfaction: Structure Theory and a Symmetric Boolean Dichotomy*, pages 1782–1801. SIAM, 2018. doi:10.1137/1.9781611975031.117.
6. A. A. Bulatov. A Dichotomy Theorem for Nonuniform CSPs. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, volume 00, pages 319–330, October 2017. doi:10.1109/FOCS.2017.37.
7. Andrei A. Bulatov, Andrei A. Krokhin, and Peter Jeavons. Constraint satisfaction problems and finite algebras. In *Automata, languages and programming (Geneva, 2000)*, volume 1853 of *Lecture Notes in Comput. Sci.*, pages 272–282. Springer, Berlin, 2000.
8. Jakub Bulín, Andrei A. Krokhin, and Jakub Oprsal. Algebraic approach to promise constraint satisfaction. *CoRR*, abs/1811.00970, 2018. arXiv:1811.00970.
9. Irit Dinur, Elchanan Mossel, and Oded Regev. Conditional Hardness for Approximate Coloring. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, STOC '06, pages 344–353, New York, NY, USA, 2006. ACM. doi:10.1145/1132516.1132567.

10 Tomás Feder and Moshe Y. Vardi. The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study through Datalog and Group Theory. *SIAM Journal on Computing*, 28(1):57–104, 1998. doi:10.1137/S0097539794266766.

11 Miron Ficak, Marcin Kozik, Miroslav Olsak, and Szymon Stankiewicz. Dichotomy for symmetric Boolean PCSPs, 2019. arXiv:1904.12424.

12 Sangxia Huang. Improved Hardness of Approximating Chromatic Number. In Prasad Raghavendra, Sofya Raskhodnikova, Klaus Jansen, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 233–243, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

13 Peter Jeavons, David Cohen, and Marc Gyssens. Closure properties of constraints. *J. ACM*, 44(4):527–548, 1997.

14 Vladimir Kolmogorov, Andrei Krokhin, and Michal Rolinek. The Complexity of General-Valued CSPs. In *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, FOCS '15, pages 1246–1258, Washington, DC, USA, 2015. IEEE Computer Society. doi:10.1109/FOCS.2015.80.

15 Thomas J. Schaefer. The complexity of satisfiability problems. In *Conference Record of the Tenth Annual ACM Symposium on Theory of Computing (San Diego, Calif., 1978)*, pages 216–226. ACM, New York, 1978.

16 D. Zhuk. A Proof of CSP Dichotomy Conjecture. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, volume 00, pages 331–342, October 2017. doi:10.1109/FOCS.2017.38.

Biasing Boolean Functions and Collective Coin-Flipping Protocols over Arbitrary Product Distributions

Yuval Filmus
Computer Science Department, Technion, Haifa, Israel
http://www.cs.toronto.edu/~yuvalf/
yuvalfi@cs.technion.ac.il

Lianna Hambardzumyan
School of Computer Science, McGill University, Montreal, QC, Canada
lianna.hambardzumyan@mail.mcgill.ca

Hamed Hatami
School of Computer Science, McGill University, Montreal, QC, Canada
https://www.cs.mcgill.ca/~hatami/
hatami@cs.mcgill.ca

Pooya Hatami
Department of Computer Science, UT Austin, Austin, TX, USA
https://pooyahatami.org/
pooyahat@gmail.com

David Zuckerman
Department of Computer Science, UT Austin, Austin, TX, USA
http://www.cs.utexas.edu/~diz/
diz@cs.utexas.edu

Abstract

The seminal result of Kahn, Kalai and Linial shows that a coalition of $O(\frac{n}{\log n})$ players can bias the outcome of *any* Boolean function $\{0,1\}^n \to \{0,1\}$ with respect to the uniform measure. We extend their result to arbitrary product measures on $\{0,1\}^n$, by combining their argument with a completely different argument that handles very biased input bits.

We view this result as a step towards proving a conjecture of Friedgut, which states that Boolean functions on the continuous cube $[0,1]^n$ (or, equivalently, on $\{1,\ldots,n\}^n$) can be biased using coalitions of $o(n)$ players. This is the first step taken in this direction since Friedgut proposed the conjecture in 2004.

Russell, Saks and Zuckerman extended the result of Kahn, Kalai and Linial to multi-round protocols, showing that when the number of rounds is $o(\log^* n)$, a coalition of $o(n)$ players can bias the outcome with respect to the uniform measure. We extend this result as well to arbitrary product measures on $\{0,1\}^n$.

The argument of Russell et al. relies on the fact that a coalition of $o(n)$ players can boost the expectation of any Boolean function from ϵ to $1-\epsilon$ with respect to the uniform measure. This fails for general product distributions, as the example of the AND function with respect to $\mu_{1-1/n}$ shows. Instead, we use a novel boosting argument alongside a generalization of our first result to arbitrary finite ranges.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Boolean function analysis, coin flipping

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.58

Category Track A: Algorithms, Complexity and Games

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 58; pp. 58:1–58:13

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

Related Version A full version of the paper is available at `https://eccc.weizmann.ac.il/report/2019/029/`.

Funding *Yuval Filmus*: Taub Fellow – supported by the Taub Foundations. The research was funded by ISF grant 1337/16.
Hamed Hatami: Supported by an NSERC grant.
Pooya Hatami: Supported by a Simons Investigator Award (#409864, David Zuckerman).
David Zuckerman: Supported by NSF Grant CCF-1705028 and a Simons Investigator Award (#409864).

Acknowledgements Part of the work on this paper was done while the first three authors were at the Simons Institute for the Theory of Computing at Berkeley, CA, USA.

1 Introduction

How can distributed processors collectively flip a somewhat fair coin if some processors may try to bias the outcome? In the *Collective Coin-Flipping Problem*, a classical problem in distributed computing, n processors wish to generate a single common random bit, even in the presence of faulty and possibly malicious processors. Collective coin-flipping protocols can be used to expedite *Byzantine Agreement* [6] and are closely related to *Leader Election Protocols* [7]. The problem has been considered in several scenarios, depending on the assumptions made on the type of the communication between the processors, the kind and number of faults, and the power of the adversary [6, 3, 7, 2].

A Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, where $\{0,1\}^n$ is endowed with a product measure μ, naturally corresponds to a *single round collective coin-flipping protocol* in the *perfect information* model introduced by Ben-Or and Linial [2], where n players each broadcast a bit according to a private distribution, and at the end, the output of the protocol is the value of f on the broadcast string. An interesting and important concept in the design of collective coin-flipping protocols is *resilience* against coalitions of a significant number of players who attempt to influence the output of the protocol towards a particular value.

A *coalition* is a subset S of players that have a particular desired value $b \in \{0,1\}$ in mind, and if possible, broadcast bits that set the output of the protocol to b. We study the model where the coalition is allowed *rushing*: the corrupt players may wait until all the other players broadcast their bits before deciding on what bit to broadcast. In other words, they *succeed* on $x \sim \mu$ if it is possible to modify x only on the coordinates in S to obtain a string y with $f(y) = b$; they *fail* if the value of f is already determined to be *not* equal to b by the bits broadcast by the players outside the coalition. The success of such a coalition can be easily quantified as the probability that the coalition succeeds on a random $x \sim \mu$.

Fix a parameter $\epsilon > 0$. A protocol f is said to be *ϵ-resilient against coalitions of ℓ players* if no coalition of size at most ℓ succeeds with probability at least $1 - \epsilon$. How resilient can a function be against large coalitions? Over the uniform distribution, perhaps the most natural candidate for a highly resilient function is the majority function, which can be easily seen to be resilient against $\Omega(\sqrt{n})$ size coalitions. However, somewhat surprisingly, it turns out that plain democracy is not the most effective way to be immune against the influence of coalitions. Indeed, Ajtai and Linial [1] gave a *randomized* construction of a Boolean function that is resilient against coalitions of size $\Omega(n/\log^2 n)$, significantly better than the $\Omega(\sqrt{n})$ bound of the majority function. More recently, Chattopadhyay and Zuckerman [5] gave an *explicit* construction of a highly resilient function over the uniform measure. This was a key ingredient in their breakthrough work that introduced explicit two-source extractors

for polylogarithmic min-entropy. Subsequently, Meka [11] gave an explicit construction of a monotone depth three Boolean function that is as resilient as the randomized construction of Ajtai and Linial.

In this article, we are mainly interested in the limitations of resilience. The most classical theorem in this direction is due to Kahn, Kalai, and Linial [10], who proved that, for the uniform distribution, no Boolean function is resilient against coalitions of size $\omega(n/\log n)$. Closing the gap between this bound and the $\Omega(n/\log^2 n)$ construction of Ajtai and Linial remains a longstanding open problem.

Starting with the work of Ben-Or and Linial [2], researchers have studied two natural ways to generalize the discussed protocols: First, allow players to broadcast longer messages, and second, allow many rounds. In this paper, we mostly focus on the latter generalization. In the multi-round setting, the voting procedure that is described above is repeated r times: at every round, first the players who are not in the coalition broadcast their random messages, and then the players in the coalition decide and broadcast their messages in an adversarial manner. When the players are sending single-bit messages, the outcome is decided by a function $f\colon (\{0,1\}^n)^r \to \{0,1\}$.

The most efficient known protocols are due to Russell and Zuckerman [13] and to Feige [8]. In the case where players are allowed to send longer messages, they constructed $\log^\star n + O(1)$ round protocols resilient against coalitions of size βn for any $\beta < 1/2$. In the case when players are allowed to broadcast single bit messages, their protocols use $(1 - o(1))\log n$ rounds, and are still resilient against coalitions of size βn for any $\beta < 1/2$. For a discussion of various models and known upper and lower bounds, see a survey of Dodis [7].

In the multi-round setting, the players in the coalition have the disadvantage that they will not see the future-round votes of the other players before voting in the current round. Thus, it becomes significantly more difficult to prove limitations on resilience as r grows, and naturally the known bounds are weaker. Russell, Saks and Zuckerman [12], building upon the work of Kahn et al. [10], showed that over the uniform measure, no Boolean function $f\colon (\{0,1\}^n)^r \to \{0,1\}$ is ϵ-resilient against coalitions of size $\omega_\epsilon\left(\frac{r^2 n}{\log^{(2r-1)} n}\right)$, where $\log^{(2r-1)} n$ is an iterated logarithm. It follows as a simple corollary that $\Omega(\log^\star n)$ rounds are necessary in order for a protocol to be resilient against coalitions of size $\Omega(n)$.

The purpose of this paper is to generalize the above results from the uniform distribution to arbitrary product distributions on the Boolean cube.

A moment of reflection reveals that there are major differences between the uniform distribution and the general case, and indeed, prior to this work, it was not clear to us whether similar results were true for general product distributions. We will elaborate on this later, but for now, we only mention that the coordinates x_i that are not highly biased, i.e. $t \le \Pr[x_i = 1] \le 1 - t$ for some t that is not too small, can be handled using the same argument as in Kahn et al. [10]. Similarly, the argument of Russell et al. [12] can be used to analyze these coordinates in the multi-round setting. However, the highly biased coordinates behave very differently, and to handle those, we need to take an entirely new approach, and employ a new set of ideas. Indeed, our proofs for the highly biased case have almost no resemblance to those in previous works.

Our first theorem concerns single round protocols. By combining the argument of Kahn, Kalai and Linial with an argument geared towards biased coordinates, we are able to show that these protocols can always be influenced towards a single value, with coalitions which are only slightly worse than those guaranteed by the KKL theorem.

▶ **Theorem 1.** *Over any product distribution μ, there is no function $f\colon \{0,1\}^n \to \{0,1\}$ that is ϵ-resilient against coalitions of size $\omega_\epsilon(\frac{n \log\log n}{\log n})$.*

(In contrast, the KKL theorem shows the impossibility of ϵ-resilience against coalitions of size $\omega_\epsilon(\frac{n}{\log n})$.)

Next, we prove an impossibility result for resilience in the multi-round setting over arbitrary product distributions. This was posed as an open problem by Russell et al. [12]. Here we face several new challenges. Generalizing our argument for the biased coordinates to the multi-round setting is far from straightforward, and combining it with the argument of Russell et al. [12] for the unbiased coordinates also requires new ideas.

▶ **Theorem 2.** *Let n and $r < n$ be given. Over any product distribution μ over $(\{0,1\}^n)^r$, there is no r-round coin-flipping protocol $f\colon (\{0,1\}^n)^r \to \{0,1\}$ that is ϵ-resilient against coalitions of size $\omega_\epsilon\left(\frac{n(\log^\star n)^2}{\log^{(4r)} n}\right)$.*

As a result, over any product distribution μ, $\Omega(\log^\star n)$ rounds are necessary in order for a protocol to be resilient against coalitions of size $\Omega(n)$.

Influences. The notion of resilience of a Boolean function is related to the influences of variables and coalitions of variables. For a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ over a product probability measure μ, the *influence* of the k-th variable is defined as

$$I_k(f) := \Pr_{x\sim\mu}\left[f \text{ is not constant on } B_k(x)\right],$$

where $B_k(x) := \{y \in \{0,1\}^n \colon y_j = x_j \text{ for all } j \neq k\}$.

The influence of the k-th variable *towards a value* $b \in \{0,1\}$ is defined as

$$I_k^b(f) := \Pr_{x\sim\mu}\left[b \in f(B_k(x))\right].$$

Similarly, the *influence of a coalition* $S \subseteq [n]$ towards a value $b \in \{0,1\}$ is defined as

$$I_S^b(f) := \Pr_{x\sim\mu}\left[b \in f(B_S(x))\right],$$

where $B_S(x) := \{y \in \{0,1\}^n \colon y_j = x_j \text{ for all } j \notin S\}$.

Equivalently, $I_S^b(f)$ is the probability that a random $x \sim \mu$ can be modified on its S variables such that the output of f becomes b.

A function f is *not* ϵ-resilient against coalitions of size ℓ if and only if there exists a set S of size at most ℓ and a value b such that $I_S^b(f) \geq 1 - \epsilon$.

The seminal work of Kahn, Kalai and Linial introduced discrete Fourier-analytic techniques to the study of influences. Their main theorem, known as the KKL inequality, states that over the uniform measure, every unbiased Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ has an influential variable. Formally, there exists k such that $I_k(f) \geq \Omega(\frac{\alpha \log n}{n})$ when $\alpha \leq \mathbb{E}[f(x)] \leq 1 - \alpha$. Let $b \in \{0,1\}$ satisfy $\Pr[f(x) = b] \geq \epsilon$. Then repeated applications of the KKL inequality imply the existence of a set S with $|S| = O_\epsilon\left(\frac{n}{\log n}\right)$ such that $I_S^b(f) \geq 1 - \epsilon$. In particular, there are no $\omega_\epsilon(n/\log n)$-resilient functions over the uniform distribution.

The above argument shows that unless f is already very biased towards 0 or 1, one can pick any $b \in \{0,1\}$ and find a small coalition S that can bias f towards b. However, this is no longer true if we consider general product distributions.

▶ **Example 3.** Consider the p-biased distribution μ_p^n over $\{0,1\}^n$, i.e. each coordinate is 1 with probability p. Set $p = 1/n$ and let f be the OR function $\bigvee_{i=1}^n x_i$. Obviously, $\mathbb{E}[f] = 1-(1-p)^n \approx 1 - \frac{1}{e}$, and yet for every S with $|S| = o(n)$, we have $I_S^0(f) = 1-(1-p)^{n-|S|} \approx 1-\frac{1}{e}$. In other words, despite the fact that the expected value of the function is bounded away from both 0 and 1, no small coalition can influence the output of the function towards 0. However, this is not a counterexample to Theorem 1 because any set S with $|S| = 1$ satisfies $I_S^1(f) = 1$, and thus the function is not even 1-resilient.

As the above example illustrates, part of the difficulty of generalizing the coalition theorem of KKL is to figure out which $b \in \{0,1\}$ to bias towards.

Using the notation $I_S^b(f)$, Theorem 1 can be restated as follows.

▶ **Theorem 4** (Theorem 1 reformulated). *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a function over a product distribution μ. There exists a set S of size $O_\epsilon(\frac{n \log\log n}{\log n})$ such that $I_S^b(f) \geq 1-\epsilon$ for some $b \in \{0,1\}$.*

▶ Remark 5. To simplify the statement, in Theorem 4, we did not explicitly state the dependence of $|S|$ on ϵ. Our proof yields the bound $|S| = O(\frac{\log(1/\epsilon)n}{\epsilon \log n} + \frac{n \log\log n}{\epsilon \log n})$.

Continuous cube and a conjecture of Friedgut. The Bernoulli distribution on $\{0,1\}$ with parameter p can be embedded in the continuous interval $[0,1]$ via the measure-preserving map $\sigma\colon [0,1] \to \{0,1\}$ defined as $\sigma(x) = 1$ if and only if $x \geq 1-p$. By taking the product of these maps, for every product probability measure μ on $\{0,1\}^n$, we obtain a measure-preserving map $\sigma_\mu\colon [0,1]^n \to \{0,1\}^n$. As a result, every function $f\colon (\{0,1\}^n, \mu) \to \{0,1\}$ naturally corresponds to a function $\overline{f}\colon [0,1]^n \to \{0,1\}$ defined by $\overline{f} = f \circ \sigma_\mu$. Note that $I_S^b[\overline{f}] = I_S^b[f]$, for every $S \subseteq [n]$ and $b \in \{0,1\}$. Thus, a more general setting for studying resilience is the set of measurable functions $f\colon [0,1]^n \to \{0,1\}$. Indeed, Bourgain et al [4] proved a generalization of the KKL inequality, but erroneously claimed that as a corollary, if $\epsilon \leq \mathbb{E}[f]$, then $I_S^1[f] \geq 1-\epsilon$ for a set S of size $|S| = o_\epsilon(n)$. Interestingly, Example 3, which was introduced in the same paper to demonstrate that the proof of the KKL inequality breaks down for the continuous cube, is also a counterexample to this false claim. Friedgut [9] pointed out this error, and suggested the following tantalizing conjecture to replace the false statement[1].

▶ **Conjecture 6** ([9]). *Let $f\colon [0,1]^n \to \{0,1\}$ be a measurable function. There exists a set S of size $o_\epsilon(n)$ such that $I_S^b(f) \geq 1-\epsilon$ for some $b \in \{0,1\}$.*

A standard compression argument shows that it suffices to prove this conjecture for increasing functions, and indeed the original form of the conjecture is stated for increasing functions. Furthermore, by discretization, the statement can be further reduced to functions $f\colon \{1,\dots,n^2\}^n \to \{0,1\}$, where the domain is endowed with the uniform measure. Note that this form of the conjecture corresponds to resilience of one-round collective coin-flipping protocols where each player is allowed to send $\log n$-bit messages.

The above discussion show that, qualitatively, Conjecture 6 is a generalization of Theorem 4, and thus our theorem can be considered as a step towards resolving Friedgut's conjecture. However, our techniques and ideas seem to fall short of proving the full conjecture.

[1] Nati Linial told the last author about this error and conjecture years earlier, but as far as we know this is the first published account.

Beyond the Boolean range. As we discussed above, the coalition theorem of KKL says that if $\mathbb{E}[f(x) = b] \geq \epsilon$ then there exists a small coalition S such that $I_S^b(f) \geq 1 - \epsilon$. Now consider a function $h\colon \{0,1\}^n \to \mathcal{R}$ over the uniform distribution, where $\mathcal{R}$ is a constant size set. Pick any $b \in \mathcal{R}$ with $\Pr[h(x) = b] \geq \epsilon$. We can apply the KKL theorem to the function $f\colon \{0,1\}^n \to \{0,1\}$ defined as $f(x) = 1$ if and only if $h(x) = b$, and conclude that there is a coalition of size $O_\epsilon(n/\log n)$ with $I_S^b(f) \geq 1 - \epsilon$. This shows that over the uniform distribution, the general range $\mathcal{R}$ easily reduces to the Boolean range.

Unfortunately, the above reduction cannot be carried for general product distributions, for in Theorem 4, the final outcome b is dictated to us by the function. To illustrate the problem, consider a function $h\colon \{0,1\}^n \to \{0,1,2\}$ and a general product distribution μ. By bundling $\{1,2\}$ into a single value and applying Theorem 4, we can conclude that there exists a small coalition S such that either it biases the outcome of the function towards 0, or it biases the outcome towards being in $\{1,2\}$. If it is the former case, then we are done, but in the latter case, it is not clear how to proceed.

We know that except for the x's that belong to a small-measure set $\mathcal{E}$, the coalition can modify x in such a way that the outcome is in $\{1,2\}$. Now at first glance, it might seem that by applying Theorem 4 again, we can find another coalition T that can modify x further to refine the outcome to a single value $b \in \{1,2\}$, and thus conclude that for most x's the alliance $S \cup T$ can influence the outcome of the function towards b. Unfortunately, this is actually not the case. One reason is that S and T might intersect, and suggest conflicting modifications to x. Even if S and T are disjoint, the proof doesn't work: denoting by x' the vector obtained from $x \sim \mu$ after modification by S, we no longer have $x' \sim \mu$, and so there is no guarantee that on most inputs T can be applied successfully. In other words, $\Pr[x' \in \mathcal{E}]$ need not be small.

The above discussion shows that one cannot deduce the general case via the simple reduction that was outlined above for the uniform measure, but surely, as cumbersome as it may be, one can go over the proof and generalize every step from $\{0,1\}$ to $\{0,1,2\}$ by making small notational adjustments. This turns out not to be the case either! The proof of Theorem 4, rather unexpectedly, relies on the assumption that the function takes only two values. Indeed, to generalize the result to larger ranges, we had to introduce new ideas, and in particular a strengthening of Theorem 4 (see Theorem 15 below) that provides stronger control over the set $\mathcal{E}$ described above.

▶ **Theorem 7** (Single round, general range)**.** *Let $\mathcal{R}$ be a constant size set, and $f\colon \{0,1\}^n \to \mathcal{R}$ be a function over a product distribution μ. There exists a set S of size $O_\epsilon(\frac{n \log\log n}{\log n})$ such that $I_S^b(f) \geq 1 - \epsilon$ for some $b \in \mathcal{R}$.*

▶ Remark 8. At the heart of the proof of Theorem 7 there is an intermediate result, Theorem 15, which states that if all coordinates are biased, say $\Pr[x_i = 1] < \alpha$, then a random coalition of size $O(\log^3 |\mathcal{R}| \log\log |\mathcal{R}| \cdot \alpha n)$ biases the outcome with high probability. This intermediate result is an essential ingredient in the proof of our result on the multi-round setting, Theorem 2. For this application, it was crucial to obtain a bound which depends only polylogarithmically in $|\mathcal{R}|$.

Even though Theorem 4 is a special case of Theorem 7, we prove them separately, as Theorem 4 can be proven using a shorter and simpler proof.

Paper organization. We prove Theorem 1, which shows that all single-round protocols can be biased using coalitions of size $o(n)$, in Section 2. We prove Theorem 7, which generalizes the preceding result to arbitrary finite domains, in Section 3. We prove our main result, Theorem 2, which shows the multi-round protocols can be biased, in Section 4.

Due to space constraints, some of the proofs are available only in the full version of the paper, which is attached as an appendix.

2 Single Round Case: Proof of Theorem 1

In this section we prove Theorem 1, showing that, under any product distribution, there exists a small coalition which can bias the output of the function towards one of the outputs.

Note that in order to prove Theorem 1, without loss of generality, we can assume that $\Pr_{x\sim\mu}[x_i = 1] \leq \frac{1}{2}$ for every $i \in [n]$, as otherwise we can simply change the role of 0 and 1 for the i-th coordinate. In light of this observation, the coordinates can be divided into two sets: the small bias coordinates, satisfying $\Pr_{x\sim\mu}[x_i = 1] \in (\alpha_0, \frac{1}{2}]$, and the highly biased coordinates, satisfying $\Pr_{x\sim\mu}[x_i = 1] \leq \alpha_0$, where α_0 is a threshold that is chosen to be $\alpha_0 = \frac{1}{\log n}$.

Indeed, we first consider the case where all the coordinates are of the same type:

- *Small bias case*: $\Pr_{x\sim\mu}[x_i = 1] \in (\alpha_0, \frac{1}{2}]$ for every $i \in [n]$.
- *Large bias case*: $\Pr_{x\sim\mu}[x_i = 1] \leq \alpha_0$ for every $i \in [n]$.

We handle the large bias case in Section 2.1, which is the novel part of the proof. The small bias case is handled in Section 2.2 via a reduction to the previous work of Russell et al. [12]. Finally, in Section 2.3 we show how to combine the two cases to handle any product distribution μ, thus completing the proof of Theorem 1.

2.1 Large Bias Case

We will sometimes identify the subsets of $[n]$ with elements of $\{0,1\}^n$. For example, $S \sim \mu$ would mean that $S = \mathrm{supp}(x)$, where x is sampled according to μ. We construct the coalitions from a certain boosted form of μ.

▶ **Definition 9** (Boosted distribution). *For a positive integer t, we denote by $\mu^{(t)}$ the distribution of $x^1 \vee \cdots \vee x^t$, where $x^1, \ldots, x^t$ are i.i.d. random variables distributed according to μ.*

The large bias case of Theorem 1 follows from the following general proposition, that holds for distributions that are not necessarily product distributions.

▶ **Proposition 10.** *Consider $f\colon (\{0,1\}^n, \mu) \to \{0,1\}$, where μ is an* arbitrary *probability measure, and let $S \sim \mu^{(k)}$, where $k \approx \frac{10 \log \frac{1}{\epsilon}}{\epsilon}$. For some $b \in \{0,1\}$, we have $\Pr_S[I^b_S[f] > 1-\epsilon] > 1-\epsilon$.*

Note that Proposition 10 implies (via a straightforward concentration bound) that in the large bias case, there exists a random coalition of expected size at most $k\alpha_0 n$ such that $\Pr_S[I^b_S[f] > 1-\epsilon] > 1-\epsilon$. As it will become apparent later, for the application to the multiround setting, it is important that in Proposition 10 the set S is chosen randomly from a distribution that does not depend on f.

Proposition 10 is a direct consequence of the following lemma, as for the Boolean range $\{0,1\}$, either Condition I holds for $b = 0$ or Condition II holds for $b = 1$. This, however, is not true for larger $\mathcal{R}$.

▶ **Lemma 11** (Key Lemma for Single Round). *Consider $f\colon (\{0,1\}^n, \mu) \to \mathcal{R}$, where μ is an* arbitrary *probability measure. Let $x, y \sim \mu$, $S \sim \mu^{(k)}$, where $k \approx \frac{10 \log \frac{1}{\epsilon}}{\epsilon}$. For $b \in \mathcal{R}$, either of*

- *Condition I:* $\Pr_x[\Pr_y[f(x \vee y) = b] \geq 1-\epsilon] > \epsilon/2$, *or*
- *Condition II:* $\Pr_x[\Pr_y[f(x \vee y) = b] \geq \epsilon] \geq 1 - \epsilon/2$,

implies $\Pr_S[I^b_S[f] > 1-\epsilon] > 1-\epsilon$.

Proof. Let $S = \mathrm{supp}(y^1 \vee \cdots \vee y^k)$, where $y^1, \ldots, y^k \sim \mu$ are drawn independently. Let the sets X^I and X^{II} denote the following subsets of the input space $\{0,1\}^n$:

$$X^I = \{x : \Pr_y[f(x \vee y) = b] \geq 1 - \epsilon\},$$
$$X^{II} = \{x : \Pr_y[f(x \vee y) = b] \geq \epsilon\}.$$

If we are in the Type I setting, then $\Pr\left[X^I\right] > \epsilon/2$, and so

$$\Pr_S[S \text{ contains some } x \in X^I] \geq 1 - \Pr\left[y^1, \ldots, y^k \notin X^I\right] \geq 1 - \left(1 - \frac{\epsilon}{2}\right)^k > 1 - \epsilon.$$

Note that if there exists $z \in X^I$ which is a subset of S then for every x, the two elements x and $x \vee z$ can only differ on a subset of S, and thus

$$I^b_S(f) \geq \Pr_x[f(x \vee z) = b] > 1 - \epsilon.$$

Now we turn our attention to Condition II. In this case, we shall prove that $\Pr_S[I^b_S[f] < 1 - \epsilon] \leq \epsilon$. Indeed,

$$\begin{aligned}\Pr_S[I^b_S[f] < 1 - \epsilon] &\leq \Pr_{y^1,\ldots,y^k}\left[\Pr_x[\exists i \in [k],\ f(x \vee y^i) = b] < 1 - \epsilon\right] \\ &= \Pr_{y^1,\ldots,y^k}\left[\Pr_x[\forall i \in [k],\ f(x \vee y^i) \neq b] \geq \epsilon\right]. \qquad (1)\end{aligned}$$

To bound the last probability, for $x \in \{0,1\}^n$ let E_x denote the event that for every $i \in [k]$, $f(x \vee y^i) \neq b$. Then

$$\Pr_x[E_x] \leq \Pr_x[x \notin X^{II}] + \Pr_x[E_x \ \wedge\ x \in X^{II}] \leq \frac{\epsilon}{2} + \Pr_x[E_x \mid x \in X^{II}].$$

Plugging this into (1), we get

$$\Pr_S[I^b_S[f] < 1 - \epsilon] \leq \Pr_{y^1,\ldots,y^k}[\Pr_x[E_x] \geq \epsilon] \leq \Pr_{y^1,\ldots,y^k}\left[\Pr_x[E_x \mid x \in X^{II}] \geq \frac{\epsilon}{2}\right] \leq \frac{1}{(\epsilon/2)} \Pr_{y^1,\ldots,y^k,x}[E_x \mid x \in X^{II}].$$

Since $k \approx \frac{10 \log \frac{1}{\epsilon}}{\epsilon}$,

$$\Pr_{x,y^1,\ldots,y^k}[E_x \mid x \in X^{II}] \leq (1 - \epsilon)^k \leq \frac{\epsilon^2}{2},$$

showing that

$$\Pr_S[I^b_S[f] < 1 - \epsilon] \leq \frac{1}{(\epsilon/2)} \cdot \frac{\epsilon^2}{2} \leq \epsilon.$$ ◀

2.2 Small Bias Case

To handle the small bias case for the sake of proving Theorem 1, one can simply repeat the argument of Kahn et al. [10], i.e. iteratively select influential variables and set them to the value that increases the probability of success. However, for the purposes of our results in the multi-round setting, we will need to prove a stronger result, which states that even if the coalition is selected *randomly*, there is a nontrivial chance of succeeding in influencing the outcome.

▶ **Lemma 12.** *Let $n \in \mathbb{N}$, $\gamma \in (0, 1/2)$ and $m \leq n$. Let $f\colon (\{0,1\}^n, \mu) \to \{0,1\}$, where μ is a product distribution such that for all i, $1/n < \alpha \leq \mathbb{E}[x_i] \leq 1/2$. Assume $m > \frac{n \log 1/\alpha}{2\gamma \log n}$. If $\mathbb{E}[f] \geq \gamma$ then*

$$\Pr_{S \subseteq [n]\colon |S|=m} \left[I_S^1[f] \geq 1 - \gamma\right] > \frac{1}{2} \left(\frac{m}{4n \log 1/\alpha}\right)^{2^{\frac{80n \log 1/\alpha}{m\gamma}}}.$$

Proof. The result is proved in [12] for constant α. The general case follows by representing a μ_p distributed variable as an AND of t variables that are distributed according μ_c, where $c \approx 1/2$ and $p = c^t$. The complete details appear in the full version of the paper. ◀

2.3 Finishing the Proof: Combining the Two Cases

We are ready to finish the proof of Theorem 1. Let $A := \{i : \Pr_{x \sim \mu}[x_i = 1] \in (\alpha_0, \frac{1}{2}]\}$, and recall that $\alpha_0 = \frac{1}{\log n}$. For every $y \in \{0,1\}^A$, define $f_y\colon \{0,1\}^{[n]\setminus A} \to \{0,1\}$ as $f_y(z) := f(y,z)$. By Proposition 10, for every $y \in \{0,1\}^A$, there exists $b := b_y \in \{0,1\}$ such that

$$\Pr_{S \sim \mu_{[n]\setminus A}^{(k)}} \left[I_S^b[f_y] > 1 - \frac{\epsilon}{2}\right] > 1 - \frac{\epsilon}{2},$$

where $k = O\left(\frac{\log(1/\epsilon)}{\epsilon}\right)$. Moreover, since every variable i in $[n]\setminus A$ satisfies $\mathbb{E}[x_i] \leq \alpha_0 = 1/\log n$, Chernoff's bound gives,

$$\Pr_{S \sim \mu_{[n]\setminus A}^{(k)}} \left[|S| \geq \frac{C \log(1/\epsilon) n}{\epsilon \log n}\right] \leq \exp\left(-\Omega\left(\frac{\log(1/\epsilon) n}{\epsilon \log n}\right)\right) \leq \frac{\epsilon}{2},$$

for some constant $C > 0$. Therefore,

$$\Pr_{S \sim \mu_{[n]\setminus A}^{(k)}} \left[I_S^b[f_y] > 1 - \frac{\epsilon}{2} \text{ and } |S| \leq \frac{C \log(1/\epsilon) n}{\epsilon \log n}\right] > 1 - \epsilon.$$

It follows that

$$\mathop{\mathbb{E}}_{S \sim \mu_{[n]\setminus A}^{(k)}} \left[\Pr_{y,b}\left[I_S^b[f_y] \geq 1 - \frac{\epsilon}{2}\right]\right] > \frac{1-\epsilon}{2} \geq \frac{1}{4},$$

assuming without loss of generality that $\epsilon \leq 1/2$. Hence, there exists a fixed $b_0 \in \{0,1\}$ and a set S, satisfying $|S| \leq \frac{C \log(1/\epsilon) n}{\epsilon \log n}$ and

$$\Pr_y \left[I_S^{b_0}[f_y] \geq 1 - \frac{\epsilon}{2}\right] \geq \frac{1}{4}.$$

Now, define $h\colon \{0,1\}^n \to \{0,1\}$ as $h(y) = 1$ if and only if $I_S^{b_0}[f_{y|_A}] \geq 1 - \epsilon/2$. Note that, h depends only on A variables. The above inequality asserts that $\mathbb{E}[h] \geq \frac{1}{4}$. Since, A contains only small bias variables, we may apply Lemma 12. Namely, there is $m = O(\frac{n \log\log n}{\epsilon \log n})$ such that

$$\Pr_{T \subseteq [n]\colon |T|=m} \left[I_T^1[h] \geq 1 - \frac{\epsilon}{2}\right] > 0.$$

Thus, there exists a coalition $T \subseteq A$ of size $O(\frac{n \log\log n}{\epsilon \log n})$ of players that can bias h towards 1. In other words, T can bias y towards cases where S is able to bias f_y towards b_0. As a result,

$$I_{S \cup T}^{b_0}[f] \geq \left(1 - \frac{\epsilon}{2}\right)\left(1 - \frac{\epsilon}{2}\right) > 1 - \epsilon.$$

Moreover, $|S \cup T| = O\left(\frac{n \log\log n}{\epsilon \log n} + \frac{\log(1/\epsilon) n}{\epsilon \log n}\right)$, as desired.

3 The Larger Range: Proof of Theorem 7

As outlined in the introduction, there are certain obstacles to generalizing Theorem 1 to larger ranges. In particular, the fact that the set $\mathcal{E}$ of all the points on which the coalition fails in Theorem 1 is of small measure does not seem to be a sufficiently strong condition for an induction to go through. We will need to prove a strengthening of Theorem 1 which shows that not only is $\mathcal{E}$ of small measure, but it is also small if it is measured via the boosted distributions introduced in Definition 9. This leads to a more general definition of influence.

▶ **Definition 13** (Boosted influence towards value)**.** *Let $\mathcal{R}$ be an arbitrary set. For a function $f\colon \{0,1\}^n \to \mathcal{R}$ and $b \in \mathcal{R}$, define $I_S^{b,t}(f) = \Pr_{x\sim\mu^{(t)}}[b \in f(B_S(x))]$.*

Note that $I_S^b(f) = I_S^{b,1}(f)$, as $\mu^{(1)} = \mu$.

The following lemma generalizes Lemma 11, as we spell out in its corollary.

▶ **Lemma 14.** *Consider $f\colon \{0,1\}^n \to \mathcal{R}$, let $t \in \mathbb{N}$, and let $S \sim \mu^{(k)}$, where $k = \frac{10t}{\delta}\log\frac{t}{\epsilon}$. Let $b \in \mathcal{R}$. We have $\Pr_S[\forall \ell \le t,\ I_S^{b,\ell}(f) \ge 1-\epsilon] \ge 1-\epsilon$, if any of the following two cases hold:*

- *Case I: For some $s \le t$, $\Pr_{u\sim\mu^{(s)}}[\Pr_{v\sim\mu^{(t)}}[f(u\vee v)=b] \ge 1-\epsilon/2] \ge \delta$.*
- *Case II: For every $s \le t$, $\Pr_{u\sim\mu^{(s)}}[\Pr_{v\sim\mu^{(t)}}[f(u\vee v)=b] \ge \delta] \ge 1-\frac{\epsilon}{2}$.*

Proof. The complete proof can be found in the full version of the paper. ◀

We can now state the main result of this section. The failure output † allows the inductive proof of Theorem 15, as well as our multi-round result, Theorem 17, to go through, as we explain in Section 4.

▶ **Theorem 15.** *Let $f\colon \{0,1\}^n \to \{0,1\}^m \cup \{\dagger\}$, and suppose that $\{0,1\}^n$ is endowed with a probability measure μ. Let t be a positive integer, and let $S \sim \mu^{(k)}$, where $k = k(m,t,\epsilon) = O(tm^3\epsilon^{-2}\log\frac{tm}{\epsilon})$. If $\Pr_{\mu^{(\ell)}}[\dagger] < \frac{\epsilon^4}{2^{16}}$ for every $\ell \le 2t$, then there exists a value $b \in \{0,1\}^m$ such that $\Pr_S\left[\forall \ell \le t,\ I_S^{b,\ell}(f) \ge 1-\epsilon\right] \ge 1-\epsilon$.*

Proof. The complete proof can be found in the full version of the paper. ◀

Theorem 7 follows from Theorem 15 using an argument very similar to that in Section 2.3, as we show in the full version of the paper.

4 Multi-Round Protocols: Proof of Theorem 2

In this section we will prove Theorem 2, showing that even in the multi-round setting, there are no protocols that are resilient against all coalitions of size $o(n)$. As described in the introduction, here at every round, first the players who are not in the coalition broadcast their random messages, and then the players in the coalition decide and broadcast their messages in an adversarial manner. The outcome is decided by a function $f\colon (\{0,1\}^n)^r \to \{0,1\}$.

To be more formal, let $\mu = \mu_1 \times \cdots \times \mu_r$ be a product distribution over $\{0,1\}^{rn} \equiv (\{0,1\}^n)^r$, where each μ_i is a product distribution over $\{0,1\}^n$. An (n,r) *coin-flipping protocol* is simply a map $f\colon (\{0,1\}^n)^r \to \{0,1\}$. Such a protocol is executed in r rounds. In the presence of a coalition $B \subseteq [n]$ of bad players, the protocol operates as follows. In round i, the players in $[n]\backslash B$ select $\alpha^i \in \{0,1\}^{[n]\backslash B}$ according to $\mu_i|_{[n]\backslash B}$. Then, the bad players B choose their values depending on $\alpha^1,\ldots,\alpha^i$. Formally, an (n,r)-strategy for a set $B \subseteq [n]$ is a sequence $\pi = (\pi_1,\ldots,\pi_r)$ of functions where $\pi_i\colon (\{0,1\}^{[n]\backslash B})^i \to \{0,1\}^B$. The function π_i describes the choice of bits the bad players make in the i-th round based on the broadcasted bits of the good players in the first i rounds.

▶ **Definition 16.** *Let $f\colon (\{0,1\}^n)^r \to \{0,1\}$ be an (n,r) coin-flipping protocol, and let μ be a product distribution on $(\{0,1\}^n)^r$. Given a Boolean value $b \in \{0,1\}$, a set $B \subseteq [n]$, and an (n,r)-strategy π for the bad players B,*

- *$I^b_{\pi,B}(f)$ is the probability that f outputs b given that the bad players B follow π.*
- *$I^b_B(f) := \sup_\pi\{I^b_{\pi,B}(f)\}$ is the influence of B on f towards b.*

Our goal is to show that there exists a coalition B of size $o(n)$ such that $I^b_B(f) \geq 1-\epsilon$ for some $b \in \{0,1\}$. For the moment, let us assume that we have only two rounds, and let $f(x,y)$ denote the protocol, where $x,y \in \{0,1\}^n$ correspond to the inputs in the first and the second round respectively. Let us also denote $f_x(y) \colon = f(x,y)$.

Russell et al. [12] proof of the uniform case. Pick $b \in \{0,1\}$ such that $\Pr[f(x,y)=b] \geq \frac{1}{2}$. Let $\mathcal{A}$ be the set of all $x \in \{0,1\}^n$ that satisfy $\Pr_y[f_x(y)=b] \geq \frac{1}{4}$, and note that $\Pr_x[x \in \mathcal{A}] \geq \frac{1}{4}$. By Lemma 12 of Russell et al. [12], for every $x \in \mathcal{A}$, a random coalition S can bias f_x towards b, with a probability δ that is not too small. Since S is chosen randomly and independently of x, it follows that there exists a fixed coalition S_0 that can bias f_x for at least a δ fraction of $x \in \mathcal{A}$, and thus for at least a $\frac{\delta}{4}$ fraction of $\{0,1\}^n$. Let $\mathcal{A}' \subseteq \mathcal{A} \subseteq \{0,1\}^n$ denote the set of such x. If $x \in \mathcal{A}'$, the coalition S_0 is able to bias the protocol by only interfering in the second round. The set $\mathcal{A}'$ is of measure at least $\frac{\delta}{4}$, which is not too small. Thus, applying Lemma 12 again, we can find another coalition $T_0 = o(n)$ which can modify most x's to fall in $\mathcal{A}'$. Now we can form the desired coalition $B = T_0 \cup S_0$: In the first round, the players in T_0 try to modify x into an element in $\mathcal{A}'$, and if they succeed, in the second round, the players in S_0 interfere to change the outcome of the protocol into b. This argument easily generalizes to more rounds.

We point out that it was crucial for the above argument, that the distribution of S in Proposition 10 is independent of f.

What fails for the general product distributions. Consider $f(x,y)$ over $\mu = \mu_1 \times \mu_2$, where μ_1 is *highly biased*, and μ_2 is the *uniform distribution*. Similar to the previous paragraph, we can find a set $\mathcal{A}' \subseteq \{0,1\}^n$, a value $b \in \{0,1\}$, and a small coalition S_0 such that $\Pr_{x\sim\mu_1}[x \in \mathcal{A}'] \geq \frac{\delta}{4}$, and moreover for every $x \in \mathcal{A}'$, the coalition S_0 is able to influence f towards b by interfering only in the second round. Now, if we are to follow the argument of Russell et al., we would like to find a set T_0 of players to add to the coalition such that, with high probability, T_0 is able to modify a random $x \sim \mu_1$ into an element in $\mathcal{A}'$. We could then conclude that $B = S_0 \cup T_0$ can bias f towards b.

Unfortunately, Proposition 10, the highly-biased counterpart of Lemma 12, only guarantees the existence of a small coalition T_0 which *either* modifies a random $x \sim \mu$ into being in $\mathcal{A}'$ *or* modifies a random $x \sim \mu$ into *not* being in $\mathcal{A}'$; in the latter case, the coalition T_0 is useless. As Example 3 shows, this is not just a caveat of the proof of the proposition. To be more concrete, suppose μ_1 is the $\frac{1}{n}$-biased distribution, and $\mathcal{A}'$ consists only of the single element $x = \vec{0}$. Even though $\Pr[x \sim \mathcal{A}] \geq \frac{1}{4}$, there is no coalition of size $o(n)$ which can, with high probability, modify a random $x \sim \mu_1$ into an element in $\mathcal{A}'$. On the other hand, even a single player can modify every x into an element outside $\mathcal{A}'$, but this is not helpful for our purposes, as the elements outside $\mathcal{A}'$ are the elements that S_0 cannot handle.

How to overcome the problem. Consider the same setting as in the previous paragraph. We know that for every x, a random coalition S of size $o(n)$ succeeds in influencing f_x towards one of the outputs, with probability at least δ, where δ is not too small. Instead of

picking one S_0, we select a collection of coalitions that cover almost all x's. More precisely, we find $S_1, \ldots, S_M$ and $b_1 \ldots, b_M$, where $M = O_\delta(1)$, such that apart from a small set of exceptions $\mathcal{E} \subseteq \{0,1\}^n$, every f_x can be biased towards some b_i using the coalition S_i.

Let $h\colon \{0,1\}^n \to \{1,\ldots,M\} \cup \{\dagger\}$ be defined as follows: If $x \in \mathcal{E}$, then $h(x) = \dagger$, and otherwise $h(x)$ is equal to some i such that S_i can bias f_x towards b_i. This brings us to the non-Boolean range case, which was analyzed in Section 3. We can apply Theorem 15 to find a coalition T that can influence h towards one of the values in $j \in \{1,\ldots,M\}$. Now $B = T \cup S_j$ will be our desired coalition. With high probability, in the first round the players in T can successfully modify a random element x into an element x' with $h(x') = j$, and then in the second round, the players in S_j can modify x' to bias the outcome towards b_j. This is the main new idea used below to resolve the multi-round setting over arbitrary distributions.

Theorem 2 is a consequence of the following more elaborate theorem which states that for sufficiently large n, and $r \leq \log^\star n/5$, no (n,r) protocol over an arbitrary product distribution is resilient against coalitions of $m = o(n)$ bad players.

▶ **Theorem 17.** *For every $\epsilon > 0$, and integers $n > 0$, and $r < \log^\star n/5$, there exists $\delta = \Omega(\frac{1}{\log(1/\epsilon)^r n})$, and $m = o(n)$ such that the following holds. For every $f\colon (\{0,1\}^n)^r \to \{0,1\}$ over a product distribution μ, there exists $b \in \{0,1\}$, such that the corresponding r-round protocol satisfies $\Pr_{S\sim\nu}[I^b_S(f) \geq 1-\epsilon] \geq \delta_r$, where ν is a distribution on $\binom{[n]}{m}$ that depends only μ but not on f. To be more precise, one can take $m = O_\epsilon\left(\frac{n\cdot r\cdot 4^r}{\log^{(4r)} n}\right) = O_\epsilon\left(\frac{n(\log^\star n)^2}{\log^{(4r)} n}\right)$.*

Proof. The complete proof can be found in the full version of the paper. ◀

5 Concluding Remarks and Open Problems

- Perhaps the most interesting next step is proving limitations for resilience of protocols where players may send longer messages. As was discussed below Conjecture 6, it is conjectured that even when the players are allowed to broadcast arbitrarily long messages, only resilience against coalitions of size $o(n)$ is possible. This question has also been studied in the multi-round setting [12, 13, 8]. In this case, if the players are allowed $\log n$-bit messages, we know of $(\log^\star n + O(1))$-round protocols resilient against coalitions of size $(1/2-\epsilon)n$ [13, 8]. On the other hand, Russell et al. [12] showed that $\Omega(\log^\star n)$ rounds are necessary if we have the added restriction that in the i-th round the players are allowed messages of length $(\log^{(2i-1)} n)^{1-o(1)}$. Strengthening this impossibility result to messages of length $\Omega(\log n)$ is another interesting problem that remains open.
- The key qualitative point of Theorems 1 and 2 is that there always exists a coalition of size $o(n)$ that can bias the outcome of the protocol towards a particular value. Interestingly, we are not aware of a simpler proof of this weaker qualitative statement even in the case of the uniform measure. The proof techniques introduced in this paper for the highly biased coordinates are more combinatorial and probabilistic in nature; however, the less biased coordinates are ultimately handled by the Fourier-analytic proof of [10]. These Fourier analytic arguments are *hard* in nature, in the sense that their purpose is to give effective bounds. It would be interesting to find more intuitive combinatorial proofs for these statements, potentially at the cost of obtaining less effective bounds, or by appealing to *soft analytic tools* such as compactness, at the cost of obtaining no quantitative bounds. We refer the reader to Terence Tao's blog post [14] for a discussion about hard and soft analysis.
- Over the uniform distribution, Kahn et al. [10] proved that there exists no Boolean function that is ϵ-resilient against coalitions of size $\omega_\epsilon\left(\frac{n}{\log n}\right)$. In this work we show that a similar bound of $\omega_\epsilon\left(\frac{n\log\log n}{\log n}\right)$ on resilience holds over arbitrary product distributions.

A natural question is whether the $\log \log n$ in our bound necessary. However, even in the uniform setting there is work left to be done. Here, the best known constructions guarantee resilience against coalitions of size $O(\frac{n}{\log^2 n})$ [11, 1], which is a factor of $\log n$ off from the impossibility result of Kahn, Kalai, and Linial.

References

1 Miklós Ajtai and Nathan Linial. The influence of large coalitions. *Combinatorica*, 13(2):129–145, 1993.

2 Michael Ben-Or and Nathan Linial. Collective coin flipping. *Advances in Computing Research*, 5:91–115, 1989.

3 Michael Ben-Or, Nathan Linial, and Michael Saks. *Collective coin flipping and other models of imperfect randomness*. IBM Thomas J. Watson Research Division, 1989.

4 Jean Bourgain, Jeff Kahn, Gil Kalai, Yitzhak Katznelson, and Nathan Linial. The influence of variables in product spaces. *Israel Journal of Mathematics*, 77(1–2):55–64, 1992.

5 Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functions. *Annals of Mathematics, to appear*, 2016. Preliminary version in STOC 2016.

6 Benny Chor and Cynthia Dwork. Randomization in Byzantine Agreement. *Advances in Computing Research*, 5:443–497, 1989.

7 Yevgeniy Dodis. Fault-tolerant leader election and collective coin-flipping in the full information model. *Survey*, 2006.

8 Uriel Feige. Noncryptographic Selection Protocols. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 142. IEEE Computer Society, 1999.

9 Ehud Friedgut. Influences in Product Spaces: KKL and BKKKL Revisited. *Combinatorics, Probability and Computing*, 13(1):17–29, 2004.

10 Jeff Kahn, Gil Kalai, and Nathan Linial. The influence of variables on Boolean functions. In *Proceedings of the 29th annual FOCS*, pages 68–80, 1988.

11 Raghu Meka. Explicit resilient functions matching Ajtai-Linial. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1132–1148. SIAM, 2017.

12 Alexander Russell, Michael Saks, and David Zuckerman. Lower bounds for leader election and collective coin-flipping in the perfect information model. *SIAM Journal on Computing*, 31(6):1645–1662, 2002.

13 Alexander Russell and David Zuckerman. Perfect information leader election in log* n+ O (1) rounds. *Journal of Computer and System Sciences*, 63(4):612–626, 2001.

14 Terence Tao. Soft analysis, hard analysis, and the finite convergence principle. URL: `https://terrytao.wordpress.com/2007/05/23/soft-analysis-hard-analysis-and-the-finite-convergence-principle/`, 2007. Accessed 10 Feb 2019.

Covering Vectors by Spaces in Perturbed Graphic Matroids and Their Duals

Fedor V. Fomin
Department of Informatics, University of Bergen, Norway
fomin@ii.uib.no

Petr A. Golovach
Department of Informatics, University of Bergen, Norway
Petr.Golovach@uib.no

Daniel Lokshtanov
Department of Computer Science, University of California Santa Barbara, USA
daniello@ucsb.edu

Saket Saurabh
The Institute of Mathematical Sciences, HBNI, Chennai, India
saket@imsc.res.in

Meirav Zehavi
Ben-Gurion University, Israel
meiravze@bgu.ac.il

Abstract

Perturbed graphic matroids are binary matroids that can be obtained from a graphic matroid by adding a noise of small rank. More precisely, an r-rank perturbed graphic matroid M is a binary matroid that can be represented in the form $I + P$, where I is the incidence matrix of some graph and P is a binary matrix of rank at most r. Such matroids naturally appear in a number of theoretical and applied settings. The main motivation behind our work is an attempt to understand which parameterized algorithms for various problems on graphs could be lifted to perturbed graphic matroids.

We study the parameterized complexity of a natural generalization (for matroids) of the following fundamental problems on graphs: Steiner Tree and Multiway Cut. In this generalization, called the Space Cover problem, we are given a binary matroid M with a ground set E, a set of *terminals* $T \subseteq E$, and a non-negative integer k. The task is to decide whether T can be spanned by a subset of $E \setminus T$ of size at most k.

We prove that on graphic matroid perturbations, for every fixed r, Space Cover is fixed-parameter tractable parameterized by k. On the other hand, the problem becomes W[1]-hard when parameterized by $r + k + |T|$ and it is NP-complete for $r \leq 2$ and $|T| \leq 2$.

On cographic matroids, that are the duals of graphic matroids, Space Cover generalizes another fundamental and well-studied problem, namely Multiway Cut. We show that on the duals of perturbed graphic matroids the Space Cover problem is fixed-parameter tractable parameterized by $r + k$.

2012 ACM Subject Classification Mathematics of computing → Combinatorial algorithms; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Binary matroids, perturbed graphic matroids, spanning set, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.59

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `http://arxiv.org/abs/1902.06957`.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 59; pp. 59:1–59:13

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

Funding The research leading to these results has received funding from the Research Council of Norway via the projects "CLASSIS" and "MULTIVAL".

Acknowledgements We thank Jim Geelen for valuable insights regarding matroid minors.

1 Introduction

In this paper we develop parameterized algorithms on low-rank perturbations of graphic matroids and their duals. These matroids and their matrices naturally appear in various settings. For example, in the emerging Matroid Minors Project of Geelen, Gerards, and Whittle [15], perturbed matroids play a significant role in the characterization of proper minor-closed classes of binary matroids. More precisely, for each proper minor-closed class $\mathcal{M}$ of binary matroids, there exists a nonnegative integer r such that every sufficiently highly connected matroid $M \in \mathcal{M}$, is either a perturbation of graphic or cographic matroid. In other words, there exist matrices $I, P \in \mathrm{GF}(2)^{\ell \times n}$ such that I is the incidence matrix of a graph, the rank of P is at most r, and either M or its dual M^* is represented by $I + P$. Another example of closely related concept is the robust Principal Component Analysis (PCA), a popular approach to robust subspace learning and tracking by decomposing the data matrix into low-rank and sparse matrices. Here data matrix M is assumed to be a superposition of a low-rank perturbation component P and a sparse component I, that is, $M = I + P$. See Candès et al. [4], Wright et al. [24], and Chandrasekaran et al. [5] for further references on robust PCA. In particular, one of the well-studied, see e.g. [22, 26], of the variants of robust PCA is when the structure of the sparse matrix I is imposed from the structure of some graph. Perturbed matroids also come naturally in the settings when a structural input is corrupted by a noise. In graph algorithms, one of the questions studied in the literature about corrupted inputs is – what happens to special graph classes when they are perturbed adversarially? For example, Magen and Moharrami [19], and Bansal, Reichman, Umboh [2], studied approximation algorithms on noisy minor-free graphs, which are the graphs obtained from minor-free graphs by corrupting a fraction of edges and vertices.

Our results. We work with the following classes of binary matroids. A binary matroid M such that M can be represented in the form $I + P$, where I is the incidence matrix of some graph and P is a binary matrix of rank at most r, is called the *r-rank perturbed graphic matroid*. Similarly, when the dual matroid M^* can be represented as $I + P$ for some incidence matrix I and r-rank matrix P, we refer to M as to an *r-rank perturbed cographic matroid*.

In this paper we study parameterized complexity on binary perturbed matroids of the following generic problem. Let us remind that in a matroid M, a set F spans T, denoted by $T \subseteq \mathrm{span}(F)$, if the sets F and $T \cup F$ are of the same rank.

Space Cover
Input: A binary matroid M with a ground set E, a set of *terminals* $T \subseteq E$, and a non-negative integer k.
Question: Is there a set $F \subseteq E \setminus T$ with $|F| \leq k$ such that $T \subseteq \mathrm{span}(F)$?

In other words, Space Cover is the problem of covering a given set of vectors T over $\mathrm{GF}(2)$ by a minimum-dimension subspace of the space generated by vectors from $E \setminus T$. Space Cover encompasses various problems arising in different domains, such as coding theory, machine learning, and graph algorithms. For example, Space Cover is a natural generalization of Matroid Girth, the problem of finding a minimum set of dependent

elements in a matroid. Matroid Girth can be reduced to Space Cover by computing for each element t of M a minimum set of elements of the remaining part of the matroid that covers $T = \{t\}$.

On graphs (equivalently, special classes of binary matroids, namely graphic and cographic matroids), Space Cover generalizes well-studied optimization problems Steiner Tree and Multiway Cut. Various algorithmic techniques were developed for these problems, see e.g. [7], and it is very interesting to see which of these techniques, if any, can be lifted to matroids.

We obtain the following results about the complexity of Space Cover on r-rank perturbed matroids. (In all these results we assume that representation of r-rank perturbed matroid in the form $I + P$ is given.)

- Our first main algorithmic result (Theorem 1) states the following: On r-rank perturbed graphic matroids, for every fixed r, Space Cover is fixed-parameter tractable (FPT) when parameterized by k.
- We also show that a "weaker" parameterization makes the problem intractable. More precisely, we prove that on r-rank perturbed graphic matroids, Space Cover is W[1]-hard when parameterized by $r + k + |T|$ and the problem is NP-complete for $r \leq 2$ and $|T| \leq 2$ (see Theorems 3 and 4 respectively of [14]).
- Our second main algorithmic result (Theorem 2) concerns r-rank perturbed cographic matroids. This theorem states that Space Cover is FPT on r-rank perturbed cographic matroids when parameterized by $r + k$. We find it a bit surprising that the parameterized complexity of Space Cover is different on r-rank perturbed graphic and cographic matroids.

Previous work. Geelen and Kapadia [17] studied the problem of computing the girth of a binary r-rank perturbed matroid. (The girth of a matroid is the length of its shortest circuit.) Geelen and Kapadia have proved that the girth of an r-rank perturbed matroid is fixed-parameter tractable being parameterized by r. Let us note that while Space Cover generalizes Matroid Girth, our results are incomparable. In our FPT result for r-rank perturbed graphic matroids the parameter is k while the parameter r should be fixed. As our complexity lower bounds show, the requirement that r should be fixed and that k should be the parameter are, most likely, unavoidable. For binary matroids, Matroid Girth has several equivalent formulations. For example, it is equivalent to the Minimum Distance problem from coding theory, which asks for a minimum dependent set of columns in a matrix over GF(2). The complexity of this problem was open until 1997, when Vardy showed it to be NP-complete [23]. On the other hand, Geelen, Gerards and Whittle in [16] conjecture that for any proper minor-closed class $\mathcal{M}$ of binary matroids, there is a polynomial-time algorithm for computing the girth of matroids in $\mathcal{M}$. The parameterized version of the problem, namely Even Set, asks whether there is a dependent set $F \subseteq X$ of size at most k. The parameterized complexity of Even Set was a long-standing open question in the area, see e.g. [10], whose complexity was resolved only recently [3].

Space Cover on graphic and cographic matroids is a generalization of Steiner Tree and Multiway Cut, two very well-studied problems on graphs. By the classical result of Dreyfus and Wagner [12], Steiner Tree is fixed-parameter tractable (FPT) parameterized by the number of terminals T. Similar approach can be used to show that Space Cover is FPT on graphic matroids. On cographic matroids Space Cover is equivalent to the Restricted Edge-Subset Feedback Edge Set introduced by Xiao and Nagamochi [25] who also showed that the problem is FPT parameterized by k. Due to its connection to Multiway Cut, the NP-completeness result of Dahlhaus et al. [8] for Multiway Cut with three terminals implies that Space Cover is NP-hard even if $|T| = 3$ on cographic

matroids. Fomin et al. in [13] extended the results for Space Cover on graphic and cographic matroids to a more general class of binary matroids, namely, regular matroids, by providing an algorithm of running time $2^{\mathcal{O}(k)} \cdot ||M||^{\mathcal{O}(1)}$. While the class of regular matroids is a proper minor-class of binary matroids, this class of matroids is incomparable to the class of perturbed matroids. It is also known that Space Cover is hard on general class of binary matroids: By the result of Downey et al. [11], Space Cover is W[1]-hard on binary matroids when parameterized by k even if restricted to the inputs with one terminal.

Organization of the paper. Due to space constraints, we only sketch the proofs of our main algorithmic results (Theorems 1 and 2) in Section 2. The detailed proofs of these theorems and our algorithmic lower bounds (Theorems 3 and 4) are given in the full version of the paper [14]. We conclude in Section 3 by stating some open problems.

2 Overview of Algorithmic Theorems

In this section, we give short descriptions of both of our algorithmic results. For standard graph and matroid-related terms, we refer to the books by Diestel [9] and Oxley [21]. We also give the formal definitions of graph and matroid-related terms in Section 3 of [14].

2.1 Perturbed Graphic Matroids

In this section, we give an overview of the proof of the first main result of the paper. The detailed proof is given in Section 4 of [14] version of the paper.

In this case, r-rank perturbed matroid M is represented by the perturbed incidence matrix $I(G)$ of a (multi) graph G. Formally we define the following problem.

Space Cover on Perturbed Graphic Matroid (Space Cover on PGM)
Input: A (multi) graph G with n vertices and m edges, an $(n \times m)$-matrix P over $GF(2)$ with $\text{rank}(P) \leq r$, a set of *terminals* $T \subseteq E$ where E is the set of columns of the matrix $A = I(G) + P$, and a non-negative integer k.
Question: Is there a set $F \subseteq E \setminus T$ with $|F| \leq k$ such that $T \subseteq \text{span}(F)$ in the binary matroid M represented by A?

▶ **Theorem 1.** *For any fixed constant r,* Space Cover on PGM *is solvable in time $k^{\mathcal{O}(k)} \cdot (n+m)^{\mathcal{O}(1)}$. In particular,* Space Cover on PGM *is* FPT *when parameterized by k whenever r is a constant.*

We underline that r is a constant here, that is, the constants hidden behind the big-O notation in the running time depend on r.

Before proceeding with the overview, it is useful to discuss how Space Cover on PGM is solvable when $r = 0$, i.e. on graphic matroids and what are the main challenges for solving the problem for $r > 0$. On graphic matroids Space Cover corresponds to the following problem. Given a set of terminal edges $T = \{e_1, e_2, \ldots, e_s\}$, we want to find a set of at most k edges $F \subseteq E \setminus T$ such that for every e_i, graph $G[F \cup e_i]$ has a cycle containing e_i. This can be seen as a variant of the Steiner Tree, and more generally, of the Steiner Forest problem. Here we are given a graph G, a collection of pairs of distinct non-adjacent terminal vertices $\{x_1, y_1\}, \ldots, \{x_s, y_s\}$ of G, and a non-negative integer k. The task is to decide whether there is a set $F \subseteq E(G)$ with $|F| \leq k$ such that for each $i \in \{1, \ldots, s\}$, graph $G[F]$ (which we can be assumed to be a forest) contains an (x_i, y_i)-path. The special case when $x_1 = x_2 = \cdots = x_s$, i.e. when edge set F is a tree spanning all demand vertices, is the Steiner Tree problem. To see that Steiner Forest is a special case of Space Cover,

we construct the following graph: For each $i \in \{1, \ldots, s\}$, we add a new edge $x_i y_i$ to G. Denote by G' the obtained graph and let T be the set of added edges and let $M(G')$ be the graphic matroid associated with G'. Then a set of edges $F \subseteq E(G)$ forms a graph containing all (x_i, y_i)-paths if and only if $T \subseteq \mathrm{span}(F)$ in $M(G')$.

Similar to Steiner Tree, Steiner Forest is fixed-parameter tractable parameterized by the number of terminals. This can be shown by applying a dynamic programming algorithm similar to the classical algorithm of Dreyfus and Wagner [12]. Notice that by [14, Theorem 4]), Space Cover on PGM is NP-complete when restricted to the instances with $r \leq 2$ and $|T| \leq 2$. This shows that for our problem the parameterization just by the number of terminals $|T|$ will not work; it also indicates that for matroids we should try a different approach. To show that Steiner Forest is FPT parameterized by the size k of the forest F, one can use the following idea. Since the size of F is at most k, there are $2^{\mathcal{O}(k)}$ non-isomorphic forests, so we can guess the structure of F. In other words, we can guess a forest H on at most k edges such that the solution F to Steiner Forest is isomorphic to H. Thus for each guess of H, the task is reduced to the following constraint variant of Subgraph Isomorphism: For given graph G and forest H, decide whether G contains a forest isomorphic to H and spanning all terminal vertices of G in the prescribed way. This problem can be solved by combining a color coding technique of Alon, Zwick, and Yuster [1] with dynamic programming.

This is exactly the approach we want to push forward for $r > 0$. However in this case reduction to constraint Subgraph Isomorphism is way more difficult. First, while perturbation matrix P is of bounded rank, adding it to $I(G)$ can change an unbounded number of its elements. On the other hand, since the rank of perturbation matrix P is bounded, we know that matrix P contains only a small number of different columns. Thus while adding P to $I(G)$ changes many elements of $I(G)$, the variety of these changes is bounded. We exploit this in order to guess the structure of a solution. Second, for graphic matroids, the way a forest H should be mapped into G is very clear – for every terminal element t, adding t to the solution should create a cycle containing t. This defines the constraints how the edges of the guessed solution should be connected to terminal edges and allows us to reduce the problem to a constraint variant of Subgraph Isomorphism. For $r > 0$, adding P to $I(G)$ completely destroys this nice property of the solution. Interestingly, the bounded rank of perturbation still allows us to establish the constraints expressed as parities of vertex degrees of a small number of vertices in G, coloring of edges of G, and some additional mappings. As a result, by a sequence of reductions, we succeed in reducing the original problem to a version of constraint Subgraph Isomorphism. Due to the nature of constraints, the solution to this problem also requires new ideas on top of color coding and dynamic programming.

We proceed with an overview of the proof of Theorem 1. The proof consists of two main parts. The first part is an FPT-Turing reduction from Space Cover to the following version of Subgraph Isomorphism, which we call Pattern Cover.

Pattern Cover
Input: A (multi) graph G with n vertices and m edges, a non-negative integer t that is a fixed constant, a function $\ell_G : E(G) \to \{1, 2, \ldots, t\}$, a non-negative integer k, a forest H with k vertices, a function $\ell_H : E(H) \to \{1, 2, \ldots, t\}$, a set $U \subseteq V(H)$ and an injective function $f : U \to V(G)$.
Question: Is there an injective homomorphism $g : V(H) \cup E(H) \to V(G) \cup E(G)$ such that (i) for all $e \in E(H)$, it holds that $\ell_H(e) = \ell_G(g(e))$, and (ii) for all $v \in U$, it holds that $g(v) = f(v)$?

In other words, we give a reduction that for an input (G, P, T, k) of Space Cover on PGM in time $k^{\mathcal{O}(k)} \cdot (n+m)^{\mathcal{O}(1)}$ constructs $k^{\mathcal{O}(k)} \cdot (n+m)^{\mathcal{O}(1)}$ instances of Pattern Cover such that (G, P, T, k) is a yes-instance if and only if at least one of the instances of Pattern Cover is.

The second part of the proof is an algorithm for solving Pattern Cover in time $k^{\mathcal{O}(k)} \cdot (n+m)^{\mathcal{O}(1)}$. The combination of the two parts provides the proof of the theorem.

In what follows, we provide a brief description of the FPT-Turing reduction. The reduction is done by a sequence of steps. For simplicity, here we explain how to construct a reduction in time $2^{\mathcal{O}(k^2)} \cdot (n+m)^{\mathcal{O}(1)}$; in Section 4 of [14] we provide more precise arguments that allow to reduce the running time.

We start by bounding $|T|$ by k. In case the columns in T are not linearly independent, we let T' denote a basis of T, and else we denote $T' = T$. We remove the columns in $T \setminus T'$ from $I(G)$ and P, and let (G', P', T', k) denote the resulting instance. Clearly, (G, P, T, k) is a yes-instance if and only if (G', P', T', k) is a yes-instance. Moreover, given a set X of size t of linearly independent vectors, for some $t \in \mathbb{N}$, there does not exist any set Y of vectors of size smaller than t such that $X \subseteq \text{span}(Y)$. Thus, in case $|T'| > k$, the input instance is a no-instance. Therefore, from now onwards we implicitly assume that $|T| \leq k$. We use the term *solution* to refer to any set $F \subseteq E \setminus T$ with $|F| \leq k$ such that $T \subseteq \text{span}(F)$ in the binary matroid M represented by A.

We define $\text{disc}(P) = \{C^1, \ldots, C^t\}$ to be the *set* of the distinct vectors that correspond to the columns in $\{P^e : e \in E(G)\}$ (we index the columns of A, $I(G)$ and P by the edges of G). Since the rank of P is r, it is easy to see that it has at most 2^r different columns, thus $t \leq 2^r$. We say that an edge $e \in E(G)$ is of *type* i, $1 \leq i \leq t$, if $P^e = C^i$ (as vectors). Given an edge $e \in E(G)$, we let $\text{type}(e)$ denote its type. Given a set of edges $E' \subseteq E(G)$, we denote $\text{type}(E', i) = |\{e \in E' : \text{type}(e) = i\}| \mod 2$. Towards to constructing the reduction to Pattern Cover, we define $\ell_G \colon E(G) \to \{1, \ldots, t\}$ by setting $\ell_G(e) = \text{type}(e)$.

We proceed by identifying a small graph that we can guess, and which will guide us how to find a solution. Let F be an inclusion-wise minimal solution; note that the minimality of F implies that F is an independent set. Consider the graph $H = G[\text{edges}(F)]$. The crucial structural lemma that we use states that H is "almost" a forest. More precisely, we show that H has at most 2^t cycles. To see it, assume that H has at least $2^t + 1$ cycles. There are at most t edge types in H. Hence by the pigeonhole principle, there are distinct sets of edges C_1 and C_2 of H that compose cycles and such that $\text{type}(C_1, i) = \text{type}(C_2, i)$ for all $i \in \{1, \ldots, t\}$. Then for the symmetric difference $C = C_1 \triangle C_2$, we obtain that $\text{type}(C, i) = 0$ for $i \in \{1, \ldots, t\}$. Thus the sum of the columns of P corresponding to edges of C is the zero-vector. Notice that since C is the union of cycles of H, the sum of the columns of matrix $I(G)$ corresponding to its edges is also the zero-vector. Hence, the sum of the corresponding vectors of A is also zero; and thus the corresponding set of columns of A, $\{A^e \mid e \in C\} \subseteq F$ is not independent. But this contradicts the minimality of F.

Let $\mathcal{H}$ denote the set of all non-isomorphic graphs with at most k edges, at most 2^t cycles, and no isolated vertices. Thus (G, P, T, k) is a yes-instance of Space Cover on PGM if and only if (G, P, T, k) has a solution isomorphic to some $H \in \mathcal{H}$. It is possible to show that all non-isomorphic graphs in $\mathcal{H}$ can be enumerated within time $2^{\mathcal{O}(k)}$. Therefore, we may explicitly examine each graph $H \in \mathcal{H}$ and check whether we have a solution F with subgraph of G, $G[\text{edges}(F)]$, isomorphic to H. In other words, we are looking for an injective

homomorphism $g : V(H) \cup E(H) \to V(G) \cup E(G)$[1] such that $F = \{A^e \mid e \in g(E(H))\}$ is a solution. This is an FPT-Turing reduction which reduces in time $2^{\mathcal{O}(k)}$ the solution of the original problem to the solution of $2^{\mathcal{O}(k)}$ new problems. We will use a less formal term *guess* to refer to such type of reductions. So we guess graph H.

Next, we observe that we can guess the types of edges of H. Since H has at most k edges, there are at most $t^k = 2^{\mathcal{O}(k)}$ distinct functions $\ell_H \colon E(H) \to \{1, \ldots, t\}$. Then for each guess of function ℓ_H, we want to decide whether there is an injective homomorphism g such that $\ell_G(g(e)) = \ell_H(e)$ for every $e \in E(H)$ and such that the set of columns F of A corresponding to the image of g, which is $F = \{A^e \mid e \in g(E(H))\}$, is a solution.

By definition, if $F = \{A^e \mid e \in g(E(H))\}$ is a solution, then for each $W \in T$, there is $F_W \subseteq F$ such that

$$W = \sum_{e \in F_W} A^e. \tag{1}$$

(The summations here are modulo 2.) We denote by $E_W = g^{-1}(\mathrm{edges}(F_W))$ the edge subset of H corresponding to F_W. Then by (1),

$$W = \sum_{e \in g(E_W)} (I^e(G) + P^e) = \sum_{e \in g(E_W)} I^e(G) + \sum_{e \in g(E_W)} P^e.$$

Each column P^e is equal to vector $C^{\ell_H(e)}$ from partition $\mathrm{disc}(P)$. Thus

$$W = \sum_{e \in g(E_W)} I^e(G) + \sum_{e \in E_W} C^{\ell_H(e)}. \tag{2}$$

Let $W' = W + \sum_{e \in E_W} C^{\ell_H(e)}$. The rows of matrix $I(G)$ and thus the elements of W' are indexed by the vertices of G. For $v \in V(G)$, we denote by w_v the element of W' indexed by v. Note that w_v is either 0 or 1. Let $V_W = \{v \in V(G) \mid w_v = 1\}$. Observe that V_W is uniquely defined by the choice of W and E_W. The crucial insight, whose proof is given in [14, Section 4], is that (2) and, therefore, (1) holds if and only if g acts as a bijection between V_W and vertices of $H[E_W]$ of odd degrees. This is the most important part of the reduction; it allows to reduce the algebraic requirement that every terminal vector should be in the span of the solution to constraints in the form of bijections, which can be guessed efficiently.

We exploit this property for the next set of guesses. For each $W \in T$, we guess a set $E_W \subseteq E(H)$ and construct V_W as described above. Since $|T| \leq k$ and $|E(H)| \leq k$, we have at most 2^{k^2} possible choices of the sets E_W. Then we find the set $U_W \subseteq V(H[E_W])$ of vertices that have odd degrees in $H[E_W]$. If $|V_W| \neq |U_W|$, we discard the choice. Otherwise, we set $U = \cup_{W \in T} U_W$. Notice that if our guesses correspond to a (potential) solution F, we have that corresponding injective homomorphism g should map U to $V' = \cup_{W \in T} V_W$ bijectively and, moreover, g should act as bijection between each U_W and V_W. We make all possible guesses of a bijection $f \colon U \to U'$. Since $|U| \leq 2k$, we have at most $(2k)^{2k}$ possible choices. Then for each U and f, we are searching for an injective homomorphism $g : V(H) \cup E(H) \to V(G) \cup E(G)$ such that (i) for all $e \in E(H)$, $\ell_H(e) = \ell_G(g(e))$, and (ii) for each $v \in U$, $g(v) = f(v)$.

Now we are ready for the final step of our reduction. Recall that H in the statement of Pattern Cover is required to be a forest. The graph H that was guessed so far does not have this property, but it is "almost" a forest, that is, it has at most 2^t cycles. To fix it, we

[1] Since we handle *multi* graphs, we define the domain and image of g to include edge-sets.

guess a set of edges $S \subseteq E(H)$ of size at most 2^t such that the graph obtained from H by the deletion of S is a forest and set $H = H - S$. Since $|S| \leq 2^t$, and t is a constant depending on r only, we can make a polynomial number of guesses how solution g could map S to $E(G)$; we have at most $|E(G)|^{2^t} = m^{\mathcal{O}(1)}$ possibilities for such partial mappings. For each guess of mapping $h : S \to V(G)$, we modify U and f respectively. Namely, we set $U = U \cup V(H[S])$ and define $f(v) = h(v)$ for $v \in V(H[S])$ as prescribed by our choice of the mapping h of S.

This concludes the description of the construction of an instance of Pattern Cover. It is possible to show that (G, P, T, k) is a yes-instance of Space Cover on PGM if and only if for at least one of the described guesses of a forest H, functions ℓ_H, ℓ_G, set $U \subseteq V(H)$ and function $f : U \to V(G)$, the instance of Pattern Cover with these parameters is a yes-instance. Since the total number of guesses we make is , $2^{\mathcal{O}(k^2)} \cdot (n+m)^{\mathcal{O}(1)}$, our construction is the required FPT-Turing reduction.

In order to solve Pattern Cover, and to complete the proof of Theorem 1, we still have to solve Pattern Cover. This is done by a non-trivial application of the color coding technique combined with dynamic programming. We give all the details in [14, Section 4].

2.2 Duals of Perturbed Graphic Matroids

In this section, we give an overview of the proof of our second main result. The detailed proof of the theorem is given in [14, Section 5].

Formally, we define the following problem.

> Space Cover on Dual of Perturbed Graphic Matroid (Space Cover on Dual-PGM)
> **Input:** A (multi) graph G with n vertices and m edges, an $(n \times m)$-matrix P over $GF(2)$ with $\mathrm{rank}(P) \leq r$, a set of *terminals* $T \subseteq E$ where E is the set of columns of the matrix $A = I(G) + P$, and a non-negative integer k.
> **Question:** Is there a set $F \subseteq E \setminus T$ with $|F| \leq k$ such that $T \subseteq \mathrm{span}(F)$ in the dual M^* of the binary matroid M represented by A?

▶ **Theorem 2.** Space Cover on Dual-PGM *is solvable in time* $2^{2^{\mathcal{O}((2^r+k^2)k)}} \cdot (n+m)^{\mathcal{O}(1)}$. *In particular,* Space Cover on Dual-PGM *is* FPT *when parameterized by* $r + k$.

As in the case with graphic matroids, it is useful to recall how Space Cover on Dual-PGM is solvable for $r = 0$, i.e. on cographic matroids. In a cographic matroid a circuit corresponds to a cut in the underlying graph G. In this case the solution set F should satisfy the following property: for every terminal element $e \in T$ there is a partition (or a cut) $(X_e, \overline{X}_e)$ of the vertex set of G such that this cut, i.e. the set of edges between X_e and $\overline{X}_e$, is of the form $\{e\} \cup F_e$, where $F_e \subseteq F$. Thus e is the only edge in the cut from T and all other edges are from F.

In graph theory this problem is known under name Edge Subset Feedback Edge Set. Xiao and Nagamochi [25] showed that this problem is FPT parameterized by $k = |F|$. The algorithm for solving Edge Subset Feedback Edge Set, as well as its special case Multiway Cut, uses the technique of Marx based on important separators [20]. The essence of this technique is that all required information about the cuts in a graph can be extracted from a carefully selected set of separators of size at most k. However, we do not see how this approach can be shifted to more general matroids, even when the rank of perturbation matrix is 1. The difficulty in this case is that solution F together with T cannot be represented as the union of the sets of edges of cuts in G anymore, and thus the sizes of important separators in G cannot be bounded by a function of k only. In order to overcome this challenge, we have to apply more powerful method of recursive understanding [6].

On a general level, the structure of the proof of Theorem 2 is similar to the structure of the proof of Theorem 1. It consists of two parts. In the first part we give FPT-Turing reduction to a cut problem on graphs and in the second part we use the method of recursive understanding to solve the problem. But here the similarities end. While on perturbation of graphic matroids Space Cover is about subgraph isomorphisms, on perturbation of cographic matroids it is about collections of cuts in graphs. This makes both parts of the proof of Theorem 2 much more challenging than in Theorem 1. In order to introduce the graph-cut problem we reduce to, we need several definitions.

Graph problem. Let G be a graph with n vertices and m edges given together with a set of terminal edges T and a partition of $V(G) = (V_1, V_2, \ldots, V_t)$. In addition, for every $e \in T$ graph G is provided with a function $f_e : E(G) \to \{0, 1\}$ and a binary vector $B^e = (b_1^e, b_2^e, \ldots, b_t^e)$.

For terminal edge $e \in T$ and a partition $(X, \bar{X})$ of $V(G)$, we say that an edge $e' \in E(G)$ *contributes to* $(e, (X, \bar{X}))$ (with respect to f_e) if one of the following conditions holds

1. Both endpoints of e' belong to X and $f_e(e') = 1$.
2. Both endpoints of e' belong to $\bar{X}$ and $f_e(e') = 1$.
3. Exactly one of the endpoints of e' belongs to X and $f_e(e') = 0$.

Accordingly, we define contribute(e, X) as the set of edges that contribute to $(e, (X, \bar{X}))$.

For partition $(X, \bar{X})$ of $V(G)$, and terminal edge $e \in T$, we say that $(X, \bar{X})$ *almost fits* e (with respect to f_e) if $T \cap$ contribute$(e, X) = \{e\}$. Moreover, if $(X, \bar{X})$ almost fits e and for all $1 \leq i \leq t$, it holds that $|X \cap V_i| = b_i^e \mod 2$, then we say that $(X, \bar{X})$ fits e (with respect to f_e and B^e).

We are now ready to define our graph problem.

Edge-Set Cover
Input: A (multi) graph G with n vertices and m edges, non-negative integers k and t, a partition $(V_1, V_2, \ldots, V_t)$ of $V(G)$, a set $T \subseteq E(G)$, a binary vector $B^e = (b_1^e, b_2^e, \ldots, b_t^e)$ for $e \in T$, and a function $f_e : E(G) \to \{0, 1\}$ for $e \in T$.
Question: Is there a set $F \subseteq E(G) \setminus T$ with $|F| \leq k$ such that for each $e \in T$, there exists a partition $(X_e, \bar{X}_e)$ of $V(G)$ that fits e and such that contribute$(e, X_e) \setminus \{e\} \subseteq F$?

In other words, we a looking for a set of edges F of size k, such that for every terminal edge e, there is a cut $(X_e, \bar{X}_e)$ such that (i) the parities of the intersections of X_e with sets V_i constitute vector B^e, (ii) e is the only terminal edge contributing to the cut and all other edges contributing to the cut are from F.

In the first part of the proof we give a reduction that for an input (G, P, T, k) of Space Cover on Dual-PGM in time $2^{\mathcal{O}(k2^r)} \cdot (n+m)^{\mathcal{O}(1)}$ constructs $2^{\mathcal{O}(k2^r)} \cdot (n+m)^{\mathcal{O}(1)}$ instances of Edge-Set Cover such that (G, P, T, k) is a yes-instance if and only if at least one of the instances of Edge-Set Cover is.

As in the case of perturbed graphic matroids, we can assume that $|T| \leq k$. Let $\mathrm{disr}(P) = \{R_1, \ldots, R_t\}$ be the set of the distinct vectors corresponding the rows of P. Since the rank of P is r, it has at most 2^r different rows, hence $t \leq 2^r$. Accordingly, we say that a vertex $v \in V(G)$ is of *type* i, $1 \leq i \leq t$, if $P_v = R_i$. Given a vertex $v \in V(G)$, we let type(v) denote its type. For $i \in \{1, \ldots, t\}$, we denote by V_i the set of vertices of type i.

Characterization of solutions. For Space Cover on Dual-PGM, we use the term *solution* to refer to a set $F \subseteq E \setminus T$ with $|F| \leq k$ such that $T \subseteq \mathrm{span}(F)$ in the dual M^* of the binary matroid M represented by A. Let I be a binary vector with m elements. Recall that given

$F \subseteq E$, edges(F) denotes the set of all edges $e \in E(G)$ such that $A^e \in F$. Now, given a set $F \subseteq E$, we say that I is the *characteristic vector* of F if the i^{th} entry of I is 1 if and only if F contains the i^{th} column of A. Moreover, a set $F \subseteq E$ is a *cocycle* in M if and only if it is a cycle in M^*. We need the following folklore result (see, e.g., [17]) characterizing cocycles of binary matroids.

▶ **Proposition 3.** *Let M be a binary matroid represented by an $(n \times m)$-matrix A, and let F be a subset of E, where E is the set of columns of A. Then, F is a cocycle in M if and only if the characteristic vector of F belongs to* span(V)*, where V is the set of rows of A.*

Note that a set $F \subseteq E \setminus T$ is a solution if and only if for each terminal $W \in T$, there exists a subset $F_W \subseteq F$ such that $F_W \cup \{W\}$ is a cocycle in M. Thus, in light of Proposition 3, we can think of a solution as follows:

▶ **Observation 4.** *A set $F \subseteq E \setminus T$ is a solution if and only if $|F| \leq k$ and for each terminal $W \in T$, there exists a subset $F_W \subseteq F$ such that the characteristic vector of $F_W \cup \{W\}$ belongs to* span(V)*, where V is the set of rows of A.*

Let F be a solution. For each $W \in T$, denote by $e(W)$ the edge of G corresponding to the terminal W. By Observation 4, for each $W \in T$, there is $F_W \subseteq F$ such that the characteristic vector I_W of $F'_W = F_W \cup \{W\}$ belongs to span(V). It means that there is a set of vertices $X_{e(W)} \subseteq V(G)$ such that $I_W = \sum_{v \in X_{e(W)}} A_v$. Hence, for each $W \in T$, we have the corresponding partition $(X_{e(W)}, \bar{X}_{e(W)})$ of $V(G)$, and the solution can be represented as a collection of cuts $\{(X_{e(W)}, \bar{X}_{e(W)}) \mid W \in T\}$ of G.

For each $W \in T$ and $i \in \{1, \ldots, t\}$, we guess the parity of $|X_{e(W)} \cap V_i|$ and define the vector $B^{e(W)} = (b_1^{e(W)}, \ldots, b_t^{e(W)})$ respectively by setting $b_i^{e(W)} = |X_{e(W)} \cap V_i| \mod 2$. Notice that we have at most 2^{tk} choices for $B^{e(W)}$, because $|T| \leq k$. For each guess, we are now looking for a solution represented by a collection of cuts $\{(X_{e(W)}, \bar{X}_{e(W)}) \mid W \in T\}$ of G such that $|X_{e(W)} \cap V_i| \mod 2 = b_i^{e(W)}$ for $W \in T$ and $i \in \{1, \ldots, t\}$.

Let $I_W = \sum_{v \in X_{e(W)}} A_v$ and let i_e^W for $e \in E(G)$ denote the elements of I_W. We have that

$$I_W = \sum_{v \in X_{e(W)}} (I_v(G) + P_v) = \sum_{v \in X_{e(W)}} I_v(G) + \sum_{v \in X_{e(W)}} P_v. \tag{3}$$

Let $P_W = \sum_{v \in X_{e(W)}} P_v$. Since $|X_{e(W)} \cap V_i| \mod 2 = b_i^{e(W)}$ for $W \in T$ and $i \in \{1, \ldots, t\}$, we obtain that $P_W = \sum_{i=1}^{t} b_i^{e(W)} R_i$. Notice that vector P_W is uniquely defined by the choice of $B^{e(W)}$. We define $f_{e(W)} \colon E(G) \to \{0, 1\}$, by setting $f_{e(W)}(e)$ to be equal to the element of P_W corresponding to e.

Recall that I_W is the characteristic vector of the cocycle F'_W. It means that $A^e \in F'_W$ if and only if $i_e^W = 1$. By making use of (3), we are able to show that for each edge $e \in E(G)$, $A^e \in F'_W$ if and only if one of the following holds:

- Both endpoints of e belong to $X_{e(W)}$ and $f_{e(W)}(e) = 1$.
- Both endpoints of e belong to $\bar{X}_{e(W)}$ and $f_{e(W)}(e) = 1$.
- Exactly one of the endpoints of e belongs to $X_{e(W)}$ and $f_{e(W)}(e) = 0$.

We have that $W \in F'_W$ and W is the unique element of T in this set. It means that for edge $e(W)$, cut $(X_{e(W)}, \bar{X}_{e(W)})$ almost fits $e(W)$ with respect to $f_{e(W)}$. Since $|X_{e(W)} \cap V_i| \mod 2 = b_i^{e(W)}$ for each $W \in T$ and $i \in \{1, \ldots, t\}$, we have that $(X_{e(W)}, \bar{X}_{e(W)})$ fits $e(W)$ with respect to $f_{e(W)}$ and $B^{e(W)}$. Moreover, we prove that for each $W \in T$ and $i \in \{1, \ldots, t\}$ the following are equivalent

- F'_W is a cocycle of M such that $|X_{e(W)} \cap V_i| \mod 2 = b_i^{e(W)}$ and such that its characteristic vector is expressible as $I_W = \sum_{v \in X_{e(W)}} A_v$;
- Cut $(X_{e(W)}, \bar{X}_{e(W)})$ fits $e(W)$ with respect to $f_{e(W)}$ and $B^{e(W)}$.

We also have that $F'_W = \text{contribute}(e(W), X_{e(W)})$.

Now we can complete the reduction to Edge-Set Cover. We consider the partition $(V_1, \ldots, V_t)$ of $V(G)$ and the set of terminal edges $T_G = \{e(W) \mid W \in T\}$. For each $W \in T$, we have a binary vector $B^{e(W)}$ and a function $f_{e(W)}$. Together, all these parameters compose an instance of Edge-Set Cover.

Solving Edge-Set Cover. The algorithm for Edge-Set Cover is the most technical part of the paper. Here we briefly highlight the approach. On a high-level, we use the method of recursive understanding [6], in which we incorporate various new, delicate subroutines. Informally, this means that at the basis, we are going to deal with a "highly-connected" or a small graph, and at each step where our graph is not highly-connected, we will break it using a very small number of edges into two graphs that are both neither too small nor too large.

Let G be a connected graph, and let p and q be positive integers. A partition (X, Y) of $V(G)$ is called (q, p)-*good edge separation* if $|X|, |Y| > q$, $|E(X, Y)| \leq p$, and $G[X]$ and $G[Y]$ are connected graphs.

Roughly speaking, a graph G is unbreakable if every partition of $V(G)$ with few edges going across must contain a large chunk of $V(G)$ in one of its two sets. Intuitively, this means that G is "highly-connected": any attempt to "break" it severely by using only few edges is futile. Formally, a graph G is (q, p)-*unbreakable* if it does not have a (q, p)-*good edge separation*.

If a graph G is not (q, p)-unbreakable, we say that it is (q, p)-breakable. Chitnis et al. [6] proved the following result.

▶ **Proposition 5** ([6]). *There exists a deterministic algorithm that given a connected graph G along with integers q and p, in time $\mathcal{O}(2^{\min\{q,p\}\cdot\log(q+p)} \cdot (n+m)^3 \log(n+m))$ either finds a (q, p)-good edge separation, or correctly concludes that G is (q, p)-unbreakable.*

In our case, we set $p = 2(k+1)$ and $q = 2^{2^{\lambda(t+k^2)|T|}}$ for some appropriate constant λ. To apply the method of recursive understanding, we introduce a special variant of Edge-Set Cover called Annotated Edge-Set Cover (see [14, Section 5] for the formal definition) that is tailored to apply recursion. We show that we can assume that the input graph G is connected. If G has bounded (by some function of r and k) size, we solve Annotated Edge-Set Cover directly. Otherwise, we use Proposition 5 to check whether G is (q, p)-unbreakable.

If G is not (q, p)-unbreakable, we find a (q, p)-good separation (X, Y) of G. Then we solve a special instance of Annotated Edge-Set Cover for one of the graphs $G[X]$ and $G[Y]$ recursively. We use the obtained solution to construct a new instance of the problem for a graph G' that has less vertices than G. Then we call our algorithm for this smaller instance.

If G is (q, p)-unbreakable, we obtain the crucial basic case that we briefly discuss here. For simplicity, we consider this case for Edge-Set Cover.

Recall that in the definition of Edge-Set Cover, we ask about a set $F \subseteq E(G) \setminus T$ with $|F| \leq k$ such that for each $e \in T$, there exists a partition $(X_e, \bar{X}_e)$ of $V(G)$ that fits e and such that $\text{contribute}(e, X_e) \setminus \{e\} \subseteq F$. We relax these conditions and look for a collection of partitions $\{(Y_e, \bar{Y}_e) \mid e \in T\}$ such that $(Y_e, \bar{Y}_e)$ almost fits e and $|\text{contribute}(e, Y_e) \setminus \{e\}| \leq k$ for $e \in T$. Then we can find such an auxiliary collection of partitions $\{(Y_e, \bar{Y}_e) \mid e \in T\}$ by

reducing the relaxed problem to at most k instances of the Edge Odd Cycle Transversal problem (also known as Edge Bipartization). The latter problem could be solved by the results of Guo et al. [18]. Finally we use auxiliary partitions $\{(Y_e, \bar{Y}_e) \mid e \in T\}$ to construct the required collection of partitions $\{(X_e, \bar{X}_e) \mid e \in T\}$ and a set F of size at most k. The final construction heavily exploit the high connectivity of G which allows to search only a "small neighborhood" of $(Y_e, \bar{Y}_e)$.

3 Conclusion

In this paper we established the fixed-parameter tractability of Space Cover on PGM and Space Cover on Dual-PGM. We also know that on the class of binary matroids Space Cover is not tractable. So where lies the tractability border for Space Cover? Our positive results on perturbed matroids, combined with the structure theorem of Geelen, Gerards, and Whittle [16], rise a natural question: could the tractability of Space Cover be extended to any proper minor-closed class $\mathcal{M}$ of binary matroids? Let us note that while we formulate Space Cover only on binary matroids, it can be naturally defined on any class of matroids. In particular, the parameterized complexity of Space Cover on proper minor-closed classes of matroids representable over a finite field is open.

Finally, two concrete open questions. First, what is the parameterized complexity of Space Cover on PGM when $|T|$ is a constant and the parameter is $r + k$? Second, we know that Space Cover on PGM is NP-complete even when $|T| = 2$ and $r \leq 2$ (see [14, Theorem 4]). On the other hand, for $r = 0$ the problem is in P for any fixed number of terminals (it is actually FPT parameterized by $|T|$). What about the case $|T| = 2$ and $r = 1$?

References

1 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

2 Nikhil Bansal, Daniel Reichman, and Seeun William Umboh. LP-based robust algorithms for noisy minor-free and bounded treewidth graphs. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1964–1979. SIAM, 2017.

3 Édouard Bonnet, László Egri, and Dániel Marx. Fixed-parameter Approximability of Boolean MinCSPs. *CoRR*, abs/1601.04935, 2016. arXiv:1601.04935.

4 Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *J. ACM*, 58(3):11:1–11:37, 2011. doi:10.1145/1970392.1970395.

5 Venkat Chandrasekaran, Sujay Sanghavi, Pablo A. Parrilo, and Alan S. Willsky. Rank-Sparsity Incoherence for Matrix Decomposition. *SIAM Journal on Optimization*, 21(2):572–596, 2011. doi:10.1137/090761793.

6 Rajesh Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, Marcin Pilipczuk, and Michal Pilipczuk. Designing FPT Algorithms for Cut Problems Using Randomized Contractions. *SIAM J. Comput.*, 45(4):1171–1229, 2016. doi:10.1137/15M1032077.

7 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

8 Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis. The Complexity of Multiterminal Cuts. *SIAM J. Comput.*, 23(4):864–894, 1994. doi:10.1137/S0097539792225297.

9 Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 3rd edition, 2005.

10 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

11 Rodney G. Downey, Michael R. Fellows, Alexander Vardy, and Geoff Whittle. The Parametrized Complexity of Some Fundamental Problems in Coding Theory. *SIAM J. Comput.*, 29(2):545–570, 1999. doi:10.1137/S0097539797323571.

12 Stuart E. Dreyfus and Robert A. Wagner. The Steiner problem in graphs. *Networks*, 1(3):195–207, 1971. doi:10.1002/net.3230010302.

13 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Covering Vectors by Spaces: Regular Matroids. *SIAM J. Discrete Math.*, 32(4):2512–2565, 2018.

14 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Covering Vectors by Spaces in Perturbed Graphic Matroids and Their Duals. *CoRR*, abs/1902.06957, 2019. arXiv:1902.06957.

15 Jim Geelen, Bert Gerards, and Geoff Whittle. Solving Rota's conjecture. *Notices Amer. Math. Soc.*, 61(7):736–743, 2014.

16 Jim Geelen, Bert Gerards, and Geoff Whittle. The Highly Connected Matroids in Minor-Closed Classes. *Ann. Comb.*, 19(1):107–123, 2015.

17 Jim Geelen and Rohan Kapadia. Computing Girth and Cogirth in Perturbed Graphic Matroids. *Combinatorica*, 38(1):167–191, 2018. doi:10.1007/s00493-016-3445-3.

18 Jiong Guo, Jens Gramm, Falk Hüffner, Rolf Niedermeier, and Sebastian Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *J. Computer and System Sciences*, 72(8):1386–1396, 2006. doi:10.1016/j.jcss.2006.02.001.

19 Avner Magen and Mohammad Moharrami. Robust Algorithms for Max Independent Set on Minor-Free Graphs Based on the Sherali-Adams Hierarchy. In *Proceedings of the 12th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX) and the 13th International Workshop on Randomization and Computation (RANDOM)*, volume 5687 of *Lecture Notes in Comput. Sci.*, pages 258–271. Springer, 2009.

20 Dániel Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006. doi:10.1016/j.tcs.2005.10.007.

21 James G. Oxley. *Matroid theory*, volume 21 of *Oxford Graduate Texts in Mathematics*. Oxford university press, 2nd edition, 2010.

22 Nauman Shahid, Nathanael Perraudin, Vassilis Kalofolias, Gilles Puy, and Pierre Vandergheynst. Fast Robust PCA on Graphs. *J. Sel. Topics Signal Processing*, 10(4):740–756, 2016. doi:10.1109/JSTSP.2016.2555239.

23 Alexander Vardy. The intractability of computing the minimum distance of a code. *IEEE Trans. Inform. Theory*, 43(6):1757–1766, 1997.

24 John Wright, Arvind Ganesh, Shankar R. Rao, YiGang Peng, and Yi Ma. Robust Principal Component Analysis: Exact Recovery of Corrupted Low-Rank Matrices via Convex Optimization. In *Proceedings of 23rd Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2080–2088. Curran Associates, Inc., 2009. URL: http://papers.nips.cc/paper/3704-robust-principal-component-analysis-exact-recovery-of-corrupted-low-rank-matrices-via-convex-optimization.

25 Mingyu Xiao and Hiroshi Nagamochi. An FPT algorithm for edge subset feedback edge set. *Inf. Process. Lett.*, 112(1-2):5–9, 2012. doi:10.1016/j.ipl.2011.10.007.

26 Mengnan Zhao, M. Devrim Kaba, René Vidal, Daniel P. Robinson, and Enrique Mallada. Sparse Recovery over Graph Incidence Matrices. In *Proceedings of the 57th IEEE Conference on Decision and Control (CDC)*, pages 364–371, 2018. doi:10.1109/CDC.2018.8619666.

Decomposition of Map Graphs with Applications

Fedor V. Fomin
University of Bergen, Norway
fomin@ii.uib.no

Daniel Lokshtanov
University of California, Santa Barbara, USA
daniello@ucsb.edu

Fahad Panolan
University of Bergen, Norway
fahad.panolan@ii.uib.no

Saket Saurabh
The Institute of Mathematical Sciences, HBNI, Chennai, India
saket@imsc.res.in

Meirav Zehavi
Ben-Gurion University of the Negev, Beer-Sheva, Israel
meiravze@bgu.ac.il

Abstract

Bidimensionality is the most common technique to design subexponential-time parameterized algorithms on special classes of graphs, particularly planar graphs. The core engine behind it is a combinatorial lemma of Robertson, Seymour and Thomas that states that every planar graph either has a $\sqrt{k} \times \sqrt{k}$-grid as a minor, or its treewidth is $\mathcal{O}(\sqrt{k})$. However, bidimensionality theory cannot be extended directly to several well-known classes of geometric graphs like unit disk or map graphs. This is mainly due to the presence of large cliques in these classes of graphs. Nevertheless, a relaxation of this lemma has been proven useful for unit disk graphs. Inspired by this, we prove a new decomposition lemma for map graphs, the intersection graphs of finitely many simply-connected and interior-disjoint regions of the Euclidean plane. Informally, our lemma states the following. For any map graph G, there exists a collection $(U_1, \ldots, U_t)$ of cliques of G with the following property: *G either contains a $\sqrt{k} \times \sqrt{k}$-grid as a minor, or it admits a tree decomposition where every bag is the union of $\mathcal{O}(\sqrt{k})$ cliques in the above collection.*

The new lemma appears to be a handy tool in the design of subexponential parameterized algorithms on map graphs. We demonstrate its usability by designing algorithms on map graphs with running time $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$ for Connected Planar $\mathcal{F}$-Deletion (that encompasses problems such as Feedback Vertex Set and Vertex Cover). Obtaining subexponential algorithms for Longest Cycle/Path and Cycle Packing is more challenging. We have to construct tree decompositions with more powerful properties and to prove sublinear bounds on the number of ways an optimum solution could "cross" bags in these decompositions.

For Longest Cycle/Path, these are the first subexponential-time parameterized algorithm on map graphs. For Feedback Vertex Set and Cycle Packing, we improve upon known $2^{\mathcal{O}(k^{0.75}\log k)} \cdot n^{\mathcal{O}(1)}$-time algorithms on map graphs.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Computational geometry

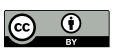
Keywords and phrases Longest Cycle, Cycle Packing, Feedback Vertex Set, Map Graphs, FPT

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.60

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `http://arxiv.org/abs/1903.01291`.

Acknowledgements This work is supported by the European Research Council (ERC) via grant LOPPRE, reference 819416, the Norwegian Research Council via project MULTIVAL, and Israel Science Foundation individual research grant no. 1176/18.



1 Introduction

In this paper, we develop new proof techniques to design parameterized subexponential-time algorithms for problems on map graphs, particularly problems that involve hitting or connectivity constraints. The class of map graphs was introduced by Chen, Grigni, and Papadimitriou [7, 8] as a modification of the class of planar graphs. Roughly speaking, map graphs are graphs whose vertices represent countries in a map, where two countries are considered adjacent if and only if their boundaries have at least one point in common; this common point can be a single common point rather than necessarily an edge as standard planarity requires. Formally, a *map* $\mathcal{M}$ is a pair $(\mathscr{E}, \omega)$ defined as follows : $\mathscr{E}$ is a plane graph (i.e., a planar graph with an embedding) where each connected component of $\mathscr{E}$ is biconnected, and ω is a function that maps each face f of $\mathscr{E}$ to 0 or 1. A face f of $\mathscr{E}$ is called *nation* if $\omega(f) = 1$ and *lake* otherwise. The graph associated with $\mathcal{M}$ is the simple graph G where $V(G)$ consists of the nations of $\mathcal{M}$, and $E(G)$ contains $\{f_1, f_2\}$ for every pair of faces f_1 and f_2 that are adjacent (that is, share at least one vertex). Accordingly, a graph G is called a map graph if there exists a map $\mathcal{M}$ such that G is the graph associated with $\mathcal{M}$.

Every planar graph is a map graph [7, 8], but the converse does not hold true. Moreover, map graphs can have cliques of any size and thus they can be "highly non-planar". These two properties of map graphs can be contrasted with those of H-minor free graphs and unit disk graphs: the class of H-minor free graphs generalizes the class of planar graphs, but can only have cliques of constant size (where the constant depends on H), while the class of unit disk graphs does not generalize the class of planar graphs, but can have cliques of any size. At least in this sense, map graphs offer the best of both worlds. Nevertheless, this comes at the cost of substantial difficulties in the design of efficient algorithms on them.

Arguably, the two most natural and central algorithmic questions concerning map graphs are as follows. First, we would like to efficiently recognize map graphs, that is, determine whether a given graph is a map graph. In 1998, Thorup [29] announced the existence of a polynomial-time algorithm for map graph recognition. Although this algorithm is complicated and its running time is about $\mathcal{O}(n^{120})$, where n is the number of vertices of the input graph, no improvement has yet been found; the existence of a simpler or faster algorithm for map graph recognition has so far remained an important open question in the area (see, e.g., [9]).

The second algorithmic question – or rather family of algorithmic questions – concerns the design of efficient algorithms for various optimization problems on map graphs. Most well-known problems that are NP-complete on general graphs remain NP-complete when restricted to planar (and hence on map) graphs. Nevertheless, a large number of these problems can be solved faster or "better" when restricted to planar graphs. For example, nowadays we know of many problems that are APX-hard on general graphs, but which admit polynomial time approximation schemes (PTASes) or even efficient PTASes (EPTASes) on planar graphs (see, e.g., [4, 14, 15, 22]). Similarly, many parameterized problems that on general graphs cannot be solved in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ unless the Exponential Time Hypothesis (ETH) of Impagliazzo, Paturi and Zane [24] fails, admit parameterized subexponential-time algorithms on planar graphs (see, e.g., [1, 2, 14, 27]). It is compelling to ask whether the algorithmic results and techniques for planar graphs can be extended to map graphs.

For approximation algorithms, Chen [6] and Demaine et al. [12] developed PTASes for the Maximum Independent Set and Minimum r-Dominating Set problems on map graphs. Moreover, Fomin et al. [21, 22] developed an EPTAS for Treewidth-η Modulator for any fixed constant $\eta \geq 0$, which encompasses Feedback Vertex Set (FVS) and Vertex Cover (VC). For parameterized subexponential-time algorithms on map graphs,

the situation is less explored. While on planar graphs there are general algorithmic methods – in particular, the powerful theory of bidimensionality [15, 13] – to design parameterized subexponential-time algorithms, we are not aware of any general algorithmic method that can be easily adapted to map graphs. Demaine et al. [12] gave a parameterized algorithm for Dominating Set, and more generally for (k,r)-Center, with running time $2^{\mathcal{O}(r\log r\sqrt{k})}n^{\mathcal{O}(1)}$ on map graphs. Moreover, Fomin et al. [21, 22] gave $2^{\mathcal{O}(k^{0.75}\log k)}n^{\mathcal{O}(1)}$-time parameterized algorithms for FVS and Cycle Packing on map graphs. Additionally, Fomin et al. [21, 22] noted that the same approach yields $2^{\mathcal{O}(k^{0.75}\log k)}n^{\mathcal{O}(1)}$-time parameterized algorithms for Vertex Cover and Connected Vertex Cover (CVC) on map graphs. However, the existence of a parameterized subexponential-time algorithm for Longest Path/Cycle on map graphs was left open. (In these problems we are asked whether an n-vertex graph contains a path/cycle of length at least k.) Furthermore, time complexities of $2^{\mathcal{O}(k^{0.75}\log k)}n^{\mathcal{O}(1)}$, although having subexponential dependency on k, remain far from time complexities of $2^{\mathcal{O}(\sqrt{k}\log k)}n^{\mathcal{O}(1)}$ and $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$ that commonly arise for planar graphs [27]. We remark that time complexities of $2^{\mathcal{O}(\sqrt{k}\log k)}n^{\mathcal{O}(1)}$ and $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$ are particularly important since they are often known to be essentially optimal under the aforementioned ETH [27].

In the field of Parameterized Complexity, Longest Path/Cycle , FVS and Cycle Packing serve as testbeds for development of fundamental algorithmic techniques such as color-coding [3], methods based on polynomial identity testing [25, 26, 30, 5], cut-and-count [11], and methods based on matroids [19]. By combining the bidimensionality theory of Demaine et al. [13] with efficient algorithms on graphs of bounded treewidth [17, 10], Longest Path/Cycle, Cycle Packing and FVS are solvable in time $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$ on planar graphs. Furthermore, the parameterized subexponential-time "tractability" of these problems can be extended to graphs excluding some fixed graph as a minor [15].

Our results. We design parameterized subexponential-time algorithms with running time $2^{\mathcal{O}(\sqrt{k}\log k)}\cdot n^{\mathcal{O}(1)}$ for a number of natural and well-studied problems on map graphs.

Let $\mathcal{F}$ be a family of connected graphs that contains at least one planar graph. Then Connected Planar $\mathcal{F}$-Deletion (or just $\mathcal{F}$-Deletion) is defined as follows. The input is a graph G and a non-negative integer k, and our objective is to test whether there exists a set S of at most k vertices such that $G-S$ does not contain any of the graphs in $\mathcal{F}$ as a minor. $\mathcal{F}$-Deletion is a general problem and several problems such as VC, FVS, Treewidth-η Vertex Deletion, Pathwidth-η Vertex Deletion, Treedepth-η Vertex Deletion, Diamond Hitting Set and Outerplanar Vertex Deletion are its special cases. We give the first parameterized subexponential algorithm for this problem on map graphs, which runs in time $2^{\mathcal{O}(\sqrt{k}\log k)}\cdot n^{\mathcal{O}(1)}$. Our approach for $\mathcal{F}$-Deletion also directly extends to yield $2^{\mathcal{O}(\sqrt{k}\log k)}\cdot n^{\mathcal{O}(1)}$-time parameterized algorithms for CVC and Connected Feedback Vertex Set (CFVS) on map graphs. (In this versions we are asked if there is a *connected* vertex cover or a feedback vertex set of size at most k.)

With additional ideas, we derive the first subexponential-time parameterized algorithm on map graphs for Longest Path/Cycle. Our technique also allows to improve the running time for Cycle Packing (does a map graph contains at least k vertex-disjoint cycles) from $2^{\mathcal{O}(k^{0.75}\log k)}\cdot n^{\mathcal{O}(1)}$ to $2^{\mathcal{O}(\sqrt{k}\log k)}\cdot n^{\mathcal{O}(1)}$. Our results are summarized in Table 1.

Our methods. The starting point of our study is the technique of bidimensionality [15, 13]. The core engine behind this technique is a combinatorial lemma of Robertson, Seymour and Thomas [28] that states that every planar graph either has a $\sqrt{k}\times\sqrt{k}$-grid as a minor, or its treewidth is $\mathcal{O}(\sqrt{k})$. Unfortunately, a clique on $k-1$ vertices has no $\sqrt{k}\times\sqrt{k}$-grid as a minor

Table 1 Parameterized complexity of problems on map graphs. For $\mathcal{F}$-Deletion, Longest Cycle, and Longest Path no faster (than on general graphs) algorithms were known.

	Our results	Previous work
(Connected) Vertex Cover	$2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$	$2^{\mathcal{O}(k^{0.75}\log k)} \cdot n^{\mathcal{O}(1)}$ [22]
(Connected) Feedback Vertex Set	$2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$	$2^{\mathcal{O}(k^{0.75}\log k)} \cdot n^{\mathcal{O}(1)}$ [22]
$\mathcal{F}$-Deletion	$2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$	$2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ [18]
Longest Cycle/Path	$2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$	$2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ [10]
Cycle Packing	$2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$	$2^{\mathcal{O}(k^{0.75}\log k)} \cdot n^{\mathcal{O}(1)}$ [22]

and its treewidth is $k-2$. Because classes of geometric graphs such as unit disk graphs and map graphs can have arbitrarily large cliques, the combinatorial lemma is inapplicable to them. Nevertheless, a relaxation of this lemma has been proven useful for unit disk graphs. Specifically, every unit disk graph G has a natural partition $(U_1, \ldots, U_t)$ of $V(G)$ such that each part induces a clique with "nice" properties – in particular, it has neighbors only in a *constant number* (to be precise, this constant is at most 24) of other parts; it was shown that G either has a $\sqrt{k} \times \sqrt{k}$-grid as a minor, or it has a tree decomposition where every bag is the union of $\mathcal{O}(\sqrt{k})$ of these cliques [20]. In particular, given a parameterized problem where any two cliques have constant-sized "interaction" in a solution, it is implied that any bag has $\mathcal{O}(\sqrt{k})$-sized "interaction" with all other bags in a solution. For any map graph G, there also exists a natural collection of subsets of $V(G)$ that induce cliques with "nice" properties. However, not only are these cliques not vertex disjoint, but each of these cliques can have neighbors in *arbitrarily many* other cliques.

In this paper, we first prove that every map graph either has a $\sqrt{k} \times \sqrt{k}$-grid as a minor, or it has a tree decomposition where every bag is the union of $\mathcal{O}(\sqrt{k})$ of the cliques in the above collection. For $\mathcal{F}$-Deletion, CVC, and CFVS, this combinatorial lemma alone already suffices to design $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$-time algorithms on map graphs. Indeed, we can choose a fixed constant $c > 0$ so that in case we have a $c\sqrt{k} \times c\sqrt{k}$-grid as a minor, there does not exist a solution, and otherwise we can solve the problem by using dynamic programming over the given tree decomposition. Specifically, since every bag is the union of $\mathcal{O}(\sqrt{k})$ cliques, and the size of each clique is upper bounded by $\mathcal{O}(k)$ (once we know that no $c\sqrt{k} \times c\sqrt{k}$-grid exists), only $\mathcal{O}(\sqrt{k})$ vertices in the bag are not to be taken into a solution – there are only $2^{\mathcal{O}(\sqrt{k}\log k)}$ choices to select these vertices, and once they are selected, the information stored about the remaining vertices is the same as in normal dynamic programming over a tree decomposition of $\mathcal{O}(\sqrt{k})$ width.

This approach already substantially improves upon the previously best known algorithms for FVS, VC and CVC of Fomin et al. [21, 22]. However, $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$-time algorithms for Longest Path/Cycle and Cycle Packing on map graphs require more efforts. The main reason why we cannot apply the same arguments as for unit disk graphs is the following. Recall that for unit disk graphs, given a parameterized problem where any two cliques have constant-sized "interaction" in a *solution* (in our case, this means a path/cycle on at least k vertices, or a cycle packing of k cycles), it is implied that any bag has $\mathcal{O}(\sqrt{k})$-sized "interaction" with all other bags in a solution. Here, interaction between two cliques refers to the number of edges in a solution "passing" between these two cliques; similarly, interaction between a bag B and a collection of other bags refers to the number of edges in a solution that have one endpoint in B and the other endpoint in some bag in the collection. In this context, dealing with map graphs is substantially more difficult than dealing with unit disk

graphs. In map graphs vertices in a clique can have neighbors in arbitrarily many other cliques in the collection rather than only in a constant number as in unit disk graphs. This is why it is difficult to obtain an $\mathcal{O}(\sqrt{k})$-sized "interaction" as for unit disk graphs.

Hence, we are forced to take a different approach for map graphs by bounding "the interaction within a clique across all the bags of a decomposition". Towards this, we first need to strengthen our tree decomposition. To explain the new properties required, we note that every clique in the aforementioned collection of cliques, say $\mathcal{K}$, is either a single vertex or the neighborhood of some "special vertex" in an exterior bipartite graph (see Section 2). Further, every vertex of G occurs as a singleton in $\mathcal{K}$. We construct our decomposition in a way such that every bag is not necessarily a union of $\mathcal{O}(\sqrt{k})$ cliques in $\mathcal{K}$, but a union of carefully chosen subcliques of $\mathcal{O}(\sqrt{k})$ cliques in $\mathcal{K}$ (with one subclique for each of these $\mathcal{O}(\sqrt{k})$ cliques); subcliques of the same clique chosen in different bags may be different. We then prove properties that roughly state that, if we look at the collection of bags that include some vertex v of G, then this collection induces a subtree and a path as follows: (♣) *the subtree consists of the bags that correspond to the singleton clique v, and the path goes "upwards" (in the tree decomposition) from the root of this subtree.* We thereby implicitly derive that in every bag B, every subclique of size larger than 1 can only have as neighbors vertices that are *(i)* in the bag B itself or in one of its descendants, or *(ii)* in cliques that have a subclique in the bag B. In particular, this means that if we prove that there exists a solution such that for any clique K in $\mathcal{K}$, the number of edges in $E(K)$ that "cross any bag B" (i.e., the edges in $E(K)$ with one endpoint in B and the other in the collection of all bags that are not descendants of B) is a constant, then we obtain a bound of $\mathcal{O}(\sqrt{k})$ on the interaction between any bag B and the collection of all bags that are not descendants of B. We prove the mentioned statement using property (♣). The proof that such a property simultaneously holds for all cliques and all bags is the most challenging part of the proof.

In Section 3 we give our special tree decomposition of map graphs and in Section 4 we explain its application in the algorithm for Longest Cycle on map graphs. For the proofs of results marked with $\star$ and all other results we refer to the full version of the paper.

2 Preliminaries

For any $t \in \mathbb{N}$, we use $[t]$ and $[t]_0$ as shorthands for $\{1, 2, \ldots, t\}$ and $\{0, 1, \ldots, t\}$, respectively. For a set U, we use 2^U to denote the power set of U. For a sequence $\sigma = x_1 x_2 \ldots x_n$ and any $1 \leq i \leq j \leq n$, the sequence $\sigma' = x_i \ldots x_j$ is called a *segment* of σ. For a sequence $\sigma = x_1 x_2 \ldots x_n$ and a subset $Z \subseteq \{x_1, \ldots, x_n\}$, the restriction of σ on Z, denoted by $\sigma|_Z$, is the sequence obtained from σ by deleting the elements of $\{x_1, \ldots, x_n\} \setminus Z$.

Graphs. We use standard notation and terminology from the book of Diestel [16] for graph-related terms. Given a graph G, let $V(G)$ and $E(G)$ denote its vertex-set and edge-set, respectively. For a set $\mathcal{Q}$ of graphs we slightly abuse terminology and let $V(\mathcal{Q})$ and $E(\mathcal{Q})$ denote the union of the sets of vertices and edges of the graphs in $\mathcal{Q}$, respectively. For a vertex subset $X \subseteq V(G)$ in a graph G, $E(X)$ denotes the set $\{\{u, v\} \in E(G) \colon u, v \in X\}$. For a graph G and a degree-2 vertex $v \in V(G)$, by *contracting* v, we mean deleting v from G and adding an edge between the two neighbors of v in G.

A binary tree is a rooted tree where each node has at most two children. In a labelled binary tree, for each node with two children one of the children is labelled as "left child" and the other child is labelled as "right child". A *postorder transversal* of a labelled binary tree T is the sequence σ of $V(T)$ where for each node $t \in V(T)$, t appears after all its descendants,

and if t has two children, then the nodes in the subtree rooted at the left child appear before the nodes in the subtree rooted at the right child. For a binary tree T, we say that a sequence σ of $V(T)$ is a postorder transversal if there is a labelling of T such that σ is its postorder transversal.

▶ **Definition 2.1** (Treewidth). *A* tree decomposition *of a graph G is a pair $\mathcal{T} = (T_\tau, \beta_\tau)$, where T is a rooted tree and β_τ is a function from $V(T_\tau)$ to $2^{V(G)}$, that satisfies the following three conditions. (We use the term* nodes *to refer to the vertices of T_τ.)*

(a) $\bigcup_{x \in V(T_\tau)} \beta_\tau(x) = V(G)$.

(b) *For every edge $\{u, v\} \in E(G)$, there exists $x \in V(T_\tau)$ such that $\{u, v\} \subseteq \beta_\tau(x)$.*

(c) *For every vertex $v \in V(G)$, the set of nodes $\{t \in V(T_\tau) : v \in \beta_\tau(t)\}$ induces a (connected) subtree of T_τ.*

The width *of $\mathcal{T}$ is $\max_{x \in V(T_\tau)} |\beta_\tau(x)| - 1$. Each set $\beta_\tau(x)$ is called a* bag. *Moreover, $\gamma_\tau(x)$ denotes the union of the bags of x and its descendants. The* treewidth *of G is the minimum width among all possible tree decompositions of G, and it is denoted by $\mathsf{tw}(G)$.*

▶ **Definition 2.2.** *A tree decomposition $\mathcal{T} = (T_\tau, \beta_\tau)$ of a graph G is* nice *if for the root r of T_τ, it holds that $\beta_\tau(r) = \emptyset$, and each node $v \in V(T_\tau)$ is of one of the following types.*

- ***Leaf**: v is a leaf in T_τ and $\beta_\tau(v) = \emptyset$. This bag is labelled with **leaf**.*
- ***Forget vertex**: v has exactly one child u, and there exists a vertex $w \in \beta_\tau(u)$ such that $\beta_\tau(v) = \beta_\tau(u) \setminus \{w\}$. This bag is labelled with **forget** (w).*
- ***Introduce vertex**: v has exactly one child u, and there exists a vertex $w \in \beta_\tau(v)$ such that $\beta_\tau(v) \setminus \{w\} = \beta_\tau(u)$. This bag is labelled with **introduce**(w).*
- ***Join**: v has exactly two children, u and w, and $\beta_\tau(v) = \beta_\tau(u) = \beta_\tau(w)$. This bag is labelled with **join**.*

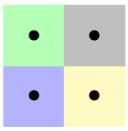
We will use the following folklore observation and proposition in the later sections.

▶ **Observation 2.3.** *Let $\mathcal{T}$ be a nice tree decomposition of a graph G. For any $v \in V(G)$, there is exactly one node $t \in V(T_\tau)$ such that t is labelled with **forget**(v).*

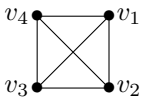
▶ **Proposition 2.4** (Theorem 7.23 in [10], [23, 28]). *There exists an $\mathcal{O}(n^2)$ time algorithm that given an n-vertex planar graph G and $t \in \mathbb{N}$, either outputs a (nice) tree decomposition of G of width less than $5t$, or constructs a $t \times t$ grid minor in G.*

Map graphs. Map graphs are the intersection graphs of finitely many connected and interior-disjoint regions of the Euclidean plane. Map graphs can be represented as the *half-squares of planar bipartite graphs*. For a bipartite graph B with bipartition $V(B) = W \uplus U$, the half-square of B is the graph G with vertex set W and edge set is defined as follows: two vertices in W are adjacent in G if they are at distance 2 in B. It is known that the half-square of a planar bipartite graph is a map graph [7, 8]. Moreover, for any map graph G, there exists a planar bipartite graph B such that G is a half-square of B [7, 8]; we refer to such B as a planar bipartite graph *corresponding* to the map graph G (see Figure 1).

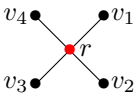
Throughout this paper, we assume that any input map graph G is given with a corresponding planar bipartite graph B. This assumption is made without loss of generality in the sense that if G is given with an embedding instead to witness that it is a map graph, then B is easily computable in linear time [7, 8]. We remark that we consider map graphs as simple graphs, that is, there are no multiple edges between two vertices u and v, even if there are two or more internally vertex-disjoint paths of length 2 between u and v in B. For a map graph G with a corresponding planar bipartite graph B having bipartition $V(B) = W \uplus U$, we refer to the vertices in $W = V(G)$ simply as *vertices* and the vertices in U as *special vertices*.



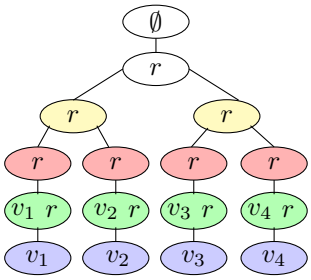
(a) A map $\mathcal{M}$.



(b) A map graph G associated with $\mathcal{M}$.



(c) A corresponding planar bipartite graph B of the map graph G. Here, red colored vertices are special vertices.



(d) A tree decomposition $\mathcal{D} = (T_\mathcal{D}, \beta_\mathcal{D})$ of B of width < 2. The nodes colored blue, green, red, and yellow are labelled with **introduce**(v_i), **introduce**(r), **forget**(v_i) and **join**, resp.

Figure 1 Example of a map graph G, and a corresponding planar bipartite graph B. Figure 1d represents a tree decomposition of B (obtained after deleting the leaves of a nice tree decomposition).

Moreover, we denote the special vertices by $S(G)$. Notice that for any $s \in S(G)$, $N_B(s)$ forms a clique in G; we refer to these cliques as *special cliques* of G. We remark that the collection $\mathcal{K}$ of cliques mentioned in Section 1 refers to $\{N_B(s) \colon s \in S(G)\} \cup \{\{v\} \colon v \in V(G)\}$.

3 Few Cliques Tree Decomposition of Map Graphs

In this section, we define a special tree decomposition for map graphs. This decomposition will be derived from a tree decomposition of the bipartite planar graph corresponding to the given map graph. Once we have defined our new decomposition, we will gather a few of its structural properties that will be useful in designing fast subexponential time algorithms.

▶ **Definition 3.1.** *Let G be a map graph with a corresponding planar bipartite graph B. Let $\mathcal{D} = (T_\mathcal{D}, \beta_\mathcal{D})$ be a tree decomposition of B of width less than ℓ. A pair $\mathcal{D}' = (T_{\mathcal{D}'}, \beta_{\mathcal{D}'})$ is called the* ℓ-few cliques tree decomposition derived from $\mathcal{D}$, *or simply an* $(\ell, \mathcal{D})$-FewCliTD, *if it is constructed as follows (see Figure 2).*

1. *The tree $T_{\mathcal{D}'}$ is equal to $T_\mathcal{D}$. Whenever $\mathcal{D}'$ and $\mathcal{D}$ are clear from context, we denote both $T_{\mathcal{D}'}$ and $T_\mathcal{D}$ by T.*
2. *For each node $t \in V(T)$, $\beta_{\mathcal{D}'}(t) = (\beta_\mathcal{D}(t) \cap V(G)) \cup (\bigcup_{s \in \beta_\mathcal{D}(t) \cap S(G)} N_B(s) \cap \gamma_\mathcal{D}(t))$. That is, for each node $t \in V(T)$, we derive $\beta_{\mathcal{D}'}(t)$ from $\beta_\mathcal{D}(t)$ by replacing every special vertex $s \in \beta_\mathcal{D}(t) \cap S(G)$ by $N_B(s) \cap \gamma_\mathcal{D}(t)$.*

In words, the second item states that for every vertex $v \in V(G)$ and node $t \in V(T)$, we have that $v \in \beta_{\mathcal{D}'}(t)$ if and only if either (i) $v \in \beta_\mathcal{D}(t) \cap V(G)$ or (ii) $v \in N_B(s)$ for some $s \in S(G) \cap \beta_\mathcal{D}(t)$ and $v \in \beta_\mathcal{D}(t')$ for some node t' in the subtree of T rooted at t.

We can prove that the $(\ell, \mathcal{D})$-FewCliTD $(T, \beta_{\mathcal{D}'})$ in Definition 3.1 is a tree decomposition of G (see the full version of the paper for a proof). We remark that if we replace the term $N_B(s) \cap \gamma_\mathcal{D}(t)$ by the term $N_B(s)$ in the second item of Definition 3.1, then we still derive a tree decomposition, but then some of the properties proved later do not hold true.

To simplify statements ahead, from now on, we have the following notation.

> Throughout the section, we fix a map graph G, a corresponding planar bipartite graph B of G, an integer $\ell \in \mathbb{N}$, a nice tree decomposition $\mathcal{D}$ of B of width less than ℓ and an *ℓ-few cliques tree decomposition* $\mathcal{D}'$ of G derived from $\mathcal{D}$ using Definition 3.1

Recall that $T = T_\mathcal{D} = T_{\mathcal{D}'}$ and that for each node $t \in V(T)$, $\beta_{\mathcal{D}'}(t)$ was obtained from $\beta_\mathcal{D}(t)$ by replacing every special vertex $s \in S(G)$ with $N_B(s) \cap \gamma_\mathcal{D}(s)$.

▶ **Definition 3.2.** *For a node $t \in V(T)$, we use* Original(t) *to denote the set $\beta_{\mathcal{D}}(t) \cap \beta_{\mathcal{D}'}(t)$,* Fake$(t)$ *to denote the set $\beta_{\mathcal{D}'}(t) \setminus \beta_{\mathcal{D}}(t)$, and* Cliques$(t)$ *to denote the set $\{N_B(s) \colon s \in S(G) \cap \beta_{\mathcal{D}}(t)\}$ of special cliques of G.*

Informally, for a node $t \in V(T)$, $\mathsf{Original}(t)$ denotes the set of vertices of $V(G)$ present in the bag $\beta_{\mathcal{D}}(t)$, $\mathsf{Fake}(t)$ denotes the set of "new" vertices added to $\beta_{\mathcal{D}'}(t)$ while replacing special vertices in $\beta_{\mathcal{D}}(t)$, and $\mathsf{Cliques}(t)$ is the set of special cliques in G that consist of one for each special vertex $s \in \beta_{\mathcal{D}}(t)$. For example, let t be the node in Figure 1d that is labelled with $\mathbf{forget}(v_1)$ by $\mathcal{D}$. Then, $\mathsf{Original}(t) = \emptyset$, $\mathsf{Fake}(t) = \{v_1\}$ and $\mathsf{Cliques}(t) = \{\{v_1, \ldots, v_4\}\}$.

In the remainder of this section we prove properties related to $\mathcal{D}$ and $\mathcal{D}'$, which we use later in the paper. Towards the formulation of the first property, consider the tree decomposition $\mathcal{D}'$ in Figure 2 and the set of its nodes whose bags contain the vertex v_1 as a "fake" vertex. This set of nodes forms a path with one end-vertex being the unique node t_{v_1} of T labelled with $\mathbf{forget}(v_1)$ by $\mathcal{D}$ and the other end-vertex being an ancestor of t_{v_1}. In fact, the set of nodes $Q = \{t \in V(T) \colon v_1 \in \mathsf{Fake}(t) \text{ and } r \in \beta_{\mathcal{D}}(t)\}$ forms the unique path in T from t_{v_1} to t_r where t_r is the unique child of the node labelled with $\mathbf{forget}(r)$ by $\mathcal{D}$. This observation is abstracted and formalized in the following lemma.

▶ **Lemma 3.3.** *Let $v \in V(G)$ and $s \in S(G)$ such that $v \in N_B(s)$ and $Q = \{t \in V(T) \colon v \in$ Fake(t) and $s \in \beta_{\mathcal{D}}(t)\} \neq \emptyset$. Let x be the node in T labelled with $\mathbf{forget}(v)$ by $\mathcal{D}$, and y be the unique child of the node labelled with $\mathbf{forget}(s)$ by $\mathcal{D}$. Then, y is an ancestor of x, and Q induces a path in T which is the unique path between x and y in T.*

Proof. First, we prove that Q induces a (connected) subtree of T. Suppose not. Then, there exist two connected components C_1 and C_2 of $T[Q]$ such that there exists a path P in T from a vertex in C_1 to a vertex in C_2 whose internal vertices all belong to $V(T) \setminus Q$. By Property (c) of the tree decomposition $\mathcal{D}$, we have that $s \in \beta_{\mathcal{D}}(t)$ for any $t \in V(P)$. Moreover, there is an internal vertex w of P such that w is an ancestor of one of the end-vertices of P. This implies that $v \in \gamma_{\mathcal{D}}(w)$, because v belong to the bags of the endpoints of P (by the definition of Q and Fake). As we have also shown that $s \in \beta_{\mathcal{D}}(t)$ for all $t \in V(P)$, this implies that $w \in Q$, which is a contradiction. Hence, we have proved that $T[Q]$ is connected.

Next, we prove that $T[Q]$ is a path such that one of its endpoints is a descendant of the other. Towards this, it is enough to prove that (i) for any distinct $t, t' \in Q$, either t is a descendant of t' or t' is a descendant of t. For the sake of contradiction, assume that there exist $t, t' \in Q$ such that neither t is a descendent of t' nor t' is a descendent of t. By the definition of Q and because $t, t' \in Q$, we have that $v \in \gamma_{\mathcal{D}}(t)$ and $v \in \gamma_{\mathcal{D}}(t')$. Thus by Property (c) of the tree decomposition $\mathcal{D}$, we have that $v \in \beta_{\mathcal{D}}(t)$ and $v \in \beta_{\mathcal{D}}(t')$. Because $v \in \mathsf{Fake}(t)$ and $v \in \mathsf{Fake}(t')$, this is a contradiction to the definition of Fake.

It remains to prove that y is an ancestor of x and that x and y are endpoints of $T[Q]$. First, we prove that x is an end-vertex of the path $T[Q]$. Let x' be the only child of x. To prove x is an end-vertex of the path $T[Q]$, it is enough to show that $x \in Q$ and $x' \notin Q$. Since x is labelled with $\mathbf{forget}(v)$ by $\mathcal{D}$, we have that $v \notin \beta_{\mathcal{D}}(x)$, $v \in \beta_{\mathcal{D}}(x')$, and $v \in \gamma_{\mathcal{D}}(x)$. This implies that $v \in \mathsf{Original}(x')$ and hence $x' \notin Q$. Now, we prove that $x \in Q$. For this purpose, let $R = \{t \in V(T) \colon s \in \beta_{\mathcal{D}}(t)\}$. Clearly, $Q \subseteq R$. By Property (c) of the tree decomposition $\mathcal{D}$, we have that $T[R]$ is connected. We have already proved that $T[Q]$ is a path and since $Q \subseteq R$, $T[Q]$ is a path in $T[R]$. Since x is labelled with $\mathbf{forget}(v)$ by $\mathcal{D}$, for any node x'' in the subtree rooted at x and $x'' \neq x$, either $v \in \beta_{\mathcal{D}}(x'')$ or $v \notin \gamma_{\mathcal{D}}(x'')$ (this fact follows from Property (c) of $\mathcal{D}$). This implies that Q contains no node in the subtree of T rooted at x and not equal to x. Moreover, observe that there exists a node $x^\star$ in the subtree of T rooted

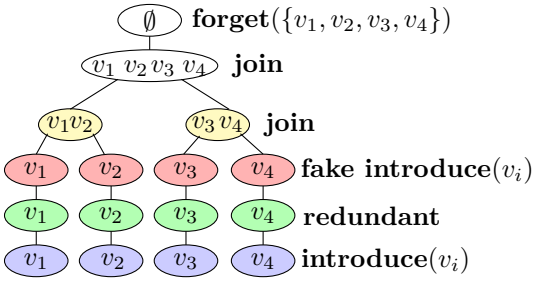


Figure 2 An $(\ell, \mathcal{D})$-*FewCliTD* $\mathcal{D}'$ derived from the nice tree decomposition $\mathcal{D}$ in Figure 1d using Definition 3.1. The labels of the nodes in $\mathcal{D}'$ are mentioned on the right.

at x such that $\{s, v\} \subseteq \beta_{\mathcal{D}}(x^\star)$ and hence $x^\star \in R$. Now, since Q is non-empty and $T[Q]$ is connected, we have that $s \in \beta_{\mathcal{D}}(x)$. Since $v \notin \beta_{\mathcal{D}}(x)$, $v \in \gamma_{\mathcal{D}}(x)$ and $s \in \beta_{\mathcal{D}}(x)$, we conclude that $x \in Q$. Thus, we have proved that x is an end-vertex of the the path $T[Q]$.

Next we prove that y is the other end-vertex of the path $T[Q]$ and y is an ancestor of x. Since y is the only child of the node y' labelled with **forget**(s), we have that $s \in \beta_{\mathcal{D}}(y)$ and $s \notin \beta_{\mathcal{D}}(y')$. This implies that $y' \notin Q$. Thus to prove that y is an end-vertex of the path $T[Q]$, it is enough to prove that $y \in Q$. Since $s \in \beta_{\mathcal{D}}(x)$, $s \in \beta_{\mathcal{D}}(y)$, $s \notin \beta_{\mathcal{D}}(y')$ and y' is the parent of y, by Property (c) of $\mathcal{D}$, we have that y is an ancestor of x. This also implies that $v \in \gamma_{\mathcal{D}}(y)$ and $v \notin \beta_{\mathcal{D}}(y)$. Hence, $y \in Q$. This completes the proof of the lemma. ◀

In the next lemma we show that for any special vertex $s \in S(G)$ and any node t in T labelled with **introduce**(s) by $\mathcal{D}$, it holds that t and its child carry the "same information".

▶ **Lemma 3.4** $(\star)$**.** *Let $s \in S(G)$ and t be a node in T labelled with **introduce**(s) by $\mathcal{D}$. Let t' be the only child of t. Then, $\mathsf{Original}(t) = \mathsf{Original}(t')$ and $\mathsf{Fake}(t) = \mathsf{Fake}(t')$.*

Next, we see a property of nodes $t \in V(T)$ labelled with **join**.

▶ **Lemma 3.5** $(\star)$**.** *Let t be a node in T labelled with **join** by D, and t_1 and t_2 are its children. Then, $\mathsf{Original}(t) = \mathsf{Original}(t_1) = \mathsf{Original}(t_2)$, $\mathsf{Cliques}(t) = \mathsf{Cliques}(t_1) = \mathsf{Cliques}(t_2)$, $\mathsf{Fake}(t_1) \cap \mathsf{Fake}(t_2) = \emptyset$, and $\mathsf{Fake}(t) = \mathsf{Fake}(t_1) \cup \mathsf{Fake}(t_2)$.*

Now, we define a notion of *nice ℓ-few cliques tree decomposition* of G as the tree decomposition of G derived from a nice tree decomposition $\mathcal{D}$ of B of width $< \ell$ (see Definition 3.1) with additional labeling of nodes. In what follows, we describe this additional labeling of nodes. Towards this, observe that because of Lemma 3.4, for any special vertex $s \in S(G)$ and any node $t \in V(T)$ labelled with **introduce**(s) by $\mathcal{D}$, the bags $\beta_{D'}(t)$ and $\beta_{D'}(t')$ carry the "same information" where t' is the only child of t. Informally, one may choose to handle these nodes by contracting them. However, to avoid redundant proofs ahead, instead of getting rid of such nodes, we label them with **redundant** in $\mathcal{D}'$. Next, we explain how to label other nodes of T in the decomposition $\mathcal{D}'$ (see Figure 2). To this end, let $t \in V(T)$.

- If t is labelled with **leaf** by $\mathcal{D}$, then we label t with **leaf**. Here, $\beta_{\mathcal{D}'}(t) = \emptyset$.
- If t is labelled with **introduce**(v) by $\mathcal{D}$ for some $v \in V(G)$, then we label t with **introduce**(v). In this case, t has only one child t' in T and $\beta_{\mathcal{D}'}(t) \setminus \{v\} = \beta_{\mathcal{D}'}(t')$.
- If t is labelled with **forget**(v) by $\mathcal{D}$ for some $v \in V(G)$ and $v \in \mathsf{Fake}(t)$, then we label t with **fake introduce**(v). In this case, t has only one child t' and $\beta_{\mathcal{D}'}(t) = \beta_{\mathcal{D}'}(t')$, but $\mathsf{Original}(t) = \mathsf{Original}(t') \setminus \{v\}$ and $\mathsf{Fake}(t) = \mathsf{Fake}(t') \cup \{v\}$.

- If t is labelled with **forget**(v) by $\mathcal{D}$ for some $v \in V(G)$ and $v \notin \mathsf{Fake}(t)$, then we label t with **forget**(v). In this case, t has only one child t', $\beta_{\mathcal{D}'}(t) = \beta_{\mathcal{D}'}(t') \setminus \{v\}$, $\mathsf{Original}(t) = \mathsf{Original}(t') \setminus \{v\}$ and $\mathsf{Fake}(t) = \mathsf{Fake}(t')$.
- Suppose t is labelled with **forget**(s) by $\mathcal{D}$ for some $s \in S(G)$. Then, t has only one child t'. Here, we label t with **forget**$(\beta_{\mathcal{D}'}(t') \setminus \beta_{\mathcal{D}'}(t))$. In this case, $\mathsf{Fake}(t) \subseteq \mathsf{Fake}(t')$ and $\mathsf{Original}(t) = \mathsf{Original}(t')$.
- If t is labelled with **join** by $\mathcal{D}$, then we label t with **join**. Let t_1 and t_2 be the children of t. Then, $\mathsf{Original}(t) = \mathsf{Original}(t_1) = \mathsf{Original}(t_2)$, $\mathsf{Cliques}(t) = \mathsf{Cliques}(t_1) = \mathsf{Cliques}(t_2)$, $\mathsf{Fake}(t_1) \cap \mathsf{Fake}(t_2) = \emptyset$, and $\mathsf{Fake}(t) = \mathsf{Fake}(t_1) \cup \mathsf{Fake}(t_2)$. (See Lemma 3.5).
- If t is labelled with **introduce**(s) for some $s \in S(G)$, then we label t with **redundant**.

This completes the definition of the nice ℓ-few cliques tree decomposition of G derived from $\mathcal{D}$, to which we simply call an $(\ell, \mathcal{D})$-NFewCliTD. Notice that for each node t in T, $|\mathsf{Original}(t)| + |\mathsf{Cliques}(t)| \leq \ell$. That is, for any node $t \in V(T)$, there exist $i, j \in \mathbb{N}$ such that $i + j \leq \ell$, the cardinality of $\mathsf{Original}(t)$ is at most i, and the vertices in $\beta_{\mathcal{D}'}(t) \setminus \mathsf{Original}(t)$ were obtained from at most j special cliques. From now on, we assume that $\mathcal{D}'$ is an $(\ell, \mathcal{D})$-NFewCliTD of a map graph G.

Since the number of nodes with label **forget**(v) in the tree decomposition $\mathcal{D}$ is exactly one for any $v \in V(B)$ (see Observation 2.3), *at most one* node in T is labelled with **fake introduce**(v) in $\mathcal{D}'$. This is formally stated in the following observation.

▶ **Observation 3.6.** *Let $t \in V(T)$ and $v \in \mathsf{Fake}(t)$. Then,*

(i) *there is a unique node $t' \in V(T)$ such that t' is labelled with **fake introduce**(v) in $\mathcal{D}'$,*

(ii) *t is an ancestor of t' or $t = t'$, and*

(iii) *for any node t'' in the unique path between t and t', we have that $v \in \mathsf{Fake}(t'')$.*

The correctness of Observation 3.6 follows from Observation 2.3 and Lemma 3.3. The discussion above, along with Proposition 2.4, implies the following lemma.

▶ **Lemma 3.7 ($\star$).** *Given a map graph G, a corresponding planar bipartite graph B, and an integer $\ell \in \mathbb{N}$, in time $\mathcal{O}(n^2)$, one can either correctly conclude that B contains an $\ell \times \ell$ grid as a minor, or compute a nice tree decomposition $\mathcal{D}$ of B of width less than 5ℓ and a $(5\ell, \mathcal{D})$-NFewCliTD of G.*

Lastly, we prove an important property of $\mathcal{D}'$. In particular, the edges considered in the following lemma are precisely those that connect the vertices "already seen" (when we use dynamic programming (DP)) with vertices to "see in the future".

▶ **Lemma 3.8.** *For any node $t \in V(T)$, the edges with one endpoint in $\gamma_{\mathcal{D}'}(t)$ and other in $V(G) \setminus \gamma_{\mathcal{D}'}(t)$ are of two kinds: (i) edges incident with vertices in $\mathsf{Original}(t)$, and (ii) edges belonging to some special clique in $\mathsf{Cliques}(t)$ (these edges are incident to vertices in $\mathsf{Fake}(t)$).*

Proof. Fix $t \in V(T)$. Since $\mathcal{D}'$ is a tree decomposition of G, for any edge $e \in E(G)$ with one endpoint in $\gamma_{\mathcal{D}'}(t)$ and other in $V(G) \setminus \gamma_{\mathcal{D}'}(t)$, the endpoint of e in $\gamma_{\mathcal{D}'}(t)$ should belong to $\beta_{\mathcal{D}'}(t)$. Let u be the endpoint of e that belongs to $\beta_{\mathcal{D}'}(t)$, and v be the other endpoint of e. Notice that the set $\beta_{\mathcal{D}'}(t)$ is partitioned into $\mathsf{Original}(t)$ and $\mathsf{Fake}(t)$, so u belongs to either $\mathsf{Original}(t)$ or $\mathsf{Fake}(t)$, and in the former case we are done. We now assume that $u \in \mathsf{Fake}(t)$.

Since $\{u, v\} = e \in E(G)$, there is a special vertex $s \in S(G)$ such that $\{u, s\}, \{v, s\} \in E(B)$. If $s \in \beta_{\mathcal{D}}(t)$, then the edge $\{u, v\}$ belongs to the special clique $K = N_B(s)$ in G and $K \in \mathsf{Cliques}(t)$. We claim that indeed $s \in \beta_{\mathcal{D}}(t)$. Towards this, notice that $u \in \mathsf{Fake}(t)$. This implies that $u \notin \beta_{\mathcal{D}}(t)$, but u is present in a bag $\beta_{\mathcal{D}}(t')$ of some descendant t' of t. Moreover, since $\{u, s\} \in E(B)$, we further know that $\{u, s\} \subseteq \beta_{\mathcal{D}}(t_1)$ for some descendent t_1 of t. Since

$\{v, s\} \in E(B)$ and $v \notin \gamma_{\mathcal{D}'}(t)$, there is a bag $\beta_{\mathcal{D}}(t_2)$ such that $\{v, s\} \subseteq \beta_{\mathcal{D}}(t_2)$ and t_2 is not a descendent of t. Thus, since $s \in \beta_{\mathcal{D}}(t_1) \cap \beta_{\mathcal{D}}(t_2)$, by Property (c) of the tree decomposition $\mathcal{D}$, we have that $s \in \beta_{\mathcal{D}}(t)$. This completes the proof of the lemma. ◀

4 Longest Cycle

In this section we prove the following theorem.

▶ **Theorem 4.1.** Longest Cycle *on map graphs can be solved in* $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$ *time.*

Notice that if there is a special vertex $s \in S(G)$ such that $|N_B(s)| \geq k$, then G has a cycle of length at least k, because $N_B(s)$ forms a clique in G. Moreover, observe that if there is a "large enough" grid in B, then we can answer Yes. These observations along with Lemma 3.7 lead to the following lemma.

▶ **Lemma 4.2** (⋆). *There is an algorithm that given an instance* (G, B, k) *of* Longest Cycle*, runs in time* $\mathcal{O}(n^2)$*, and either correctly concludes that* G *has a cycle of length at least* k*, or outputs a nice tree decomposition* $\mathcal{D}$ *of* B *of width* $< 5\sqrt{2k}$ *and a* $(5\sqrt{2k}, \mathcal{D})$*-NFewCliTD* $\mathcal{D}'$ *of* G *such that for each node* $t \in V(T)$*,* $|\beta_{\mathcal{D}'}(t)| \leq 5\sqrt{2} \cdot k^{1.5}$*.*

Because of Lemma 4.2, to prove Theorem 4.1, we may assume that the input contains a $(5\sqrt{2k}, \mathcal{D})$-NFewCliTD $\mathcal{D}'$ of G (derived from a nice tree decomposition $\mathcal{D}$ of B) such that for each node $t \in V(T)$, $|\beta_{\mathcal{D}'}(t)| \leq 5\sqrt{2} \cdot k^{1.5}$. That is, from now on, we fix our input to be an instance (G, B, k) of Longest Cycle, a nice tree decomposition $\mathcal{D}$ of B and a $(5\sqrt{2k}, \mathcal{D})$-NFewCliTD $\mathcal{D}'$ of G such that for each node $t \in V(T)$, $|\beta_{\mathcal{D}'}(t)| \leq 5\sqrt{2} \cdot k^{1.5}$. Towards the proof of Theorem 4.1, the main ingredient is to prove the following claim: if G has a cycle of length ℓ, then there is a cycle C of length ℓ, with the following property.

For each node $t \in V(T)$, the number of edges of $E(C)$ with one endpoint in $\beta_{\mathcal{D}'}(t)$ and the other in $V(G) \setminus \gamma_{\mathcal{D}'}(t)$ is upper bounded by $\mathcal{O}(\sqrt{k})$.

The above mentioned property is encapsulated in the following sublinear crossing lemma.

▶ **Lemma 4.3** (Sublinear Crossing Lemma (⋆))**.** *Let* C *be a cycle in* G*. Then there is a cycle* C' *of the same length as* C *such that for any node* $t \in V(T)$*, the number of edges in* $E(C')$ *with one endpoint in* $\beta_{\mathcal{D}'}(t)$ *and the other in* $V(G) \setminus \gamma_{\mathcal{D}'}(t)$ *is at most* $20\sqrt{2k}$*.*

Lemma 4.3 lies at the heart of the proof of Theorem 4.1 and is one of the main technical contributions of the paper. Assuming Lemma 4.3, the proof of Theorem 4.1 is by designing a DP algorithm on a $(5\sqrt{2k}, \mathcal{D})$-NFewCliTD of G. This part is similar to the algorithm for Longest Cycle in [20] on a so called *special path decomposition*. The main ingredient of Lemma 4.3 is the following lemma (Lemma 4.4). The proof of Lemma 4.3 is by an inductive argument assuming Lemma 4.4. The rest of the section is devoted to the proof of Lemma 4.4.

▶ **Lemma 4.4.** *Let* C *be a cycle in* G *and* K *be a special clique in* G*. Then, there is a cycle* C' *of the same length as* C *such that* $E(C') \setminus E(K) = E(C) \setminus E(K)$ *and for any node* $t \in V(T)$*, the number of edges of* $E(C') \cap E(K)$ *with one endpoint in* $\mathsf{Fake}(t) \cap K$ *and the other in* $V(G) \setminus \gamma_{\mathcal{D}'}(t)$ *is at most* 4*.*

Before formally proving Lemma 4.4, we give a high level overview of the proof and an auxiliary lemma which we use in the proof of Lemma 4.4. The proof idea is to change the edges of $E(K) \cap E(C)$ in C (because in Lemma 4.4 our objective is to bound the "crossing edges" from a subset of $E(K)$ for each node $t \in V(T)$) to obtain a new cycle C' of the same

length as C that satisfies the following property: (i) for any node $t \in V(T)$, the number of edges of $E(C') \cap E(K)$ with one endpoint in $\mathsf{Fake}(t) \cap K$ and the other in $V(G) \setminus \gamma_{\mathcal{D}'}(t)$ is at most 4. For the ease of presentation, assume that $K \subseteq V(C)$. Now, consider the graph $\mathcal{P}$ obtained from the cycle C after deleting edges in $E(K)$. Without loss of generality assume that $E(C) \cap E(K) \neq \emptyset$. Otherwise, Lemma 4.4 is true where $C' = C$. We consider $\mathcal{P}$ as a collection of vertex-disjoint paths where the end-vertices of the paths are in K. Some paths in $\mathcal{P}$ may be of length 0. Let Z be the set of end-vertices of the paths in $\mathcal{P}$. Clearly, $Z \subseteq K$. We will "complete" the collection of paths $\mathcal{P}$ to a cycle by adding edges from $E(Z)$ satisfying Statement (i). Any cycle C' with $V(C') = V(\mathcal{P})$ has the same length as C. So, all the work that is required for us is to complete the collection of paths $\mathcal{P}$ to a cycle by adding edges from $E(Z)$ satisfying Statement (i). Towards that, let $\widehat{\sigma} = v_1, \ldots, v_{k'}$ be an arbitrary sequence of vertices in Z. We show (in Claim 4.7) that (ii) there is a subset of edges $F \subseteq E(Z)$ such that $E(\mathcal{P}) \cup F$ forms a cycle C' with vertex set $V(\mathcal{P})$ and for any $j \in [k']$, the number of edges in F with one endpoint in $\{v_1, \ldots, v_j\}$ and the other in $\{v_{j+1}, \ldots, v_{k'}\}$ is at most 2. This implies that for any $1 \leq i \leq j \leq k'$, the number of edges in F with one endpoint in $\{v_i, \ldots, v_j\}$ and the other in $Z \setminus \{v_i, \ldots, v_j\}$ is at most 4. In the light of Statement (ii), our aim will be to prove that (iii) for any node $t \in V(T)$, there exist $1 \leq i \leq j \leq k'$ such that $\mathsf{Fake}(t) \cap Z \subseteq \{v_i, \ldots, v_j\}$ and no vertex in $Z \setminus \gamma_{\mathcal{D}'}(t)$ belongs to $\{v_i, \ldots, v_j\}$. Then, Statement (i) will follow (because edges of C' incident with vertices in $K \setminus Z$ are from $E(G) \setminus E(K)$ and will not be counted in Statement (i)). In fact, we will prove that there is a sequence σ on Z (derived from a postorder transversal of T) such that Statement (iii) is true (see Claim 4.8). The proof of Statement (ii) is encapsulated in the following lemma.

▶ **Lemma 4.5** $(\star)$**.** *Let $\ell \geq 3$ be an integer. Let $u_1, \ldots, u_\ell$ be a sequence of vertices in a graph H where $X = \{u_1, \ldots, u_\ell\}$ is a clique in H. Let $\mathcal{Q}$ be a family of vertex-disjoint paths in H (which possibly contains paths of length 0) such that each $v \in X$ is an end-vertex of a path in $\mathcal{Q}$ and $E(\mathcal{Q}) \cap E(X) = \emptyset$. Then, there is a set $F \subseteq E(X)$ with the following conditions.*

(a) *$E(\mathcal{Q}) \cup F$ forms a cycle containing all the vertices of $V(\mathcal{Q})$,*

(b) *For any $j \in [\ell]$, the number of edges in F with one endpoint in $\{u_1, \ldots, u_j\}$ and the other in $\{u_{j+1}, \ldots, u_\ell\}$ is at most 2.*

Next, we move to a formal proof of Lemma 4.4.

Proof of Lemma 4.4. Without loss of generality, assume that $K \subseteq V(C)$. Otherwise, we can consider the statement of the lemma for cycle C in the graph $G' = G - (K \setminus V(C))$ and special clique $K \cap V(C)$ of G'. We also assume that $E(C) \cap E(K) \neq \emptyset$, else the correctness is trivial because we can take C' as C.

Let π' be a postorder transversal of the nodes in the rooted binary tree T, and let π be the restriction of π' where we only keep the nodes that are labeled with **fake introduce**(v) for some $v \in K$. Denote $\pi = t_1, \ldots, t_{k''}$ such that each t_i, $i \in [k'']$, is labelled with **fake introduce**(x_i) where $x_i \in K$. Notice that $\bigcup_{t \in V(T)} \mathsf{Fake}(t) \cap K = \{x_1, \ldots, x_{k''}\}$ (by Observation 3.6). Let σ_1 be the sequence $x_1, \ldots, x_{k''}$ and $U = \{x_1, \ldots, x_{k''}\}$. Let σ_2 be a fixed arbitrary sequence of $K \setminus U$, i.e., all the vertices of K that are never "fakely introduced". Let σ be the sequence which is a concatenation of σ_1 and σ_2.

Let $\mathcal{P} = (V(C), E(C) \setminus E(K))$. That is, $\mathcal{P}$ is the graph obtained by deleting edges of $E(K)$ from the cycle C. Notice that each connected component of $\mathcal{P}$ is a path (may be of length 0) with end-vertices in K. Let Z be the set of end-vertices of the paths in $\mathcal{P}$. Notice that for any vertex $u \in K \setminus Z$, both edges of C incident with u are from $E(C) \setminus E(K)$ (see the left part of Figure 3). That is, $E(C) \setminus E(K) = E(C) \setminus E(Z) = E(\mathcal{P})$. Since we seek a cycle C' in which $E(C') \setminus E(K) = E(C) \setminus E(K)$, no edge of C' incident with u for any vertex $u \in K \setminus Z$, is in $E(K)$. That is, all the edges of $E(C') \cap E(K)$ will belong to $E(Z)$.

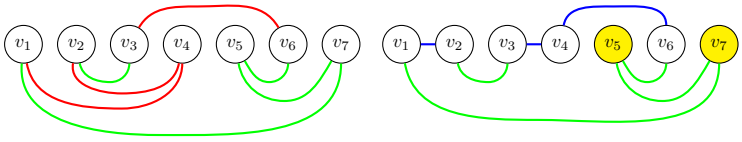


Figure 3 Left part illustrates a cycle C interacting with a special clique $K = \{v_1, \ldots, v_7\}$. The red curves represent edges in $E(C) \cap E(K)$ and green curves represent paths in C with endpoints in K and (at least one) internal vertices in $V(G) \setminus K$. Thus, $\mathcal{P}$ is the collection of paths that is a union of the set of two "green" paths $((v_2 - v_3)$ and $(v_1 - v_7 - v_5 - v_6))$ and $\{[v_4]\}$. Here $Z = \{v_1, v_2, v_3, v_4, v_6\}$. Any edge of $E(C)$ incident with v_5 and v_7 (i.e., vertices in $K \setminus Z$) are from $E(C) \setminus E(K)$. The right part illustrates the proof of Claim 4.7. The edges of $E(C') \setminus E(C)$ mentioned in the proof of Claim 4.7 are colored blue.

▶ **Observation 4.6.** *Let C' be a cycle in G such that $E(C') \setminus E(K) = E(C) \setminus E(K)$ and $t \in V(T)$. The number of edges of $E(C') \cap E(K)$ with one endpoint in $\mathsf{Fake}(t) \cap K$ and the other in $V(G) \setminus \gamma_{\mathcal{D}'}(t)$ is equal to the number of edges of $E(C') \cap E(Z)$ with one endpoint in $\mathsf{Fake}(t) \cap Z$ and the other in $V(G) \setminus \gamma_{\mathcal{D}'}(t)$.*

Let $Z = \{v_1, \ldots, v_{k'}\}$ and $\sigma' = \sigma|_Z = v_1, \ldots, v_{k'}$. The main ingredients of the proof are the following two claims.

▷ Claim 4.7. There is a cycle C' of the same length as C such that (i) $E(C') \setminus E(Z) = E(C) \setminus E(Z)$, and (ii) for any $j \in [k']$, the number of edges of $E(C') \cap E(Z)$ with one endpoint in $\{v_1, \ldots, v_j\}$ and the other in $\{v_{j+1}, \ldots, v_{k'}\}$ is at most 2.

The proof of Claim 4.7 follows from Lemma 4.5.

▷ Claim 4.8. For any node $t \in V(T)$, there is a segment σ'' of σ' such that each vertex in $\mathsf{Fake}(t) \cap Z$ appears in σ'', and each vertex in $Z \setminus \gamma_{\mathcal{D}'}(t)$ does not appear in σ''.

Proof. Fix a node $t \in V(T)$. Recall that $\sigma = \sigma_1\sigma_2$ and $\sigma' = \sigma|_Z$. Here, the set of vertices present in σ_1 is $U = (\bigcup_{t \in V(T)} \mathsf{Fake}(t) \cap K) \supseteq (\bigcup_{t \in V(T)} \mathsf{Fake}(t) \cap Z)$ (because $Z \subseteq K$), and no vertex in σ_2 is from U. This implies that all the vertices of $\mathsf{Fake}(t) \cap Z$ are in the sequence σ_1. That is, the sequence σ'' we seek is also a sequence of $\sigma_1|_Z$ and this is the reason we defined σ to be $\sigma_1\sigma_2$. Thus, to prove the claim it is enough to prove that there is a segment σ_1' of $\sigma_1|_Z$ such that each vertex in $\mathsf{Fake}(t) \cap Z$ appears in σ_1' and each vertex in $Z \setminus \gamma_{\mathcal{D}'}(t)$ does not appear in σ_1'.

Recall that $\sigma_1 = x_1 \ldots x_{k''}$ is obtained from the sequence $\pi = t_1, \ldots, t_{k''}$. In turn, recall that π is the restriction of the postorder transversal π' of T, where for each $i \in [k'']$, t_i is labelled with **fake introduce**(x_i) for $x_i \in K$. Let W_t be the nodes of the subtree of T rooted at t, and $V_t = \{v \in K$: there is $t' \in W_t$ such that t' is labelled with **fake introduce**$(v)\}$.

The vertices in W_t appear consecutively in π. Thus, we can let π_t be the minimal segment of π that contains all the nodes in V_t. Let $i, j \in [k'']$ be such that $\pi_t = t_i, \ldots, t_j$. Now, we define σ_t be the segment $x_i, \ldots, x_j$ of σ_1. Now we prove the claim. By conditions (i) and (ii) in Observation 3.6, $\mathsf{Fake}(t) \cap Z \subseteq V_t$. Clearly, no vertex in $Z \setminus \gamma_{\mathcal{D}'}(t)$ is in V_t. This implies that each vertex in $\mathsf{Fake}(t) \cap Z$ appears in σ_t and no vertex from $Z \setminus \gamma_{\mathcal{D}'}(t)$ appears in σ_t. In turn, this implies that $\sigma_t|_Z$ is the required segment σ'' of $\sigma' = \sigma|_Z$. ◁

Now, having the above two claims, we are ready to prove the lemma. By Claim 4.7, we have that there is a cycle C' such that (i) $E(C') \setminus E(Z) = E(C) \setminus E(Z)$, and (ii) for any $j \in [k']$, the number of edges of $E(C) \cap E(Z)$ with one endpoint in $\{v_1, \ldots, v_j\}$ and other in $\{v_{j+1}, \ldots, v_{k'}\}$ is at most 2. By Claim 4.8, we know that for any $t \in V(T)$, there is a segment

σ'' of σ' such that all vertices in $\mathsf{Fake}(t) \cap Z$ appear in a segment σ'' and no vertex from $Z \setminus \gamma_{\mathcal{D}'}(t)$ appears in σ''. That is, there exist $i, j \in [k']$ such that $\mathsf{Fake}(t) \cap Z \subseteq \{v_i, \ldots, v_j\}$ and $(Z \setminus \gamma_{\mathcal{D}'}(t)) \cap \{v_i, \ldots, v_j\} = \emptyset$. Therefore, by (ii), the number of edges of $E(C') \cap E(Z)$ with one endpoint in $\mathsf{Fake}(t) \cap Z$ and the other in $V(G) \setminus \gamma_{\mathcal{D}'}(t)$ is at most 4. Then, by Observation 4.6, the proof of the lemma is complete. ◀

References

1 J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, and R. Niedermeier. Fixed parameter algorithms for dominating set and related problems on planar graphs. *Algorithmica*, 33(4):461–493, 2002.

2 Jochen Alber, Henning Fernau, and Rolf Niedermeier. Parameterized complexity: exponential speed-up for planar graph problems. *J. Algorithms*, 52(1):26–56, 2004.

3 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. Assoc. Comput. Mach.*, 42(4):844–856, 1995.

4 Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. Assoc. Comput. Mach.*, 41(1):153–180, 1994.

5 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *J. Comput. Syst. Sci.*, 87:119–139, 2017. doi:10.1016/j.jcss.2017.03.003.

6 Zhi-Zhong Chen. Approximation algorithms for independent sets in map graphs. *Journal of Algorithms*, 41:20–40, 2001.

7 Zhi-Zhong Chen, Enory Grigni, and Christos H. Papadimitriou. Planar map graphs. In *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing (STOC '98)*, pages 514–523. ACM, 1998.

8 Zhi-Zhong Chen, Enory Grigni, and Christos H. Papadimitriou. Map graphs. *J. Assoc. Comput. Mach.*, 49(2):127–138, 2002.

9 Zhi-Zhong Chen, Michelangelo Grigni, and Christos H. Papadimitriou. Recognizing Hole-Free 4-Map Graphs in Cubic Time. *Algorithmica*, 45(2):227–262, 2006. doi:10.1007/s00453-005-1184-8.

10 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

11 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 150–159. IEEE, 2011.

12 Erik D. Demaine, Fedor V. Fomin, Mohammadtaghi Hajiaghayi, and Dimitrios M. Thilikos. Fixed-parameter algorithms for (k, r)-center in planar graphs and map graphs. *ACM Trans. Algorithms*, 1(1):33–47, 2005. doi:10.1145/1077464.1077468.

13 Erik D. Demaine, Fedor V. Fomin, Mohammadtaghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential Parameterized Algorithms on Graphs of Bounded Genus and H-minor-free Graphs. *J. ACM*, 52(6):866–893, 2005.

14 Erik D. Demaine and MohammadTaghi Hajiaghayi. Bidimensionality: new connections between FPT algorithms and PTASs. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2005)*, pages 590–601, New York, 2005. ACM-SIAM.

15 Erik D. Demaine and MohammadTaghi Hajiaghayi. The Bidimensionality Theory and Its Algorithmic Applications. *Comput. J.*, 51(3):292–302, 2008. doi:10.1093/comjnl/bxm033.

16 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

17 Frederic Dorn, Eelko Penninkx, Hans L. Bodlaender, and Fedor V. Fomin. Efficient Exact Algorithms on Planar Graphs: Exploiting Sphere Cut Decompositions. *Algorithmica*, 58(3):790–810, 2010. doi:10.1007/s00453-009-9296-1.

18 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar F-Deletion: Approximation, Kernelization and Optimal FPT Algorithms. In *Proceedings of the 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 470–479. IEEE, 2012.

19 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient Computation of Representative Families with Applications in Parameterized and Exact Algorithms. *J. ACM*, 63(4):29, 2016. doi:10.1145/2886094.

20 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Finding, Hitting and Packing Cycles in Subexponential Time on Unit Disk Graphs. *Discrete & Computational Geometry*, January 2019. doi:10.1007/s00454-018-00054-x.

21 Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Bidimensionality and geometric graphs. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1563–1575. SIAM, 2012.

22 Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Excluded grid minors and efficient polynomial-time approximation schemes. *J. ACM*, 65(2):Art. 10, 44, 2018. doi:10.1145/3154833.

23 Qian-Ping Gu and Hisao Tamaki. Improved Bounds on the Planar Branchwidth with Respect to the Largest Grid Minor Size. *Algorithmica*, 64(3):416–453, November 2012. doi:10.1007/s00453-012-9627-5.

24 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity. *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

25 Ioannis Koutis. Faster Algebraic Algorithms for Path and Packing Problems. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP 2008)*, volume 5125 of *Lecture Notes in Computer Science*, pages 575–586, 2008.

26 Ioannis Koutis and Ryan Williams. Algebraic fingerprints for faster algorithms. *Commun. ACM*, 59(1):98–105, 2016. doi:10.1145/2742544.

27 Dániel Marx. The Square Root Phenomenon in Planar Graphs. In *Proceedings of the 40th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 7966 of *Lecture Notes in Computer Science*, page 28. Springer, 2013.

28 N. Robertson, P. Seymour, and R. Thomas. Quickly Excluding a Planar Graph. *Journal of Combinatorial Theory, Series B*, 62(2):323–348, 1994. doi:10.1006/jctb.1994.1073.

29 Mikkel Thorup. Map Graphs In Polynomial Time. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS 1998)*, pages 396–405. IEEE Computer Society, 1998.

30 Ryan Williams. Finding Paths of Length k in $O^*(2^k)$ Time. *Inf. Process. Lett.*, 109(6):315–318, 2009.

The Satisfiability Threshold for Non-Uniform Random 2-SAT

Tobias Friedrich
Algorithm Engineering Group, Hasso Plattner Institute, University of Potsdam, Germany
tobias.friedrich@hpi.de

Ralf Rothenberger
Algorithm Engineering Group, Hasso Plattner Institute, University of Potsdam, Germany
ralf.rothenberger@hpi.de

Abstract

Propositional satisfiability (SAT) is one of the most fundamental problems in computer science. Its worst-case hardness lies at the core of computational complexity theory, for example in the form of NP-hardness and the (Strong) Exponential Time Hypothesis. In practice however, SAT instances can often be solved efficiently. This contradicting behavior has spawned interest in the average-case analysis of SAT and has triggered the development of sophisticated rigorous and non-rigorous techniques for analyzing random structures.

Despite a long line of research and substantial progress, most theoretical work on random SAT assumes a *uniform* distribution on the variables. In contrast, real-world instances often exhibit large fluctuations in variable occurrence. This can be modeled by a *non-uniform* distribution of the variables, which can result in distributions closer to industrial SAT instances.

We study satisfiability thresholds of non-uniform random 2-SAT with n variables and m clauses and with an arbitrary probability distribution $(p_i)_{i\in[n]}$ with $p_1 \geqslant p_2 \geqslant \ldots \geqslant p_n > 0$ over the n variables. We show for $p_1^2 = \Theta\left(\sum_{i=1}^n p_i^2\right)$ that the asymptotic satisfiability threshold is at $m = \Theta\left(\left(1 - \sum_{i=1}^n p_i^2\right) / \left(p_1 \cdot \left(\sum_{i=2}^n p_i^2\right)^{1/2}\right)\right)$ and that it is coarse. For $p_1^2 = o\left(\sum_{i=1}^n p_i^2\right)$ we show that there is a sharp satisfiability threshold at $m = \left(\sum_{i=1}^n p_i^2\right)^{-1}$. This result generalizes the seminal works by Chvatal and Reed [FOCS 1992] and by Goerdt [JCSS 1996].

2012 ACM Subject Classification Theory of computation → Generating random combinatorial structures

Keywords and phrases random SAT, satisfiability threshold, sharpness, non-uniform distribution, 2-SAT

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.61

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1904.02027`.

Funding This paper is partially funded by the project *Skalenfreie Erfüllbarkeit* (project no. 416061626) of the German Research Foundation (DFG).

1 Introduction

Satisfiability of Propositional Formulas (SAT) is one of the most thoroughly researched topics in theoretical computer science. It was one of the first problems shown to be NP-complete by Cook [15] and, independently, by Levin [30]. Today SAT stands at the core of many techniques in modern complexity theory, for example NP-completeness proofs [29] or running time lower bounds assuming the (Strong) Exponential Time Hypothesis [10, 17, 26, 27].

In addition to its importance for theoretical research, Propositional Satisfiability is also famously applied in practice. Despite the theoretical hardness of SAT, many problems arising in practice can be transformed to SAT instances and then solved efficiently with

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 61; pp. 61:1–61:14

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

state-of-the-art solvers. Problems like hard- and software verification, automated planning, and circuit design are often transformed into SAT instances. Such formulas arising from practical and industrial problems are therefore referred to as *industrial SAT instances*. The efficiency of SAT solvers on these instances suggests that they have a structure that makes them easier to solve than the theoretical worst-case.

1.1 Uniform Random k-SAT and the satisfiability threshold conjecture:

Random k-SAT is used to study the average-case complexity of Boolean Satisfiability. In the model, a random formula Φ with n variables, m clauses, and k literals per clause is generated in conjunctive normal form. Each of these formulas has the same uniform probability to be generated. Therefore, we also refer to this model as *uniform random k-SAT*.

One of the most prominent questions related to studying uniform random k-SAT is trying to prove the *satisfiability threshold conjecture*. The conjecture states that for a uniform random k-SAT formula Φ with n variables and m clauses there is a real number r_k such that

$$\lim_{n\to\infty} \Pr\left(\Phi \text{ is satisfiable}\right) = \begin{cases} 1 & m/n < r_k; \\ 0 & m/n > r_k. \end{cases}$$

Chvatal and Reed [11] and, independently, Goerdt [24] proved the conjecture for $k = 2$ and showed that $r_2 = 1$. For larger values of k upper and lower bounds have been established, e. g., $3.52 \leqslant r_3 \leqslant 4.4898$ [18, 25, 28]. Methods from statistical mechanics [32] were used to derive a numerical estimate of $r_3 \approx 4.26$. Coja-Oghlan and Panagiotou [12, 13] showed a bound (up to lower order terms) for $k \geqslant 3$ with $r_k = 2^k \log 2 - \frac{1}{2}(1 + \log 2) \pm o_k(1)$. Finally, Ding, Sly, and Sun [19] proved the exact position of the threshold for sufficiently large values of k. Still, for k between 3 and the values determined by Ding, Sly, and Sun the conjecture remains open.

The satisfiability threshold is also connected to the average hardness of solving instances. For uniform random k-SAT for example, the on average hardest instances are concentrated around the threshold [33].

1.2 Non-Uniform Random SAT

There is a large body of work which considers other random SAT models, e. g. regular random k-SAT [7, 8, 14, 38], random geometric k-SAT [9] and $2 + p$-SAT [1, 35, 34, 36]. However, most of these are not motivated by modeling the properties of industrial instances. One such property is community structure [6], i. e. some variables have a bias towards appearing together in clauses. It is clear by definition that such a bias does not exist in uniform random k-SAT. Therefore, Giráldez-Cru and Levy [23] proposed the Community Attachment Model, which creates random formulas with clear community structure. However, the work of Mull et al. [37] shows that instances generated by this model have exponentially long resolution proofs with high probability, making them hard on average for solvers based on conflict-driven clause learning.

Another important property of industrial instances is their degree distribution. The degree distribution of a formula Φ is a function $f\colon \mathbb{N} \to \mathbb{N}$, where $f(x)$ denotes the fraction of different Boolean variables that appear x times in Φ (negated or unnegated). Instances created with the uniform random k-SAT model have a binomial distribution, while some families of industrial instances appear to follow a power-law distribution [4], i. e. $f(x) \sim x^{-\beta}$, where β is a constant intrinsic to the instance. Therefore, Ansótegui et al. [5] proposed a random k-SAT model with a power-law degree distribution. Empirical studies by the

same authors [2, 3, 4, 5] found that this distribution is beneficial for the runtime of SAT solvers specialized in industrial instances. However, it looks like instances generated with their model can be solved faster than uniform instances, but not as fast as industrial ones: median runtimes around the threshold still seem to scale exponentially for several state-of-the-art solvers [21].

Therefore, we want to consider a generalization of the model by Ansótegui et al. [4]. Our model allows instances with *any* given ensemble of variable distributions instead of only power laws: We draw m clauses of length k at random. For each clause the k variables are drawn with a probability proportional to the n-th distribution in the ensemble, then they are negated independently with a probability of $1/2$ each. This means, the probability ensemble is part of the model, but the number of variables n determines which distribution from the ensemble we actually use. We call this model *non-uniform random k-SAT* and denote it by $\mathcal{D}(n, k, (\vec{p}_x)_{x\in\mathbb{N}}, m)$. Although $\mathcal{D}(n, k, (\vec{p}_x)_{x\in\mathbb{N}}, m)$ cannot capture all properties of industrial instances, e.g. community structure, it can help us to investigate the influence of the degree distribution on the structure and on the computational complexity of such instances in an average-case scenario.

As one of the steps in analyzing this connection, we would like to find out for which ensembles of variable probability distributions an equivalent of the satisfiability threshold conjecture holds in non-uniform random k-SAT. In previous works we already proved upper and lower bounds on the threshold position [20] and showed sufficient conditions on sharpness [22]. In this work we are interested in actually determining the satisfiability threshold for $k = 2$. This is helpful in determining bounds on the satisfiability threshold for higher values of k, since 2-SAT instances appear as parts of k-SAT instances. We already successfully used this approach in [20] to derive lower bounds on the satisfiability threshold for non-uniform random k-SAT with a power-law distribution on the variables.

It has to be noted that Cooper et al. [16] and Levy [31] already studied thresholds in a similar random 2-SAT model. The difference is that in their models the degrees are fixed and the random instances determined in a configuration-model-like fashion, while in our model we only have a sequence of expected degrees from which the actual degrees might deviate. Note that it is not clear if the satisfiability thresholds in these two models coincide if they use the same sequence of (expected) degrees. Cooper et al. derive the position of the satisfiability threshold in their model if the maximum degree is sufficiently bounded. Levy only shows necessary conditions on unsatisfiability in the model of Cooper et al. This is not enough to derive the actual threshold position. In contrast, our result allows us to determine the position of the satisfiability threshold for *any* probability ensemble in the model we consider.

1.3 Our Results

We investigate the position and behavior of the satisfiability threshold for non-uniform random 2-SAT. That is, we fix the number of variables n and the variable distribution $\vec{p}_n$ from the ensemble and vary the number of clauses $m(n)$. To this end, we use the following definition and say that a function $m^*(n)$ is an *asymptotic satisfiability threshold* if

$$\Pr_{\Phi\sim\mathcal{D}(n,k,(\vec{p}_x)_{x\in\mathbb{N}},m)}(\Phi \text{ satisfiable}) = \begin{cases} 1-o(1) & \text{if } m(n) = o(m^*(n)) \\ o(1) & \text{if } m(n) = \omega(m^*(n)). \end{cases}$$

We also say that an asymptotic satisfiability threshold $m^*(n)$ is *sharp* if for all $\varepsilon > 0$

$$\Pr_{\Phi\sim\mathcal{D}(n,k,(\vec{p}_x)_{x\in\mathbb{N}},m)}(\Phi \text{ satisfiable}) = \begin{cases} 1-o(1) & \text{if } m(n) \leqslant (1-\varepsilon)\cdot m^*(n) \\ o(1) & \text{if } m(n) \geqslant (1+\varepsilon)\cdot m^*(n). \end{cases}$$

If an asymptotic threshold is not sharp, we call it *coarse*.

Let $\vec{p}_n = (p_1, p_2, \ldots, p_n)$ be the variable probability distribution we use. W. l. o. g. we assume $p_1 \geqslant p_2 \geqslant \ldots \geqslant p_n$. We are going to show that there are three cases depending on $\vec{p}_n$:

1. If $p_1^2 = \Theta\left(\sum_{i=1}^n p_i^2\right)$ and $p_2^2 = \Theta\left(\sum_{i=2}^n p_i^2\right)$, then we can show that the asymptotic satisfiability threshold is at $m = \Theta\left(q_{\max}^{-1}\right)$, where $q_{\max} = \Theta\left((p_1 \cdot p_2) \,/\, \left(1 - \sum_{i=1}^n p_i^2\right)\right)$ is the maximum clause probability. We can also show that this threshold is coarse. The coarseness stems from the emergence of an unsatisfiable sub-formula of size 4, which contains only the two most probable variables.
2. If $p_1^2 = \Theta\left(\sum_{i=1}^n p_i^2\right)$ and $p_2^2 = o\left(\sum_{i=2}^n p_i^2\right)$, then the asymptotic threshold is at $m = \Theta\left(\left(1 - \sum_{i=1}^n p_i^2\right) \,/\, \left(p_1 \cdot \left(\sum_{i=2}^n p_i^2\right)^{1/2}\right)\right)$ and it is again coarse. This time the coarseness stems from the emergence of an unsatisfiable sub-formula with 3 variables and 4 clauses.
3. If $p_1^2 = o\left(\sum_{i=1}^n p_i^2\right)$, then there is a sharp threshold at exactly $m = 1/\left(\sum_{i=1}^n p_i^2\right)$.

Note that these three cases give us a complete dichotomy of coarseness and sharpness for the satisfiability threshold of non-uniform random 2-SAT. This result generalizes the seminal works by Chvatal and Reed [11] and by Goerdt [24] to arbitrary variable probability distributions and includes their findings as a special case (c. f. Section 6). We summarize our findings in the following theorem.

▶ **Theorem 1.1.** *Let $\mathcal{D}(n, 2, (\vec{p}_x)_{x\in\mathbb{N}}, m)$ be the non-uniform random 2-SAT model with n variables, m clauses, and an ensemble of probability distributions $(\vec{p}_x)_{x\in\mathbb{N}}$. Let $\vec{p}_n = (p_1, p_2, \ldots, p_n)$ be the n-th distribution from the ensemble. W. l. o. g. let $p_1 \geqslant p_2 \geqslant \ldots \geqslant p_n$. If $p_1^2 = o\left(\sum_{i=1}^n p_i^2\right)$, then $\mathcal{D}(n, 2, (\vec{p}_x)_{x\in\mathbb{N}}, m)$ has a sharp satisfiability threshold at $m = \left(\sum_{i=1}^n p_i^2\right)^{-1}$. Otherwise, $\mathcal{D}(n, 2, (\vec{p}_x)_{x\in\mathbb{N}}, m)$ has a coarse satisfiability threshold at $m = \Theta\left(\left(1 - \sum_{i=1}^n p_i^2\right) \,/\, \left(p_1 \cdot \left(\sum_{i=2}^n p_i^2\right)^{1/2}\right)\right)$.*

1.4 Techniques

For the sharp threshold result, we only show the conditions on sharpness. These also imply the existence of an asymptotic threshold. For the coarse threshold results, however, we first have to show the existence of an asymptotic threshold at some number of clauses $m^*(n)$. Then, we have to show that for some range of constants $\varepsilon \in [\varepsilon_1, \varepsilon_2]$ the probability to generate a satisfiable instance at $\varepsilon \cdot m^*(n)$ is a constant bounded away from zero and one.

We extend and generalize the proof ideas of Chvatal and Reed [11]. In order to show a lower bound on the threshold, we investigate the existence of bicycles. Bicycles were introduced by Chvatal and Reed. They are sub-formulas which appear in every unsatisfiable formula. We can show with a first moment argument, that these do not appear below a certain number of clauses, thus making formulas satisfiable.

In order to show an upper bound on the threshold, we investigate the existence of snakes. Snakes are unsatisfiable sub-formulas and have also been introduced by Chvatal and Reed. We can show with a second-moment argument that snakes of certain sizes do appear above a certain number of clauses, thus making formulas unsatisfiable. However, we need to be careful and distinguish more possibilities of partially mapping snakes onto each other than in the uniform case. Unfortunately, this method does not work if $p_1^2 = \Theta\left(\sum_{i=1}^n p_i^2\right)$ and $p_2^2 = \Theta\left(\sum_{i=2}^n p_i^2\right)$. In that case we lower-bound the probability that an unsatisfiable sub-formula containing only the two most-probable variables exists. This can be done with a simple inclusion-exclusion argument and the resulting lemma also work for $k \geqslant 3$.

2 Preliminaries

We analyze non-uniform random k-SAT on n variables and m clauses. We denote by $X_1, \dots, X_n$ the Boolean variables. A clause is a disjunction of k literals $\ell_1 \vee \dots \vee \ell_k$, where each literal assumes a (possibly negated) variable. For a literal ℓ_i let $|\ell_i|$ denote the variable of the literal. A formula Φ in conjunctive normal form is a conjunction of clauses $c_1 \wedge \dots \wedge c_m$. We conveniently interpret a clause c both as a Boolean formula and as a set of literals. We say that Φ is satisfiable if there exists an assignment of variables $X_1, \dots, X_n$ such that the formula evaluates to 1. Now let $(\vec{p}_n)_{n\in\mathbb{N}}$ be an ensemble of probability distributions, where $\vec{p}_n = (p_{n,1}, p_{n,2}, \dots, p_{n,n})$ is a probability distribution over n variables with $\Pr(X = X_i) = p_{n,i} =: p_n(X_i)$.

▶ **Definition 2.1** (Clause-Drawing Non-Uniform Random k-SAT). *Let m, n, k be given, and consider any ensemble of probability distributions $(\vec{p}_n)_{n\in\mathbb{N}}$, where $\vec{p}_n = (p_{n,1}, p_{n,2}, \dots, p_{n,n})$ is a probability distribution over n variables with $\sum_{i=1}^n p_{n,i} = 1$. The* clause-drawing non-uniform random k-SAT (non-uniform random k-SAT) *model $\mathcal{D}(n, k, (\vec{p}_x)_{x\in\mathbb{N}}, m)$ constructs a random SAT formula Φ by sampling m clauses independently at random. Each clause is sampled as follows:*

1. *Select k variables independently at random from the distribution $\vec{p}_n$. Repeat until no variables coincide.*
2. *Negate each of the k variables independently at random with probability $1/2$.*

For the sake of simplicity and since we will always only consider one distribution from the ensemble, we will omit the index n throughout the paper, e. g. the probability distribution $\vec{p}_n$ will be denoted as $(p_1, p_2, \dots, p_n)$. W. l. o. g. we will assume $p_1 \geqslant p_2 \geqslant \dots p_n$.

The clause-drawing non-uniform random k-SAT model is equivalent to drawing each clause independently at random from the set of all k-clauses which contain no variable more than once. The probability to draw a clause c over n variables is then

$$q_c := \frac{\prod_{\ell\in c} p(|\ell|)}{2^k \sum_{J\in\mathcal{P}_k(\{1,2,\dots,n\})} \prod_{j\in J} p_j}, \tag{2.1}$$

where $\mathcal{P}_k(\cdot)$ denotes the set of cardinality-k elements of the power set. The factor 2^k in the denominator comes from the different possibilities to negate variables. Note that $k! \sum_{J\in\mathcal{P}_k(\{1,2,\dots,n\})} \prod_{j\in J} p_{n,j}$ is the probability of choosing a k-clause that contains no variable more than once. We can now write

$$q_c = C \frac{k!}{2^k} \prod_{X\in S} p_n(X), \tag{2.2}$$

where we define $C := 1/\left(k! \cdot \sum_{J\in\mathcal{P}_k(\{1,2,\dots,n\})} \prod_{j\in J} p_{n,j}\right)$. For $k = 2$ it holds that $C = 1/\left(1 - \left(\sum_{i=1}^n p_i^2\right)\right)$. Hiding this factor in C makes clause probabilities easier to handle. Throughout the paper we let $q_{\max}$ denote the maximum clause probability as defined in equation (2.2). In Section 3 and Section 4 we will assume $q_{\max} = o(1)$. The case $q_{\max} = \Theta(1)$ will be handled in Section 5. Note that this case can only happen for $p_1^2 = \Theta\left(\sum_{i=1}^n p_i^2\right)$ and $p_2^2 = \Theta\left(\sum_{i=2}^n p_i^2\right)$.

3 Bi-Cycles and a Lower Bound on the Satisfiability Threshold

Chvatal and Reed [11] define the following sub-structure of 2-SAT formulas and show that every unsatisfiable 2-CNF contains this substructure.

▶ **Definition 3.1** (bi-cycle). *We define a bicycle of length t to be a sequence of $t+1$ clauses of the form*

$$(u, w_1), (\overline{w}_1, w_2), \ldots, (\overline{w}_{t-1}, w_t), (\overline{w}_t, v),$$

where $w_1, \ldots, w_t$ are literals of distinct variables and $u, v \in \{w_1, \ldots, w_t, \overline{w}_1, \ldots, \overline{w}_t\}$.

To lower-bound the probability for a random 2-CNF to be satisfiable it is therefore sufficient to upper-bound the probability that such a formula contains a bicycle. This is done in the following two lemmas. Their proofs are oriented along the lines of the proof of Theorem 3 from [11].

▶ **Lemma 3.1.** *Consider a non-uniform random 2-SAT formula Φ with $p_1^2 = o\left(\sum_{i=1}^n p_i^2\right)$. Then, Φ is satisfiable with probability at least $1 - o(1)$ for a number of clauses $m < (1-\varepsilon)\left(\sum_{i=1}^n p_i^2\right)^{-1}$, where $\varepsilon > 0$ is a constant.*

Proof. To show this result, we show that the expected number of bicycles is $o(1)$ for the setting we consider. The result then follows by Markov's inequality.

First, we fix a set $S \subseteq [n]$ of variables to appear in a bicycle with $|S| = t \geqslant 2$. The probability that a *specific* bicycle B with these variables appears in Φ is

$$\Pr(B \text{ in } \Phi) = \underbrace{\binom{m}{t+1}(t+1)!}_{\text{positions of } B \text{ in } \Phi} \cdot \Pr(u \vee w_1) \cdot \Pr(\overline{w}_t \vee v) \prod_{h=1}^{t-1} \Pr(\overline{w}_h \vee w_{h+1}).$$

For literals w_i over variables x_i it holds that

$$\Pr(w_j \vee w_i) = \frac{C}{2} p_i \cdot p_j,$$

where $1 \leqslant C = \left(1 - \sum_{i=1}^n p_i^2\right)^{-1} = 1 + o(1)$, since $\sum_{i=1}^n p_i^2 = o(1)$ due to the requirement $p_1^2 = o\left(\sum_{i=1}^n p_i^2\right)$. There are at most $t!$ possibilities to arrange the t variables in a bicycle and 2^t possibilities to choose literals from the t variables. For the probability that *any* bicycle with the variables from S appears in Φ it now holds that

$$\Pr(S\text{-bicycle in } \Phi) \leqslant m^{t+1} \cdot t! \cdot 2^t \cdot \left(\frac{C}{2}\right)^{t+1} \cdot \prod_{i \in S} p_i^2 \left(2 \cdot \sum_{i \in S} p_i\right)^2$$

where the last factor accounts for the possibilities to choose u and v. It now holds that

$$\begin{aligned}
\Pr(\Phi \text{ contains a bicycle}) &\leqslant \sum_{t=2}^{n} \sum_{S \subseteq \mathcal{P}_t(V)} m^{t+1} \cdot t! \cdot 2^t \cdot \left(\frac{C}{2}\right)^{t+1} 2^2 \cdot \prod_{i \in S} p_i^2 \left(\sum_{i \in S} p_i\right)^2 \\
&\leqslant 2 \cdot \sum_{t=2}^{n} (C \cdot m)^{t+1} \cdot t! \cdot t^2 \cdot p_1^2 \cdot \sum_{S \subseteq \mathcal{P}_t(V)} \prod_{i \in S} p_i^2 \\
&\leqslant 2 \cdot \sum_{t=2}^{n} (C \cdot m)^{t+1} \cdot t^2 \cdot p_1^2 \cdot \left(\sum_{i \in S} p_i^2\right)^t \\
&= o\left(2 \cdot \sum_{t=2}^{n} \left(C \cdot m \left(\sum_{i \in S} p_i^2\right)\right)^{t+1} \cdot t^2\right),
\end{aligned}$$

where we used $\sum_{i\in S} p_i \leqslant t \cdot p_1$ in the second, $\sum_{S\subseteq\mathcal{P}_t(V)} \prod_{i\in S} p_i^2 \leqslant \frac{1}{t!} \cdot \left(\sum_{i\in S} p_i^2\right)^t$ in the third line, and the requirement $p_1^2 = o\left(\sum_{i=1}^n p_i^2\right)$ in the fourth line. It is obvious that this probability is o(1) as soon as the sum becomes a constant. This holds for $m < (1-\varepsilon)\left(\sum_{i=1}^n p_i^2\right)^{-1} < \left(C \cdot \sum_{i=1}^n p_i^2\right)^{-1}$, where $\varepsilon > 0$ is a constant. ◀

It has to be noted that in the former lemma we ignored the factor C in our bound. We can do this, since for $p_1^2 = o\left(\sum_{i=1}^n p_i^2\right)$ it always is $1+o(1)$ and does not make a difference for sharpness due to our definition. In the case of $p_1^2 = \Theta\left(\sum_{i=1}^n p_i^2\right)$, we can show the following result with a similar proof, but now we have to take C into account, since it might become super-constant. Also we have to do a case distinction between the terms with $p_1 \in S$ and $p_1 \notin S$ to get $\sum_{S\subseteq\mathcal{P}_t(V)} \left(\prod_{i\in S} p_i^2\right) \cdot \left(\sum_{i\in S} p_i\right)^2 = \mathcal{O}\left(t^3 \cdot p_1^4 \cdot \frac{1}{t!} \cdot \left(\sum_{i=2}^n p_i^2\right)^{t-1}\right)$. See the full version of the paper for the whole proof.

▶ **Lemma 3.2.** *Consider a non-uniform random 2-SAT formula Φ with $p_1^2 = \Theta\left(\sum_{i=1}^n p_i^2\right)$ and $q_{\max} = o(1)$. Then, Φ is satisfiable with probability at least $1 - o(1)$ for a number of clauses $m = o\left(\left(C \cdot p_1 \cdot \left(\sum_{i=2}^n p_i^2\right)^{1/2}\right)^{-1}\right)$. Also, there is a constant $\varepsilon \in (0,1)$ such that Φ is satisfiable with a positive constant probability for a number of clauses $m \leqslant (1-\varepsilon)\left(C \cdot p_1 \cdot \left(\sum_{i=2}^n p_i^2\right)^{1/2}\right)^{-1}$.*

Note that this lemma captures both cases for $p_1^2 = \Theta\left(\sum_{i=1}^n p_i^2\right)$. If also $p_2^2 = \Theta\left(\sum_{i=2}^n p_i^2\right)$, then $\left(C \cdot p_1 \cdot \left(\sum_{i=2}^n p_i^2\right)^{1/2}\right)^{-1} = \Theta\left(q_{\max}^{-1}\right)$ is the asymptotic threshold as we stated in the introduction. The case $q_{\max} = \Theta(1)$ has to be excluded, since for that case the asymptotic threshold is a constant. The above lemma might then give us a value so small that the ranges where we can lower- and upper-bound satisfiability to constants away from zero resp. one do not overlap. Thus, this case is handled separately in Section 5.

4 Snakes and an Upper Bound on the Satisfiability Threshold

The two lemmas from the previous section provided a lower bound on the satisfiability threshold for non-uniform random 2-SAT. By using the second moment method, we can also derive an upper bound on the threshold. Again, this proof is inspired by Chvatal and Reed [11, Theorem 4], who provide us with the following definition.

▶ **Definition 4.1** (snake). *A* snake *of size t is a sequence of literals $w_1, w_2, \dots, w_{2t-1}$ over distinct variables. Each snake A is associated with a set F_A of $2t$ clauses $(\overline{w}_i, w_{i+1})$, $0 \leqslant i \leqslant 2t-1$, such that $w_0 = w_{2t} = \overline{w}_t$.*

We will also call the variable $|w_t|$ of a snake its *central* variable. Note that the set of clauses F_A defined by a snake A is unsatisfiable. Also, the snakes $(w_1, \dots, w_{t-1}, w_t, w_{t+1}, \dots, w_s)$, $(\overline{w_{t-1}}, \overline{w_{t-2}} \dots, \overline{w_1}, w_t, w_{t+1}, \dots, w_s)$, $(w_1, \dots, w_{t-1}, w_t, \overline{w_s}, \overline{w_{s-1}} \dots, \overline{w_{t+1}})$ and $(\overline{w_{t-1}}, \overline{w_{t-2}} \dots, \overline{w_1}, w_t \overline{w_s}, \overline{w_{s-1}} \dots, \overline{w_{t+1}})$ create the same set of formulas.

The *variable-variable incidence graph (VIG)* for a formula Φ is a simple graph $G_\Phi = (V_\Phi, E_\Phi)$ with V_Φ consisting of all variables appearing in Φ and two variables being connected by an edge if they appear together in at least one clause of Φ. An example for a snake's VIG can be seen in Figure 1. This representation will come in handy later in the proofs of Lemmas 4.4 and 4.6.

In order to show our upper bounds, we will prove that snakes of a certain length t appear with sufficiently high probability in a random formula $\Phi \sim \mathcal{D}\left(n, k, (\vec{p}_x)_{x\in\mathbb{N}}, m\right)$ To this end we utilize the second moment method: If $X \geqslant 0$ is a random variable with finite variance, then

$$\Pr(X > 0) \geqslant \frac{\mathbb{E}[X]^2}{\mathbb{E}[X^2]}.$$

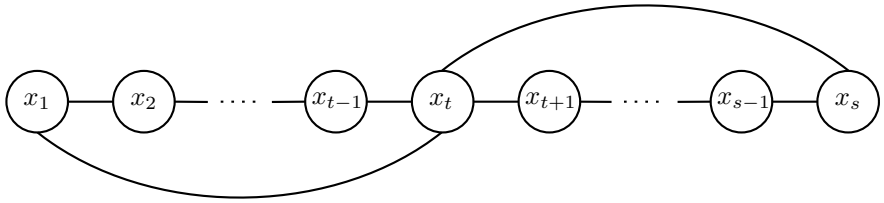


Figure 1 Variable-variable-incidence graph of a snake $w_1, w_2, \ldots, w_s$ where $|w_i| = x_i$ (the variable of the literal w_i) for $1 \leqslant i \leqslant s = 2t - 1$.

We define the following indicator variables for each snake A of size t

$$X_A = \begin{cases} 1 & \text{if } F_A \text{ appears exactly once in } \Phi \\ 0 & \text{otherwise} \end{cases}$$

and their sum $X_t = \sum_A X_A$. For carefully chosen t we will show $\mathbb{E}\left[X_t^2\right] = \mathcal{O}(\mathbb{E}[X_t]^2)$ to show a coarse and $\mathbb{E}\left[X_t^2\right] = (1 + o(1)) \cdot (\mathbb{E}[X_t]^2)$ to show a sharp threshold. This implies a constant resp. $1 - o(1)$ probability to be unsatisfiable due to the second moment method. In the case of $p_1^2 = o(\sum_{i=1}^n p_i^2)$, we will chose $t = \Theta\left(\log^2 f(n)\right)$, where we define $f(n) = \left(\sum_{i=1}^n p_i^2\right)/p_1^2$. For $p_1^2 = \Theta(\sum_{i=1}^n p_i^2)$ and $p_2^2 = o(\sum_{i=2}^n p_i^2)$ we choose $t = 2$. We only want to use the method for these two cases. The third case with $p_1 = \Theta\left(\sum_{i=1}^n p_i^2\right)$ and $p_2 = \Theta\left(\sum_{i=2}^n p_i^2\right)$ will be handled with the more general Lemma 4.7.

Now, if we want to use the second moment method, we first have to ensure that the expected number of snakes of a certain size is large enough. The following lemma provides a lower bound on this expected number. Due to space limitations, its proof can be found in the full version.

▶ **Lemma 4.1.** *Let X_t be the number of snakes of size $s + 1 = 2t$ whose associated formulas appear exactly once in a non-uniform random 2-SAT formula. Then it holds that*

$$\mathbb{E}[X_t] \geqslant \frac{1}{2}(m - 2t)^{2t} \cdot C^{2t} \cdot e^{-(m-2t)\frac{2t \cdot q_{\max}}{1-2t \cdot q_{\max}}} \cdot \left(\sum_{i=1}^n p_i^4\right) \cdot \left(\sum_{i=2}^n p_i^2 - (2t-2) \cdot p_2^2\right)^{2t-2}.$$

In order to use the second moment method we have to show that this expected value is at least a constant if we want to show a coarse threshold and asymptotically bigger than a constant if we want to show a sharp threshold. Hence, the following lemmas give lower bounds on $\mathbb{E}[X_t]$ for the first two cases and the respective ranges of t we consider. Again, their proofs can be found in the full version of this paper.

▶ **Lemma 4.2.** *Let X_t be the number of snakes of size t that appear exactly once in a non-uniform random 2-SAT formula with $p_1^2 = o(\sum_{i=1}^n p_i^2)$ and $m = (1 + \varepsilon)\left(\sum_{i=1}^n p_i^2\right)^{-1}$ for some $\varepsilon > 0$. Then it holds that*

$$\mathbb{E}[X_t] \geqslant (1 - o(1)) \cdot m^{2t} \left(\sum_{i=1}^n p_i^4\right) \cdot \left(\sum_{i=1}^n p_i^2\right)^{2t-2} = \omega(1)$$

if $t = o\left(\sqrt{f(n)}\right) \cap \omega(\log f(n))$, where $f(n) = \left(\sum_{i=1}^n p_i^2\right)/p_1^2$.

▶ **Lemma 4.3.** *Let X_t be the number of snakes of size t that appear exactly once in a non-uniform random 2-SAT formula with $p_1^2 = \Theta(\sum_{i=1}^n p_i^2)$ and $p_2^2 = o(\sum_{i=2}^n p_i^2)$. For $t = 2$ and $m = \Omega\left(\left(C \cdot p_1 \cdot \left(\sum_{i=2}^n p_i^2\right)^{1/2}\right)^{-1}\right) \cap o\left(q_{\max}^{-1}\right)$ it holds that*

$$\mathbb{E}[X_2] \geqslant (1 - o(1)) \cdot m^4 \cdot C^4 \cdot p_1^4 \cdot \left(\sum_{i=1}^n p_i^2\right)^2.$$

Furthermore,

$$\mathbb{E}[X_2] = \begin{cases} \Omega(1) & , m = \Theta\left(\left(C \cdot p_1 \left(\sum_{i=2}^n p_i^2\right)^{1/2}\right)^{-1}\right) \text{ and} \\ \omega(1) & , m = \omega\left(\left(C \cdot p_1 \left(\sum_{i=2}^n p_i^2\right)^{1/2}\right)^{-1}\right) \cap o\left((q_{\max})^{-1}\right). \end{cases}$$

Now we are ready to prove an upper bound on the non-uniform random 2-SAT threshold. To get to know the proof technique, we start with the much simpler case $p_1^2 = \Theta\left(\sum_{i=1}^n p_i^2\right)$ and $p_2^2 = o\left(\sum_{i=2}^n p_i^2\right)$. The proof contains a small case distinction depending on how the shared clauses of two snakes A and B influence $\Pr(X_A \wedge X_B)$. The next lemma establishes that there is a regime of m where random formulas are unsatisfiable with a positive constant probability. Its proof is in the full version.

▶ **Lemma 4.4.** *Consider a non-uniform random 2-SAT formula Φ with $p_1^2 = \Theta\left(\sum_{i=1}^n p_i^2\right)$ and $p_2^2 = o\left(\sum_{i=2}^n p_i^2\right)$. Then Φ is unsatisfiable with positive constant probability for $m = \Theta\left(\left(C \cdot p_1 \left(\sum_{i=2}^n p_i^2\right)^{1/2}\right)^{-1}\right)$.*

The following lemma complements the former one, showing that above that regime of m random formulas are unsatisfiable with probability $1 - o(1)$. Its proof is very similar and is therefore omitted. It can be found in the full version of this work.

▶ **Lemma 4.5.** *Consider a non-uniform random 2-SAT formula Φ with $p_1^2 = \Theta\left(\sum_{i=1}^n p_i^2\right)$ and $p_2^2 = o\left(\sum_{i=2}^n p_i^2\right)$. Then Φ is unsatisfiable with probability $1 - o(1)$ for $m = \omega\left(\left(C \cdot p_1 \left(\sum_{i=2}^n p_i^2\right)^{1/2}\right)^{-1}\right)$.*

The former two lemmas together with Lemma 3.2 establish that in the case of $p_1^2 = \Theta\left(\sum_{i=1}^n p_i^2\right)$ and $p_2^2 = o\left(\sum_{i=2}^n p_i^2\right)$ the asymptotic threshold is at $m = \Theta\left(\left(C \cdot p_1 \left(\sum_{i=2}^n p_i^2\right)^{1/2}\right)^{-1}\right)$ and that it is coarse.

We now turn to the case $p_1^2 = \Theta\left(\sum_{i=1}^n p_i^2\right)$. Again, we have to consider different possibilities for the shared clauses of snakes A and B to influence $\Pr(X_A \wedge X_B)$. In the proofs of the former case this was rather easy, since we only considered the smallest possible snakes of size 3. Now the distinction becomes a bit more difficult. We will distinguish several cases: If the number of shared clauses is at least $t - 1$ then $\Pr(X_A \wedge X_B)$ is by roughly a factor of $(1 + \varepsilon)^t$ smaller than $\mathbb{E}[X_t]^2$. If the shared clauses form at least two connected sub-formulas, then there are enough variable appearances pre-defined for B to make $\Pr(X_A \wedge X_B)$ sufficiently small. The last case is that there is only one connected sub-formula, which is a lot smaller than $t - 1$. In that case we have to carefully consider what happens to the central variable from B, since this variable appears most times in B and the many appearances take degrees of freedom away from other variables, therefore making $\Pr(X_A \wedge X_B)$ small. The whole proof of the lemma can be found in the full version.

▶ **Lemma 4.6.** *Consider a non-uniform random 2-SAT formula Φ with $p_1^2 = o\left(\sum_{i=1}^n p_i^2\right)$. Then Φ is unsatisfiable with probability $1 - o(1)$ for $m > (1+\varepsilon) \cdot \left(\sum_{i=1}^n p_i^2\right)^{-1}$, where $\varepsilon > 0$ is a constant.*

Lemma 4.6 and Lemma 3.1 now establish the existence of a sharp threshold at $m = \left(\sum_{i=1}^n p_i^2\right)^{-1}$.

Now we still have to consider the case $p_1^2 = \Theta\left(\sum_{i=1}^n p_i^2\right)$ and $p_2^2 = \Theta\left(\sum_{i=2}^n p_i^2\right)$. In the following lemma, we give a lower bound on the probability to be unsatisfiable by showing the existence of an unsatisfiable sub-formula consisting only of the two most-probable variables. The lemma generally holds for $k \geqslant 2$, but it especially serves our purpose of considering the remaining case. The proof uses a simple inlcusion-exclusion argument and can be found in the full version.

▶ **Lemma 4.7.** *Consider a non-uniform random k-SAT formula Φ with $q_{\max} = o(1)$. Then Φ is unsatisfiable with probability at least*

$$\sum_{l=0}^{2^k} \left(\binom{2^k}{l} (-1)^l \left(1 - l \cdot q_{\max}\right)^m \right) \geqslant \left(1 - \exp\left(-q_{\max} \cdot m\right)\right)^{2^k} - q_{\max}^2 \cdot 2^{2k} \cdot m \cdot \left(1 + \exp\left(-q_{\max} \cdot m\right)\right)^{2^k}.$$

The former lemma now yields the following corollary.

▶ **Corollary 4.1.** *Consider a non-uniform random k-SAT formula Φ with $q_{\max} = o(1)$. Then*

1. $\Pr(\Phi$ *unsatisfiable*$) = \Omega(1)$ *for* $m = \Theta\left(q_{\max}^{-1}\right)$ *and*
2. $\Pr(\Phi$ *unsatisfiable*$) = 1 - o(1)$ *for* $m = \omega(q_{\max}^{-1})$.

In the second case the result follows from Lemma 4.7 for $m = \omega(q_{\max}^{-1}) \cap o(q_{\max}^{-2})$ and by monotonicity of the satisfiability probability in m. This corollary together with Lemma 3.2 establishes the existence of a coarse threshold at $m = \Theta\left(\left(C \cdot p_1 \left(\sum_{i=2}^n p_i^2\right)^{1/2}\right)^{-1}\right) = \Theta\left(q_{\max}^{-1}\right)$ for non-uniform random 2-SAT with $p_1^2 = \Theta\left(\sum_{i=1}^n p_i^2\right)$, $p_2^2 = \Theta\left(\sum_{i=2}^n p_i^2\right)$.

5 Constant Clause Probabilities

We assumed $q_{\max} = o(1)$ throughout the paper. For the sake of completeness we still have to take care of the case $q_{\max} = \Theta(1)$. It is easy to see that for $\Phi \sim \mathcal{D}\left(n, 2, (\vec{p}_x)_{x \in \mathbb{N}}, m\right)$ and a constant $m \geqslant 4$ it holds that $\Pr(\Phi \text{ unsatisfiable}) \geqslant q_{\max}^m$, since this is the probability of an unsatisfiable instance, where the most probable clause appears with all four combinations of signs and then one of these clauses appears an additional $m - 4$ times. Similarly, $\Pr(\Phi \text{ satisfiable}) \geqslant q_{\max}^m$, as this is the probability of a satisfiable instance, where the same most probable clause appears m times with the same sign. Since $0 < q_{\max} \leqslant 1/4$ is a constant, the probability is a constant bounded away from zero and one. It remains to show that Φ is unsatisfiable with probability $1 - o(1)$ for $m = \omega(1)$. The following lemma establishes this. Again, this lemma also holds for $k \geqslant 2$ in general.

▶ **Lemma 5.1.** *Consider a non-uniform random k-SAT formula Φ. Then Φ is unsatisfiable with probability at least*

$$2 - \left(1 + \exp\left(-q_{\max} \cdot m\right)\right)^{2^k}.$$

Proof. As in Lemma 4.7, it holds that

$$\Pr(\Phi \text{ unsat}) \geqslant \sum_{l=0}^{2^k} \left(\binom{2^k}{l} (-1)^l (1 - l \cdot q_{\max})^m \right).$$

We can now estimate

$$\begin{aligned}
\sum_{l=0}^{2^k} \left(\binom{2^k}{l} (-1)^l (1 - l \cdot q_{\max})^m \right) &\geqslant 1 - \sum_{l=1}^{2^k} \left(\binom{2^k}{l} (1 - l \cdot q_{\max})^m \right) \\
&\geqslant 1 - \sum_{l=1}^{2^k} \left(\binom{2^k}{l} \exp\left(-m \cdot \frac{l \cdot q_{\max}}{1 - l \cdot q_{\max}} \right)^m \right) \\
&\geqslant 1 - \sum_{l=1}^{2^k} \left(\binom{2^k}{l} \exp\left(-m \cdot l \cdot q_{\max} \right) \right) \\
&= 2 - \left(1 + \exp\left(-m \cdot q_{\max}\right)\right)^{2^k}
\end{aligned}$$

◀

For $q_{\max} = \Theta(1)$ and $m = \omega\left(q_{\max}^{-1}\right)$ this lemma implies $\Pr(\Phi \text{ unsatisfiable}) \geqslant 1 - o(1)$. All lemmas together now imply our main theorem.

▶ **Theorem 1.1.** *Let $\mathcal{D}(n, 2, (\vec{p}_x)_{x\in\mathbb{N}}, m)$ be the non-uniform random 2-SAT model with n variables, m clauses, and an ensemble of probability distributions $(\vec{p}_x)_{x\in\mathbb{N}}$. Let $\vec{p}_n = (p_1, p_2, \dots, p_n)$ be the n-th distribution from the ensemble. W. l. o. g. let $p_1 \geqslant p_2 \geqslant \dots \geqslant p_n$. If $p_1^2 = o\left(\sum_{i=1}^n p_i^2\right)$, then $\mathcal{D}(n, 2, (\vec{p}_x)_{x\in\mathbb{N}}, m)$ has a sharp satisfiability threshold at $m = \left(\sum_{i=1}^n p_i^2\right)^{-1}$. Otherwise, $\mathcal{D}(n, 2, (\vec{p}_x)_{x\in\mathbb{N}}, m)$ has a coarse satisfiability threshold at $m = \Theta\left(\left(1 - \sum_{i=1}^n p_i^2\right) / \left(p_1 \cdot \left(\sum_{i=2}^n p_i^2\right)^{1/2}\right)\right)$.*

6 Example Applications of our Theorem

We will now show on some examples how our main theorem can be applied.

6.1 Uniform Distribution

The simplest distribution we can apply our theorem to is the uniform distribution, i. e. $\vec{p}_n = \left(\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}\right)$ for all $n \in \mathbb{N}$. It holds that $p_1^2 = \frac{1}{n^2}$ and $\sum_{i=1}^n p_i^2 = \frac{1}{n}$. Thus, Theorem 1.1 implies a sharp threshold at $m^*(n) = n$ for all $n \in \mathbb{N}$. This reproves [11, 24] the satisfiability threshold conjecture for $k = 2$, since these sharp threshold are all at $m^*(n) = n$.

6.2 Power Law Distribution

Another ensemble of distributions we can choose are power-law distributions, i.e. we consider the power law random 2-SAT model introduced by Ansótegui et al. [5]. Thus, for a constant $\beta > 2$ we choose $\vec{p}_n = (p_1, p_2, \dots, p_n)$ with

$$p_i = \frac{(n/i)^{\frac{1}{\beta-1}}}{\left(\sum_{j=1}^n (n/j)^{\frac{1}{\beta-1}}\right)}.$$

It already holds that $p_1 \geqslant p_2 \geqslant \dots \geqslant p_n$. Now it is an easy exercise to show that

$$\left(\sum_{j=1}^n (n/j)^{\frac{1}{\beta-1}}\right) = (1 - o(1)) \cdot \frac{\beta - 1}{\beta - 2}.$$

Furthermore

$$p_1^2 = (1 \pm o(1)) \cdot \left(\frac{\beta - 1}{\beta - 2}\right)^2 \cdot n^{-2\frac{\beta-2}{\beta-1}}.$$

Finally, one can show that

$$\sum_{i=1}^{n} p_i^2 = \begin{cases} (1 \pm o(1)) \cdot \frac{(\beta-2)^2}{(\beta-3)\cdot(\beta-1)} \cdot n^{-2\frac{\beta-2}{\beta-1}} & \text{for } \beta < 3 \\ (1 \pm o(1)) \cdot \frac{1}{4} \cdot \frac{\ln n}{n} & \text{for } \beta = 3 \\ (1 \pm o(1)) \cdot \frac{(\beta-2)^2}{(\beta-3)\cdot(\beta-1)} \cdot n^{-1} & \text{for } \beta > 3. \end{cases}$$

Thus, applying our theorem we can see that for $\beta < 3$ there is a coarse threshold at $m = \Theta\left(n^{-2\frac{\beta-2}{\beta-1}}\right)$, since $p_1^2 = \Theta\left(\sum_{i=1}^{n} p_i^2\right) = \Theta\left(n^{2\frac{\beta-2}{\beta-1}}\right)$ and $C = 1 + o(1)$. For $\beta = 3$ there is a sharp threshold at $4 \cdot \frac{n}{\ln n}$, since $p_1^2 = \Theta\left(n^{-1}\right) = o\left(\frac{\ln n}{n}\right)$. Also, there is a sharp threshold at $\frac{(\beta-3)\cdot(\beta-1)}{(\beta-2)^2} \cdot n$ for $\beta > 3$, since $p_1^2 = \Theta\left(n^{-2\frac{\beta-2}{\beta-1}}\right) = o(n)$. We already observed the behavior for the latter case experimentally in previous works [21, 20]. Thus, we can say that for power-law random 2-SAT with a fixed power-law exponent $\beta \geqslant 3$ an equivalent of the satisfiability threshold conjecture holds, since the sharp thresholds converge to a function with the same leading constant factor.

6.3 Geometric Distribution

Ansótegui et al. [5] also considered an ensemble of geometric distributions with

$$p_i = \frac{b \cdot (1 - b^{-1/n})}{b - 1} \cdot b^{-(i-1)/n}$$

for $i = 1, \ldots, n$ and for some constant $b > 1$. Again, it already holds that $p_1 \geqslant p_2 \geqslant \ldots \geqslant p_n$. It holds that

$$p_1^2 = \frac{b^2 \cdot (1 - b^{-1/n})^2}{(b-1)^2}$$

and

$$\sum_{i=1}^{n} p_i^2 = \frac{b+1}{b-1} \cdot \frac{1 - b^{-1/n}}{1 + b^{-1/n}}.$$

One can show that $p_1^2 = o(\sum_{i=1}^{n} p_i^2)$. Theorem 1.1 now tells us that there is a sharp threshold at $\frac{b-1}{b+1} \cdot \frac{1+b^{-1/n}}{1-b^{-1/n}}$. This function grows as fast as $\frac{2\cdot(b-1)}{(b+1)\cdot\ln b} \cdot n$ in the limit. Again, we can say that an equivalent of the satisfiability threshold conjecture holds for geometric random 2-SAT with some fixed $b > 1$, since the sharp thresholds $m^*(n)$ converge to $\frac{2\cdot(b-1)}{(b+1)\cdot\ln b} \cdot n$.

7 Discussion and Future Work

We showed a dichotomy of coarse and sharp thresholds for the non-uniform random 2-SAT model depending on the variable probability distribution. In the case of a coarse threshold, the coarseness either stems from two variables being present in too many clauses and forming an unsatisfiable sub-formula of size 4 with constant probability or from a snake with three variables which emerges with constant probability. Furthermore we determined the exact position of the satisfiability threshold in the case of a sharp threshold. Hence, our result generalizes the seminal works by Chvatal and Reed [11] and by Goerdt [24] to arbitrary variable probability distributions. It allows us to prove or disprove an equivalent of the

satisfiability threshold conjecture for non-uniform random 2-SAT. For example for power-law random 2-SAT, an equivalent of the conjecture holds for power law exponents $\beta \geqslant 3$ and the satisfiability threshold is at exactly $\frac{(\beta-3)\cdot(\beta-1)}{(\beta-2)^2} \cdot n$ for $\beta > 3$ and exactly at $4 \cdot \frac{n}{\ln n}$ for $\beta = 3$.

The grand goal of our works is to show similar results for higher values of k, where we already made a first step by showing sharpness for certain variable probability distributions [22]. Another direction we are interested in for $k \geqslant 3$ is proving bounds on the average computational hardness of formulas around the threshold, for example by showing resolution lower bounds like Mull et al. [37].

References

1 Dimitris Achlioptas, Lefteris M. Kirousis, Evangelos Kranakis, and Danny Krizanc. Rigorous results for random (2+p)-SAT. *Theor. Comput. Sci.*, 265(1-2):109–129, 2001.

2 Carlos Ansótegui, Maria Luisa Bonet, Jesús Giráldez-Cru, and Jordi Levy. The Fractal Dimension of SAT Formulas. In *7th Intl. Joint Conf. Automated Reasoning (IJCAR)*, pages 107–121, 2014.

3 Carlos Ansótegui, Maria Luisa Bonet, Jesús Giráldez-Cru, and Jordi Levy. On the Classification of Industrial SAT Families. In *18th Intl. Conf. of the Catalan Association for Artificial Intelligence (CCIA)*, pages 163–172, 2015.

4 Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. On the Structure of Industrial SAT Instances. In *15th Intl. Conf. Principles and Practice of Constraint Programming (CP)*, pages 127–141, 2009.

5 Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. Towards Industrial-Like Random SAT Instances. In *21st Intl. Joint Conf. Artificial Intelligence (IJCAI)*, pages 387–392, 2009.

6 Carlos Ansótegui, Jesús Giráldez-Cru, and Jordi Levy. The Community Structure of SAT Formulas. In *15th Intl. Conf. Theory and Applications of Satisfiability Testing (SAT)*, pages 410–423, 2012.

7 Victor Bapst and Amin Coja-Oghlan. The Condensation Phase Transition in the Regular k-SAT Model. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016*, pages 22:1–22:18, 2016.

8 Yacine Boufkhad, Olivier Dubois, Yannet Interian, and Bart Selman. Regular Random k-SAT: Properties of Balanced Formulas. *J. Autom. Reasoning*, 35(1-3):181–200, 2005.

9 Milan Bradonjic and Will Perkins. On Sharp Thresholds in Random Geometric Graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014*, pages 500–514, 2014.

10 Karl Bringmann. Why Walking the Dog Takes Time: Frechet Distance Has No Strongly Subquadratic Algorithms Unless SETH Fails. In *55th Symp. Foundations of Computer Science (FOCS)*, pages 661–670, 2014.

11 Václav Chvatal and Bruce Reed. Mick gets some (the odds are on his side). In *33rd Symp. Foundations of Computer Science (FOCS)*, pages 620–627, 1992.

12 Amin Coja-Oghlan. The Asymptotic k-SAT Threshold. In *46th Symp. Theory of Computing (STOC)*, pages 804–813, 2014.

13 Amin Coja-Oghlan and Konstantinos Panagiotou. The asymptotic k-SAT threshold. *Advances in Mathematics*, 288:985–1068, 2016.

14 Amin Coja-Oghlan and Nick Wormald. The Number of Satisfying Assignments of Random Regular k-SAT Formulas. *Combinatorics, Probability and Computing*, 27(4):496–530, 2018.

15 Stephen A. Cook. The Complexity of Theorem-Proving Procedures. In *3rd Symp. Theory of Computing (STOC)*, pages 151–158, 1971.

16 Colin Cooper, Alan Frieze, and Gregory Sorkin. Random 2SAT with Prescribed Literal Degrees. *Algorithmica*, 48:249–265, July 2007.

17 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving Connectivity Problems Parameterized by Treewidth in Single Exponential Time. In *52nd Symp. Foundations of Computer Science (FOCS)*, pages 150–159, 2011.

18 Josep Díaz, Lefteris M. Kirousis, Dieter Mitsche, and Xavier Pérez-Giménez. On the satisfiability threshold of formulas with three literals per clause. *Theoretical Computer Science*, 410(30-32):2920–2934, 2009.

19 Jian Ding, Allan Sly, and Nike Sun. Proof of the Satisfiability Conjecture for Large K. In *47th Symp. Theory of Computing (STOC)*, pages 59–68, 2015.

20 Tobias Friedrich, Anton Krohmer, Ralf Rothenberger, Thomas Sauerwald, and Andrew M. Sutton. Bounds on the Satisfiability Threshold for Power Law Distributed Random SAT. In *25th European Symposium on Algorithms (ESA)*, pages 37:1–37:15, 2017.

21 Tobias Friedrich, Anton Krohmer, Ralf Rothenberger, and Andrew M. Sutton. Phase Transitions for Scale-Free SAT Formulas. In *31st Conf. Artificial Intelligence (AAAI)*, pages 3893–3899, 2017.

22 Tobias Friedrich and Ralf Rothenberger. Sharpness of the Satisfiability Threshold for Non-uniform Random k-SAT. In *21st Intl. Conf. Theory and Applications of Satisfiability Testing (SAT)*, pages 273–291, 2018.

23 Jesús Giráldez-Cru and Jordi Levy. A Modularity-Based Random SAT Instances Generator. In *24 thIntl. Joint Conf. Artificial Intelligence (IJCAI)*, pages 1952–1958, 2015.

24 Andreas Goerdt. A Threshold for Unsatisfiability. *J. Comput. Syst. Sci.*, 53(3):469–486, 1996.

25 Mohammad Taghi Hajiaghayi and Gregory B. Sorkin. The Satisfiability Threshold of Random 3-SAT is at Least 3.52. Technical Report RC22942, IBM, October 2003.

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

27 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? In *39th Symp. Foundations of Computer Science (FOCS)*, pages 653–663, 1998.

28 Alexis C. Kaporis, Lefteris M. Kirousis, and Efthimios G. Lalas. The probabilistic analysis of a greedy satisfiability algorithm. *Random Struct. Algorithms*, 28(4):444–480, 2006.

29 Richard M. Karp. Reducibility Among Combinatorial Problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York.*, pages 85–103, 1972.

30 Leonid A. Levin. Universal sorting problems. *Problems of Information Transmission*, 9:265–266, 1973.

31 Jordi Levy. Percolation and Phase Transition in SAT. *CoRR*, abs/1708.06805, 2017. `arXiv:1708.06805`.

32 Marc Mézard, Giorgio Parisi, and Riccardo Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297(5582):812–815, 2002.

33 David G. Mitchell, Bart Selman, and Hector J. Levesque. Hard and Easy Distributions of SAT Problems. In *10th Conf. Artificial Intelligence (AAAI)*, pages 459–465, 1992.

34 Rémi Monasson and Riccardo Zecchina. Statistical mechanics of the random K-satisfiability model. *Phys. Rev. E*, 56:1357–1370, August 1997.

35 Rémi Monasson, Riccardo Zecchina, Scott Kirkpatric, Bart Selman, and Lidror Troyansky. Phase transition and search cost in the 2+ p-sat problem. *4th Workshop on Physics and Computation, Boston, MA, 1996.*, 1996.

36 Rémi Monasson, Riccardo Zecchina, Scott Kirkpatrick, Bart Selman, and Lidror Troyansky. 2+p-SAT: Relation of typical-case complexity to the nature of the phase transition. *Random Struct. Algorithms*, 15(3-4):414–435, 1999.

37 Nathan Mull, Daniel J. Fremont, and Sanjit A. Seshia. On the Hardness of SAT with Community Structure. In *19th Intl. Conf. Theory and Applications of Satisfiability Testing (SAT)*, pages 141–159, 2016.

38 Vishwambhar Rathi, Erik Aurell, Lars K. Rasmussen, and Mikael Skoglund. Bounds on Threshold of Regular Random k-SAT. In *13th Intl. Conf. Theory and Applications of Satisfiability Testing (SAT)*, pages 264–277, 2010.

Determinant Equivalence Test over Finite Fields and over $\mathbb{Q}$

Ankit Garg
Microsoft Research India, Bangalore, India
garga@microsoft.com

Nikhil Gupta
Department of Computer Science and Automation, Indian Institute of Science, India
nikhilg@iisc.ac.in

Neeraj Kayal
Microsoft Research India, Bangalore, India
neeraka@microsoft.com

Chandan Saha
Department of Computer Science and Automation, Indian Institute of Science, India
chandan@iisc.ac.in

Abstract

The determinant polynomial $\mathsf{Det}_n(\mathbf{x})$ of degree n is the determinant of a $n \times n$ matrix of formal variables. A polynomial f is equivalent to $\mathsf{Det}_n(\mathbf{x})$ over a field $\mathbb{F}$ if there exists a $A \in GL(n^2, \mathbb{F})$ such that $f = \mathsf{Det}_n(A \cdot \mathbf{x})$. *Determinant equivalence test over* $\mathbb{F}$ is the following algorithmic task: Given black-box access to a $f \in \mathbb{F}[\mathbf{x}]$, check if f is equivalent to $\mathsf{Det}_n(\mathbf{x})$ over $\mathbb{F}$, and if so then output a transformation matrix $A \in GL(n^2, \mathbb{F})$. In (Kayal, STOC 2012), a randomized polynomial time determinant equivalence test was given over $\mathbb{F} = \mathbb{C}$. But, to our knowledge, the complexity of the problem over finite fields and over $\mathbb{Q}$ was not well understood.

In this work, we give a randomized $\mathrm{poly}(n, \log |\mathbb{F}|)$ time determinant equivalence test over finite fields $\mathbb{F}$ (under mild restrictions on the characteristic and size of $\mathbb{F}$). Over $\mathbb{Q}$, we give an efficient randomized reduction from factoring square-free integers to determinant equivalence test for quadratic forms (i.e. the $n = 2$ case), assuming GRH. This shows that designing a polynomial-time determinant equivalence test over $\mathbb{Q}$ is a challenging task. Nevertheless, we show that determinant equivalence test over $\mathbb{Q}$ is decidable: For bounded n, there is a randomized polynomial-time determinant equivalence test over $\mathbb{Q}$ with access to an oracle for integer factoring. Moreover, for *any* n, there is a randomized polynomial-time algorithm that takes input black-box access to a $f \in \mathbb{Q}[\mathbf{x}]$ and if f is equivalent to Det_n over $\mathbb{Q}$ then it returns a $A \in GL(n^2, Ł)$ such that $f = \mathsf{Det}_n(A \cdot \mathbf{x})$, where Ł is an extension field of $\mathbb{Q}$ and $[Ł : \mathbb{Q}] \leq n$.

The above algorithms over finite fields and over $\mathbb{Q}$ are obtained by giving a polynomial-time randomized reduction from determinant equivalence test to another problem, namely the *full matrix algebra isomorphism* problem. We also show a reduction in the converse direction which is efficient if n is bounded. These reductions, which hold over any $\mathbb{F}$ (under mild restrictions on the characteristic and size of $\mathbb{F}$), establish a close connection between the complexity of the two problems. This then leads to our results via applications of known results on the full algebra isomorphism problem over finite fields (Rónyai, STOC 1987 and Rónyai, J. Symb. Comput. 1990) and over $\mathbb{Q}$ (Ivanyos et al., Journal of Algebra 2012 and Babai et al., Mathematics of Computation 1990).

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Determinant equivalence test, full matrix algebra isomorphism, Lie algebra

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.62

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://eccc.weizmann.ac.il/report/2019/042/`.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 62; pp. 62:1–62:15

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

Acknowledgements We would like to thank Youming Qiao for pointing us to the module decomposition algorithm in [4]. NG would like to thank Vineet Nair for discussions on the structure of the Lie algebra of Det. We thank him for sharing his proof of Theorem 10. We also thank anonymous reviewers for their comments.

1 Introduction

Two m-variate polynomials $f(\mathbf{x})$ and $g(\mathbf{x})$ with coefficients from a field $\mathbb{F}$ are said to be *equivalent over* $\mathbb{F}$ if there exists a $A \in GL(m, \mathbb{F})$ such that $f = g(A \cdot \mathbf{x})$. The algorithmic task of determining if f is equivalent to g, and if so then finding a linear transformation A such that $f = g(A \cdot \mathbf{x})$, is known as the polynomial *equivalence test* problem. It is a natural problem arising in algebraic complexity theory, becoming more important with the advent of Geometric Complexity Theory (GCT) [22] – which proposes the uses of deep tools and insights from group theory, representation theory and algebraic geometry towards the study of the VP vs VNP question.

A naïve approach for equivalence test is to reduce it to solving a system of polynomial equations over $\mathbb{F}$. But, unfortunately, polynomial solvability problem is NP-hard over $\mathbb{C}$ and finite fields and not known to be decidable over $\mathbb{Q}$. Nevertheless, it does appear that the complexity of equivalence test is much lower than the complexity of solving polynomial systems. It is known that over finite fields, the polynomial equivalence problem can not be NP-hard unless PH collapses (when the polynomials are given as lists of coefficients) [29, 28].

Can we hope to solve equivalence test over $\mathbb{C}$ and over finite fields [1] in (randomized) polynomial time? Finding such an algorithm is indeed quite demanding as it was shown in [1, 2] that the graph isomorphism problem reduces in polynomial time to equivalence test for cubic forms (i.e. homogeneous degree three polynomials) over *any* field. Over $\mathbb{Q}$, it is not even known if cubic form equivalence is decidable. On the other hand, we have a fairly good understanding of the complexity of quadratic form equivalence test: Over $\mathbb{C}$ and finite fields, equivalence of two quadratic forms can be tested in polynomial time due to well-known results on classification of quadratic forms. Quadratic form equivalence over $\mathbb{Q}$ can be done in polynomial-time with access to an oracle for integer factoring (IntFact). Moreover, IntFact reduces in randomized polynomial time to quadratic form equivalence over $\mathbb{Q}$ (see [31]). Given this state of affairs, designing efficient equivalence tests for even bounded degree polynomials seems like a difficult proposition. Indeed, there is a cryptographic authentication scheme based on the presumed average-case hardness of equivalence test for constant degree polynomials (see [23]).

The work in [15] initiated the study of a kind of equivalence test in which one polynomial f is given as input and the other polynomial g belongs to a well-defined polynomial family. Some of the polynomial families that are well-studied in algebraic complexity theory, particularly in the context of arithmetic circuit lower bounds, are those defined by the power symmetric polynomial, the elementary symmetric polynomial, the permanent, the determinant and the iterated matrix multiplication polynomial. In [15], randomized polynomial time equivalence tests over $\mathbb{C}$ were given for the power symmetric polynomial and the elementary symmetric polynomial families. These equivalence tests, which also hold over finite fields and $\mathbb{Q}$, work

[1] Typically, a computation model over $\mathbb{C}$ assumes that basic arithmetic operations with complex numbers and root finding of univariate polynomials over $\mathbb{C}$ can be done efficiently. Also, we will work with finite fields that have sufficiently large size and characteristic.

even if f is given as a black-box[2]. Henceforth, let us assume that the input polynomial f is given as a black-box. Subsequently, in [16], randomized polynomial time equivalence tests over $\mathbb{C}$ were given for the permanent and the determinant polynomial families. The test for the permanent holds over finite fields and $\mathbb{Q}$, but the same is *not true* for the determinant equivalence test in [16]. In [18], an equivalence test for the iterated matrix multiplication (IMM) was given which holds over $\mathbb{C}$, finite fields and $\mathbb{Q}$ (see also [13]). The iterated matrix multiplication and the determinant families have very similar circuit complexity: Both the families are complete under p-projections for class of algebraic branching programs (ABP) (see [20, 21]). But, it was unclear if determinant admits an efficient equivalence test over finite fields and $\mathbb{Q}$, just like the iterated matrix multiplication polynomial. In this paper, we fill in this gap in our understanding.

It is worth noting that determinant equivalence test is interesting in the context of the permanent versus determinant problem [30], which conjectures that the permanent is not an affine projection of a polynomial-size determinant. Geometric Complexity Theory [22], an approach to resolving this conjecture, suggests (among other things) to look for an algorithm to determine if the (padded) permanent is in the orbit closure of a polynomial-size determinant. In this language, determinant equivalence testing is the related problem of checking if a given polynomial is in the orbit of the determinant polynomial.

1.1 Our results

Let $n \in \mathbb{N}^{\times}$, $X = (x_{ij})_{i,j\in[n]}$ be a $n \times n$ matrix of formal variables, and $\mathbf{x} = (x_{11}\ x_{12}\ \ldots\ x_{n\,n-1}\ x_{nn})^T$ a column vector consisting of the variables in X arranged in a row-major fashion. The polynomial $\mathsf{Det}_n(\mathbf{x}) := \det(X)$; we will drop the subscript n whenever it is clear from the context. Hereafter, we will use the acronym DET for Determinant Equivalence Test.

▶ **Theorem 1** (DET over finite fields). *Let $\mathbb{F}$ be a finite field such that $|\mathbb{F}| \geq 10n^4$ and $char(\mathbb{F}) \nmid n(n-1)$. There is a randomized $\mathsf{poly}(n, \log|\mathbb{F}|)$ time algorithm that takes input black-box access to a $f \in \mathbb{F}[\mathbf{x}]$ of degree n and does the following with high probability: If f is equivalent to $\mathsf{Det}(\mathbf{x})$ over $\mathbb{F}$ then it outputs a $A \in GL(n^2, \mathbb{F})$ such that $f = \mathsf{Det}(A \cdot \mathbf{x})$; otherwise, it outputs "Fail".*

In [17], a DET over a finite field $\mathbb{F}_q$ was given that is similar to the equivalence test for the permanent in [16], but the test outputs a $A \in GL(n^2, \mathbb{F}_{q^n})$. Whereas, our algorithm (which is different and relatively more involved) outputs a $A \in GL(n^2, \mathbb{F}_q)$. One consequence of this is that the average-case ABP reconstruction algorithm in [17] holds over the base field $\mathbb{F}_q$.

▶ **Theorem 2** (DET over $\mathbb{Q}$).

(a) *There is a randomized algorithm, with oracle access to $\mathsf{IntFact}$, that takes input black-box access to a $f \in \mathbb{Q}[\mathbf{x}]$ of degree n and does the following with high probability: If f is equivalent to $\mathsf{Det}(\mathbf{x})$ over $\mathbb{Q}$ then it outputs a $A \in GL(n^2, \mathbb{Q})$ such that $f = \mathsf{Det}(A \cdot \mathbf{x})$; otherwise, it outputs "Fail". If n is bounded then the algorithm runs in time polynomial in the bit length of the coefficients of f.*

[2] An algorithm with black-box access to a m-variate polynomial f is only allowed to query the black-box for evaluations of f at points in $\mathbb{F}^m$.

(b) *There is a randomized algorithm that takes input black-box access to a $f \in \mathbb{Q}[\mathbf{x}]$ of degree n and does the following with high probability: If f is equivalent to $\mathsf{Det}(\mathbf{x})$ over $\mathbb{Q}$ then it outputs a $A \in GL(n^2, Ł)$ such that $f = \mathsf{Det}(A \cdot \mathbf{x})$, where Ł is an extension field of $\mathbb{Q}$ and $[Ł : \mathbb{Q}] \leq n$. The algorithm runs in time polynomial in n and the bit length of the coefficients of f.*

To our knowledge, it was not known if DET over $\mathbb{Q}$ is decidable prior to this work. It is natural to wonder if we can get rid of the IntFact oracle from part (a) of the above theorem. In this regard, we show the following.

▶ **Theorem 3** (IntFact reduces to DET for quadratic forms)**.** *Assuming GRH, we give a randomized polynomial-time reduction from factoring square-free integers to finding a $A \in M_2(\mathbb{Q})$ such that a given quadratic form $f \in \mathbb{Q}[\mathbf{x}]$ equals $\mathsf{Det}_2(A \cdot \mathbf{x})$, if f is equivalent to Det_2.*

The complexity of IntFact is the same as that of DET over $\mathbb{Q}$ for quadratic forms (modulo GRH and the use of randomization). Theorem 3 is a reduction from a result in [24].

Theorem 1 and 2 are proved by reducing DET to the *full matrix algebra isomorphism* problem. An $\mathbb{F}$-algebra $\mathcal{A}$ has two binary operations $+$ and $\cdot$ defined on its elements such that $(\mathcal{A}, +)$ is a $\mathbb{F}$-vector space, $(\mathcal{A}, +, \cdot)$ is an associative ring, and for every $a, b \in \mathbb{F}$ and $B, C \in \mathcal{A}$ it holds that $(aB)C = B(aC) = a(BC)$. For example, the set $M_n(\mathbb{F})$ of all $n \times n$ matrices over $\mathbb{F}$ is a $\mathbb{F}$-algebra with respect to the usual matrix addition and multiplication operations; it is called the full matrix algebra. Two $\mathbb{F}$-algebra $\mathcal{A}_1$ and $\mathcal{A}_2$ are isomorphic, denoted by $\mathcal{A}_1 \cong \mathcal{A}_2$, if there is a bijection ϕ from $\mathcal{A}_1$ to $\mathcal{A}_2$ such that for every $a, b \in \mathbb{F}$ and $B, C \in \mathcal{A}_1$ it holds that $\phi(aB + bC) = a\phi(B) + b\phi(C)$ and $\phi(BC) = \phi(B)\phi(C)$. Any finite dimensional $\mathbb{F}$-algebra is isomorphic to a $\mathbb{F}$-algebra $\mathcal{A}' \subseteq M_m(\mathbb{F})$, where $m = \dim_{\mathbb{F}}(\mathcal{A})$. A $\mathbb{F}$-algebra $\mathcal{A} \subseteq M_m(\mathbb{F})$ can be specified by a $\mathbb{F}$-basis $B_1, \ldots, B_r \in M_m(\mathbb{F})$.

▶ **Definition 4.** *The full matrix algebra isomorphism (FMAI) problem over $\mathbb{F}$ is the following: Given a basis of a $\mathbb{F}$-algebra $\mathcal{A} \subseteq M_m(\mathbb{F})$, check if $\mathcal{A} \cong M_n(\mathbb{F})$, where $n^2 = \dim_{\mathbb{F}}(\mathcal{A})$. If $\mathcal{A} \cong M_n(\mathbb{F})$ then output an isomorphism from $\mathcal{A}$ to $M_n(\mathbb{F})$.*

In [24, 25], a $\mathsf{poly}(m, \log|\mathbb{F}|)$ time randomized algorithm was given to solve FMAI over a finite field $\mathbb{F}$. Over $\mathbb{Q}$, the FMAI problem is more difficult. In [14, 6], a randomized algorithm (with access to a IntFact oracle) was given to solve FMAI over $\mathbb{Q}$. The algorithm runs in polynomial-time if $\dim_{\mathbb{Q}}(\mathcal{A})$ is bounded. In [3, 10], randomized polynomial time algorithms were given to compute an isomorphism from $\mathcal{A} \otimes_{\mathbb{Q}} Ł$ to $M_n(Ł)$ for some extension field $Ł \supseteq \mathbb{Q}$ satisfying $[Ł : \mathbb{Q}] \leq n$, if $\mathcal{A} \cong M_n(\mathbb{Q})$ to begin with. We give a randomized polynomial-time reduction from DET to FMAI over any sufficiently large $\mathbb{F}$ in Section 4, thereby proving Theorem 1 and 2. The reduction is obtained by giving an algorithm to decompose the Lie algebra of f into its two simple Lie subalgebras over any sufficiently large $\mathbb{F}$ (see Section 3). The same reduction also gives DET over $\mathbb{R}$ and $\mathbb{C}$ via the FMAI algorithms in [9, 26] (thereby giving alternative algorithms to the one presented in [16]). We also show a reduction from FMAI to DET (in Section 7) which is efficient if the dimension n is bounded.

The above results underscore the close connection between the DET and the FMAI problems. In order to get efficient DET over $\mathbb{Q}$ for even bounded degree polynomials, we *need to* solve FMAI efficiently for $\mathbb{Q}$-algebras of bounded dimensions. Currently, the best known algorithm for FMAI over $\mathbb{Q}$ uses an IntFact oracle [14]. This situation of the determinant is somewhat surprising as it contrasts that of IMM (the close cousin of the determinant) – IMM equivalence test over $\mathbb{Q}$ can be solved efficiently for polynomials of degree greater than four [18].

2 Preliminaries

2.1 Notations

The set of trace zero or traceless matrices in $M_n(\mathbb{F})$ is denoted by $\mathcal{Z}_n(\mathbb{F})$; we will drop $\mathbb{F}$ from $M_n(\mathbb{F})$ and $\mathcal{Z}_n(\mathbb{F})$ when it is clear from the context. Let I_n be the $n \times n$ identity matrix. $\otimes$ denotes tensor product of two matrices. Define,

$$\mathcal{M}_{\text{col}} := I_n \otimes M_n, \quad \mathcal{M}_{\text{row}} := M_n \otimes I_n \qquad \text{and} \qquad \mathcal{L}_{\text{col}} := I_n \otimes \mathcal{Z}_n, \quad \mathcal{L}_{\text{row}} := \mathcal{Z}_n \otimes I_n.$$

Observe $\mathcal{M}_{\text{col}}$, $\mathcal{M}_{\text{row}} \subseteq M_{n^2}$ are $\mathbb{F}$-algebras isomorphic to M_n, and $\mathcal{L}_{\text{col}}$, $\mathcal{L}_{\text{row}}$ are their subspaces , respectively, of dimension n^2-1 each. Henceforth, we set $m = n^2$ and $r = n^2 - 1$.

2.2 Definitions

▶ **Definition 5** (Lie bracket). *For $A, B \in M_n$, the Lie bracket operation $[A, B] := AB - BA$.*

▶ **Definition 6** (Lie algebra of a polynomial). *The Lie algebra $\mathfrak{g}_f$ of a m-variate polynomial $f \in \mathbb{F}[\mathbf{x}]$ is the set of matrices $B = (b_{i,j})_{i,j\in[m]}$ satisfying,*

$$\sum_{i,j\in[m]} b_{i,j} \cdot x_j \cdot \frac{\partial f}{\partial x_i} = 0.$$

It is easy to verify that $[\cdot,\cdot]$ is a $\mathbb{F}$-bilinear map on M_n, and $\mathfrak{g}_f$ is an $\mathbb{F}$-vector space.[3] Let $\mathcal{V}$ be a $\mathbb{F}$-vector space, $\text{End}_{\mathbb{F}}(\mathcal{V}) := \{\varphi : \varphi \text{ is a } \mathbb{F}\text{-linear map from } \mathcal{V} \text{ to } \mathcal{V}\}$ and $\mathcal{T} \subseteq \text{End}_{\mathbb{F}}(\mathcal{V})$.

▶ **Definition 7.** *A subspace $\mathcal{U}$ of $\mathcal{V}$ is called $\mathcal{T}$-invariant if for every $\varphi \in \mathcal{T}$, $\varphi(\mathcal{U}) \subseteq \mathcal{U}$.*

If $\mathcal{T} \subseteq M_{2r}$, the terminology "invariant subspace of $\mathcal{T}$" means $\mathcal{T}$-invariant subspace of $\mathbb{F}^{2r}$.

▶ **Definition 8** (Irreducible invariant subspace). *A $\mathcal{T}$-invariant subspace $\mathcal{U}$ of $\mathcal{V}$ is irreducible if there do not exist proper $\mathcal{T}$-invariant subspaces $\mathcal{U}_1, \mathcal{U}_2$ of $\mathcal{U}$, such that $\mathcal{U} = \mathcal{U}_1 \oplus \mathcal{U}_2$.*

▶ **Definition 9** (Closure of a vector). *Let $\mathbf{w} \in \mathcal{V}$. Then, the closure of $\mathbf{w}$ with respect to $\mathcal{T}$, denoted $\text{closure}_{\mathcal{T}}(\mathbf{w})$, is the smallest $\mathcal{T}$-invariant subspace of $\mathcal{V}$ containing $\mathbf{w}$.*

2.3 Some basic results

▶ **Observation 2.1.** *For $i, j \in [n], i \neq j$, let $E_{ij} \in M_n$ be such that the (i,j)-th entry is 1 and other entries are 0, and for $\ell \in [2, n]$, let $E_\ell \in M_n$ be a diagonal matrix with the $(1,1)$-th and (ℓ,ℓ)-th entries as 1 and -1 respectively and other entries as 0. Then,*

1. *$\{I_n \otimes E_{ij},\ I_n \otimes E_\ell \ :\ i, j \in [n], i \neq j,\ \text{and } \ell \in [2, n]\}$ is a basis of $\mathcal{L}_{col}$. Denote the elements of this standard basis as $S_1, \ldots, S_r$.*
2. *$\{E_{ij} \otimes I_n,\ E_\ell \otimes I_n \ :\ i, j \in [n], i \neq j,\ \text{and } \ell \in [2, n]\}$ is a basis of $\mathcal{L}_{row}$. Denote the elements of this standard basis as $S_{r+1}, \ldots, S_{2r}$.*

▶ **Observation 2.2.** *For every $F \in \mathcal{M}_{row}$ and $L \in \mathcal{M}_{col}$, $[F, L] = FL - LF = 0$.*

▶ **Observation 2.3.** *For every $L_1, L_2 \in \mathcal{L}_{col}$ (or $\mathcal{L}_{row}$), $[L_1, L_2] \in \mathcal{L}_{col}$ (respectively. $\mathcal{L}_{row}$).*

[3] Over $\mathbb{C}$, $\mathfrak{g}_f$ also turns out to be a Lie algebra i.e. closed under the Lie bracket operation. However, over finite fields, it is not clear if it is closed under the bracket operation. We still stick with the terminology Lie algebra of a polynomial since in many cases, it does turn out to be closed under the bracket operation.

A proof of the following standard fact is given in Section A.1 of the Appendix of the full version [11].

▶ **Fact 1.** *Let $B \in M_n$. Then, the dimension of the space of matrices in M_n that commute with B is at least n, and the dimension of the space of matrices in $\mathcal{Z}_n$ that commute with B is at least $n-1$.*

We would also need the following facts (see [16, 18] for their proofs).

▶ **Fact 2.** *If $g \in \mathbb{F}[\mathbf{x}]$ and $f(\mathbf{x}) = g(A \cdot \mathbf{x})$ for some $A \in GL(m, \mathbb{F})$ then $\mathfrak{g}_f = A^{-1} \cdot \mathfrak{g}_g \cdot A$.*

▶ **Fact 3.** *Suppose we have black box access to a m-variate polynomial $f \in \mathbb{F}[\mathbf{x}]$, where $|\mathbb{F}| \geq 2n^3$. Then, a basis of $\mathfrak{g}_f$ can be computed in randomized polynomial time.*

▶ **Fact 4.** *Given a basis $\{T_1, \ldots, T_s\}$ of $\mathcal{T} \subseteq M_{2r}$ and a $\mathbf{w} \in \mathbb{F}^{2r}$, a basis of $\text{closure}_{\mathcal{T}}(\mathbf{w})$ can be computed in time polynomial in r and the bit length of the entries in $\mathbf{w}$ and $T_1, \ldots, T_s$.*

The following theorem on the Lie algebra of Det is well-known over $\mathbb{C}$. We give a proof over any field (with a mild condition on the characteristic) in Section A.2 of the Appendix of [11].

▶ **Theorem 10** (Lie algebra of Det)**.** *Let $n \geq 2$ and $\mathbb{F}$ be a field such that $char(\mathbb{F}) \nmid n$. Then, the Lie algebra of Det_n equals the direct sum of the spaces $\mathcal{L}_{row}$ and $\mathcal{L}_{col}$, i.e., $\mathfrak{g}_{\mathsf{Det}} = \mathcal{L}_{row} \oplus \mathcal{L}_{col}$.*

The theorem implies that $\{S_1, \ldots, S_{2r}\}$, in Observation 2.1, forms a basis of $\mathfrak{g}_{\mathsf{Det}}$. The rows and columns of every element in $\mathfrak{g}_{\mathsf{Det}}$ are indexed by the $\mathbf{x}$ variables, in order. Let $f = \mathsf{Det}(A \cdot \mathbf{x})$ for some $A \in GL(m, \mathbb{F})$. Then, Theorem 10 and Fact 2 imply that $\mathfrak{g}_f = A^{-1} \cdot \mathcal{L}_{\text{row}} \cdot A \oplus A^{-1} \cdot \mathcal{L}_{\text{col}} \cdot A$. We denote $A^{-1} \cdot \mathcal{L}_{\text{row}} \cdot A$ and $A^{-1} \cdot \mathcal{L}_{\text{col}} \cdot A$ by $\mathcal{F}_{\text{row}}$ and $\mathcal{F}_{\text{col}}$ respectively, and refer to $\mathcal{F}_{\text{row}}$ and $\mathcal{F}_{\text{col}}$ (similarly, $\mathcal{L}_{\text{row}}$ and $\mathcal{L}_{\text{col}}$) as the Lie subalgebras of $\mathfrak{g}_f$ (respectively, $\mathfrak{g}_{\mathsf{Det}}$) [4]. From Theorem 10, Observation 2.2 and 2.3, we get the following.

▶ **Observation 2.4.** *For every $E, F \in \mathfrak{g}_f$, $[E, F] \in \mathfrak{g}_f$.*

▶ **Observation 2.5.** *Let $\mathcal{A} \subseteq M_m$ be the $\mathbb{F}$-algebra generated by a basis of $\mathcal{F}_{col}$. Then,*

$$\mathcal{A} = A^{-1} \cdot (I_n \otimes M_n) \cdot A.$$

This can be proved easily. Finally, we record a special case of the Skolem-Noether theorem which will be used in Section 4. Its general statement can be found in [19] (page 173).

▶ **Theorem 11** (Skolem-Noether)**.** *Let $n, s \in \mathbb{N}^{\times}$ such that $n \mid s$, and $\mathcal{A} \subseteq M_s$ be a $\mathbb{F}$-algebra (containing I_s) that is isomorphic to M_n via $\phi : M_n \to \mathcal{A}$. Then there exists a $K \in GL(s, \mathbb{F})$ s.t.,*

$$\phi(C) = K^{-1} \cdot (I_{s/n} \otimes C) \cdot K, \quad \text{for every } C \in M_n.$$

3 Decomposition of $\mathfrak{g}_f$ into its Lie subalgebras

We show how to compute bases of $\mathcal{F}_{\text{row}}$ and $\mathcal{F}_{\text{col}}$ from black box access to $f = \mathsf{Det}(A \cdot \mathbf{x})$.

▶ **Theorem 12** (Decomposition of $\mathfrak{g}_f$)**.** *Let $n \geq 2$, $|\mathbb{F}| \geq 10n^4$ and $\text{char}(\mathbb{F}) \nmid n(n-1)$. There is a randomized algorithm, which takes input black box access to f and outputs bases of $\mathcal{F}_{row}$ and $\mathcal{F}_{col}$ with high probability. The running time is $\text{poly}(n, \gamma)$, where γ is the bit length of coefficients of f.*

We first present the proof idea, and then the algorithm and its proof of correctness. The missing proofs are given in Sections B,C and D of the Appendix of [11].

[4] Observation 2.3 implies that $\mathcal{F}_{\text{row}}$ and $\mathcal{F}_{\text{col}}$ are closed under the Lie bracket operation and hence they are matrix Lie algebras.

3.1 Proof of Theorem 12: The idea

The algorithm relies on finding the irreducible invariant subspaces of a set of $\mathbb{F}$-linear maps on $\mathfrak{g}_f$. These linear maps (a.k.a adjoint homomorphisms of $\mathfrak{g}_f$) are defined for every $F \in \mathfrak{g}_f$,

$$\rho_F : \mathfrak{g}_f \to \mathfrak{g}_f \ ; \ E \mapsto [E, F].$$

It is easy to see that ρ_F is linear. Let $\{B_1, \ldots, B_{2r}\}$ be a basis of $\mathfrak{g}_f$ which can be computed in randomized polynomial time (by Fact 3). As ρ_F is $\mathbb{F}$-linear, we can associate a matrix $P_F \in M_{2r}$ with ρ_F, after fixing an ordering of the basis $(B_1, \ldots, B_{2r})$. Let $\mathcal{P} := \{P_F : F \in \mathfrak{g}_f\}$.

▷ Claim 13. $\mathfrak{g}_f$ and $\mathcal{P}$ are isomorphic as vector spaces via the map $F \mapsto P_F$ for every $F \in \mathfrak{g}_f$.

Its proof is given in Section B.1 of the Appendix of [11]. This implies the following.

▶ **Observation 3.1.** *The matrices $\{P_{B_1}, \ldots, P_{B_{2r}}\}$ is a basis of $\mathcal{P}$, which can be efficiently computed from $\{B_1, \ldots, B_{2r}\}$ (by considering the elements $[B_i, B_j]$, for $i, j \in [2r]$).*

We intend to study the irreducible invariant subspaces of $\mathcal{P}$ in order to compute bases of $\mathcal{F}_{\text{row}}$ and $\mathcal{F}_{\text{col}}$. The following Claim 14 would be useful in this regard.

It follows from Fact 2 that $J_i := A \cdot B_i \cdot A^{-1}$, for $i \in [2r]$, is a basis of $\mathfrak{g}_{\mathsf{Det}}$. Like ρ_F, we can associate a $\mathbb{F}$-linear map (i.e. adjoint homomorphism) χ_L with every $L \in \mathfrak{g}_{\mathsf{Det}}$ as follows:

$$\chi_L : \mathfrak{g}_{\mathsf{Det}} \to \mathfrak{g}_{\mathsf{Det}} \ ; \ K \mapsto [K, L].$$

Let $Q_L \in M_{2r}$ be the matrix corresponding to the linear map χ_L, with respect to the (ordered) basis $(J_1, \ldots, J_{2r})$. The following claim implies that $\mathcal{P}$ does not depend on the transformation matrix A. Thus, it is sufficient to focus on $\mathfrak{g}_{\mathsf{Det}}$ to study the invariant subspaces of $\mathcal{P}$. The proof of the claim is given in Section B.2 of the Appendix of [11].

▷ Claim 14. For every $i \in [2r]$, $Q_{J_i} = P_{B_i}$ and so the space $\mathcal{P} = \{Q_L : L \in \mathfrak{g}_{\mathsf{Det}}\}$.

Like Claim 13, $\mathfrak{g}_{\mathsf{Det}}$ and $\mathcal{P}$ are isomorphic as vector spaces via the map $L \mapsto Q_L$, for $L \in \mathfrak{g}_{\mathsf{Det}}$. The algorithm computes two invariant subspaces $\mathcal{V}_1$ and $\mathcal{V}_2$ of $\mathcal{P}$ that are defined as follows

$$\mathcal{V}_1 = \left\{ \mathbf{v} = (a_1, \ldots, a_{2r})^T \in \mathbb{F}^{2r} : \sum_{i \in [2r]} a_i \cdot J_i \in \mathcal{L}_{\text{col}} \right\},$$

$$\mathcal{V}_2 = \left\{ \mathbf{v} = (b_1, \ldots, b_{2r})^T \in \mathbb{F}^{2r} : \sum_{i \in [2r]} b_i \cdot J_i \in \mathcal{L}_{\text{row}} \right\}. \tag{1}$$

Clearly, $\dim(\mathcal{V}_1) = \dim(\mathcal{V}_2) = r$. As $B_i = A^{-1} \cdot J_i \cdot A$, for $i \in [2r]$, we get

$$\mathcal{V}_1 = \left\{ \mathbf{v} = (a_1, \ldots, a_{2r})^T \in \mathbb{F}^{2r} : \sum_{i \in [2r]} a_i \cdot B_i \in \mathcal{F}_{\text{col}} \right\},$$

$$\mathcal{V}_2 = \left\{ \mathbf{v} = (b, \ldots, b_{2r})^T \in \mathbb{F}^{2r} : \sum_{i \in [2r]} b_i \cdot B_i \in \mathcal{F}_{\text{row}} \right\}. \tag{2}$$

From bases of $\mathcal{V}_1$ and $\mathcal{V}_2$, and $(B_1, \ldots, B_{2r})$, we get bases of $\mathcal{F}_{\text{col}}$ and $\mathcal{F}_{\text{row}}$ readily. The aspects of the space $\mathcal{P}$ that help in computing $\mathcal{V}_1$ and $\mathcal{V}_2$ are the facts that these are the only two irreducible invariant subspaces of $\mathcal{P}$ and bases of these can be computed from a random element of $\mathcal{P}$. These facts are proved in the proof of correctness of Algorithm 1.

3.2 The decomposition algorithm

Algorithm 1 Computation of bases of $\mathcal{F}_{row}$ and $\mathcal{F}_{col}$.

Input: Black box access to f.
Output: Bases of spaces $\mathcal{V}_1$ and $\mathcal{V}_2$ (as in Equation (2)).

1: Compute a basis $B_1, \ldots, B_{2r}$ of $\mathfrak{g}_f$ (see Fact 3), and form the basis $P_{B_1}, \ldots, P_{B_{2r}}$ of $\mathcal{P}$.
2: Pick a random element $Q = r_1 P_{B_1} + \cdots + r_{2r} P_{B_{2r}}$ from $\mathcal{P}$, where every r_i is chosen uniformly and independently at random from a fixed subset of $\mathbb{F}$ of size $10n^4$.
3: Compute the characteristic polynomial $h(z)$ of Q.
4: Factor $h(z)$ into irreducible factors over $\mathbb{F}$. Let $h(z) = z^{2(n-1)} \cdot h_1(z) \cdots h_k(z)$, where $z, h_1, \ldots, h_k$ are mutually coprime and irreducible. If this is not the split, output "Fail".
5: For every $i \in [k]$, compute a basis of the null space $\mathcal{N}_i$ of $h_i(Q)$, pick a vector $\mathbf{v}$ from the basis of $\mathcal{N}_i$ and compute a basis of $\mathcal{C}_i := \text{closure}_{\mathcal{P}}(\mathbf{v})$ (using Fact 4).
6: Remove repetitive spaces from the set $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$. After this, if we are *not* left with exactly two spaces $\mathcal{U}_1$ and $\mathcal{U}_2$ then output "Fail". Else, output bases of $\mathcal{U}_1$ and $\mathcal{U}_2$.

3.3 Analysis of the algorithm

Let us view the space $\mathcal{P}$ through the lens of a convenient basis of $\mathfrak{g}_{\mathsf{Det}}$, namely the standard basis $\{S_1, \ldots, S_{2r}\}$ (given in Observation 2.1). For $K \in \mathfrak{g}_{\mathsf{Det}}$, let $\mathbf{w}_K, \mathbf{v}_K \in \mathbb{F}^{2r}$ be the coordinate vectors of K with respect to the ordered bases $(S_1, \ldots, S_{2r})$ and $(J_1, \ldots, J_{2r})$ respectively. There is a basis change matrix $H \in \mathrm{GL}(2r, \mathbb{F})$, such that for every $K \in \mathfrak{g}_{\mathsf{Det}}$,

$$\mathbf{v}_K = H \cdot \mathbf{w}_K. \tag{3}$$

Recall Q_L from Claim 14. Let $R_L := H^{-1} \cdot Q_L \cdot H$, for every $L \in \mathfrak{g}_{\mathsf{Det}}$, and

$$\mathcal{R} := \{R_L \ : \ L \in \mathfrak{g}_{\mathsf{Det}}\} = H^{-1} \cdot \mathcal{P} \cdot H. \tag{4}$$

Observe that $\{R_{S_1}, \ldots, R_{S_{2r}}\}$ is a basis of $\mathcal{R}$. Also, for every $L, K \in \mathfrak{g}_{\mathsf{Det}}$.

$$R_L \cdot \mathbf{w}_K = \mathbf{w}_{[K,L]}, \tag{5}$$

▶ **Observation 3.2.** *Every $R \in \mathcal{R} \subseteq M_{2r}$ is a block diagonal matrix having two blocks of size $r \times r$ each, i.e, the non-zero entries of R are confined to the entries $\{(S_i, S_j) \ : \ i, j \in [r]\}$ and $\{(S_i, S_j) \ : \ i, j \in [r+1, 2r]\}$.*

The proof of Observation 3.2 is given in Section C.1 of the Appendix of [11]. We refer to the two blocks of R as $R^{(1)}$ and $R^{(2)}$, corresponding to $\{S_1, \ldots, S_r\}$ and $\{S_{r+1}, \ldots, S_{2r}\}$, respectively. Observation 3.3 follows directly from definition of $\mathcal{R}$.

▶ **Observation 3.3.** *$\mathcal{W}$ is an invariant subspace of $\mathcal{R}$ iff $H \cdot \mathcal{W}$ is an invariant subspace of $\mathcal{P}$.*

It allows us to switch from $\mathcal{P}$ to $\mathcal{R}$ while studying the invariant subspaces of $\mathcal{P}$. The following lemmas on the invariant subspaces of $\mathcal{R}$ are crucial in arguing the correctness of Algorithm 1. Their proofs are given in Sections C.2 and C.3 of the Appendix of [11].

▶ **Lemma 15** (Irreducible invariant subspaces). *Let $\mathbf{w}_K \in \mathbb{F}^{2r}$ for a nonzero K in $\mathcal{L}_{col}$ or in $\mathcal{L}_{row}$.*

$$\begin{aligned} \textit{Then,}\ \ \text{closure}_{\mathcal{R}}(\mathbf{w}_K) &= \{\mathbf{w}_L \ : \ L \in \mathcal{L}_{col}\} =: \mathcal{W}_1, \quad \text{if } K \in \mathcal{L}_{\text{col}}, \\ \text{closure}_{\mathcal{R}}(\mathbf{w}_K) &= \{\mathbf{w}_L \ : \ L \in \mathcal{L}_{row}\} =: \mathcal{W}_2, \quad \text{if } K \in \mathcal{L}_{\text{row}}. \end{aligned}$$

Moreover, $\mathcal{W}_1$ and $\mathcal{W}_2$ are the only two irreducible invariant subspaces of $\mathcal{R}$, and $\mathbb{F}^{2r} = \mathcal{W}_1 \oplus \mathcal{W}_2$.

▶ **Lemma 16** (Characteristic polynomial). *Let $R = \sum_{i\in[2r]} \ell_i(r_1,\ldots,r_{2r})\cdot R_{S_i}$, where $\ell_1,\ldots,\ell_{2r}$ are $\mathbb{F}$-linearly independent linear forms and $r_1,\ldots,r_{2r}$ are picked uniformly and independently at random from a fixed subset of $\mathbb{F}$ of size $10n^4$. Then, with high probability, the characteristic polynomial $h_R(z)$ of R factors as $z^{2(n-1)}\cdot h_1(z)\cdots h_k(z)$, where $z, h_1(z),\ldots,h_k(z)$ are mutually coprime irreducible polynomials over $\mathbb{F}$.*

3.3.1 Proof of correctness of Algorithm 1

In Step 2, we choose a random Q from $\mathcal{P}$. By Equation (4), there is a $R\in\mathcal{R}$, such that,

$$R = H^{-1}\cdot Q\cdot H \;=\; r_1R_{J_1}+\cdots+r_{2r}R_{J_{2r}} \;=\; \ell_1(r_1,\ldots,r_{2r})\cdot R_{S_1}+\cdots+\ell_{2r}(r_1,\ldots,r_{2r})\cdot R_{S_{2r}},$$

where $\ell_1,\ldots,\ell_{2r}$ are $\mathbb{F}$-linearly independent linear forms in $r_1,\ldots,r_{2r}$. By Lemma 16, Step 4 holds with high probability. From Observation 3.2, R is a block diagonal matrix with blocks $R^{(1)}$ and $R^{(2)}$. Let $h(z) = g_1(z)\cdot g_2(z)$, where $g_1(z)$ and $g_2(z)$ are the characteristic polynomials of $R^{(1)}$ and $R^{(2)}$, respectively. There are a couple of factors of h, say h_1 and h_2, that divide g_1 and g_2, respectively. In Step 5, we compute the null spaces $\mathcal{N}_1$ and $\mathcal{N}_2$ of $h_1(Q)$ and $h_2(Q)$ respectively. As $h_1(R) = H^{-1}\cdot h_1(Q)\cdot H$ and $h_2(R) = H^{-1}\cdot h_2(Q)\cdot H$, the null spaces of $h_1(R)$ and $h_2(R)$, denoted by $\mathcal{O}_1$ and $\mathcal{O}_2$ respectively, satisfy $\mathcal{O}_1 = H^{-1}\cdot\mathcal{N}_1$ and $\mathcal{O}_2 = H^{-1}\cdot\mathcal{N}_2$ (due to Equation (3)).

▷ Claim 17. If $\mathbf{w}_K\in\mathcal{O}_1$ (similarly, $\mathbf{w}_K\in\mathcal{O}_2$) then $K\in\mathcal{L}_{\text{col}}$ (respectively, $K\in\mathcal{L}_{\text{row}}$).

Its proof is given in Section D.1 of the Appendix of [11]. In Step 5, we also pick a vector $\mathbf{v}$ from a null space, say $\mathcal{N}_1$, and compute $\text{closure}_{\mathcal{P}}(\mathbf{v})$. Clearly, $\mathbf{v} = \mathbf{v}_K$ for some $K\in\mathfrak{g}_{\mathsf{Det}}$. So, $\mathbf{v}_K\in\mathcal{N}_1$ if and only if $\mathbf{w}_K = H^{-1}\cdot\mathbf{v}_K\in\mathcal{O}_1$. As $\mathcal{R} = H^{-1}\cdot\mathcal{P}\cdot H$, Observation 3.3 implies

$$\begin{aligned}\text{closure}_{\mathcal{P}}(\mathbf{v}_K) &= H\cdot\text{closure}_{\mathcal{R}}(\mathbf{w}_K)\\ &= H\cdot\mathcal{W}_1 \quad (\text{ by Claim 17 and Lemma 15})\\ &= \mathcal{V}_1 \qquad (\text{ by Equations (1) and (3), as } \mathcal{V}_1 = \{\mathbf{v}_L \;:\; L\in\mathcal{L}_{\text{col}}\}).\end{aligned}$$

Similarly, if we pick a $\mathbf{v}\in\mathcal{N}_2$ then $\text{closure}_{\mathcal{P}}(\mathbf{v}) = \mathcal{V}_2$. Thus, in Step 6, one of $\mathcal{U}_1$ and $\mathcal{U}_2$ is $\mathcal{V}_1$ and the other is $\mathcal{V}_2$. Finally, we can take $\mathcal{U}_1 = \mathcal{V}_1$ and $\mathcal{U}_2 = \mathcal{V}_2$ without loss of generality: Let $P\in M_m$ be the permutation matrix, such that when multiplied to $\mathbf{x}$, P maps x_{ij} to x_{ji}. Clearly, $P^{-1} = P$. The following equation holds because P is a symmetry of Det.

$$\mathsf{Det}(\mathbf{x}) = \mathsf{Det}(P\cdot\mathbf{x}) \quad\text{and hence}\quad f(\mathbf{x}) = \mathsf{Det}(A\cdot\mathbf{x}) = \mathsf{Det}(PA\cdot\mathbf{x}).$$

Observe that $\mathcal{L}_{\text{col}} = P^{-1}\cdot\mathcal{L}_{\text{row}}\cdot P$. Hence,

$$\mathcal{F}_{\text{col}} = A^{-1}P^{-1}\cdot\mathcal{L}_{\text{row}}\cdot PA \quad\text{and}\quad \mathcal{F}_{\text{row}} = A^{-1}P^{-1}\cdot\mathcal{L}_{\text{col}}\cdot PA.$$

As the transformation matrix is unknown to the algorithm, we can take it to be either A or PA.

A comparison with [8], [4]: In [8, 7], a polynomial time algorithm was given to decompose a semisimple Lie algebra over $\mathbb{Q}$ (more generally, a characteristic 0 field) into a direct sum of simple Lie subalgebras. The Lie algebra $\mathfrak{g}_{\mathsf{Det}}$ is semisimple and $\mathcal{L}_{\text{col}}$ and $\mathcal{L}_{\text{row}}$ are its two simple Lie subalgebras. So, our decomposition problem is a special case of the problem studied in [8]. However, our algorithm works over any sufficiently large field $\mathbb{F}$ (in particular, finite fields), if $\text{char}(F)\nmid n(n-1)$. It is not quite clear to us if the algorithm in [8] (which is somewhat different from our algorithm) can be easily adapted to achieve

the same result in this special case. Lemma 15 shows that the decomposition of $\mathbb{F}^{2r}$ into irreducible invariant subspaces of $\mathcal{R}$ is unique. Using this information, it is possible to use the module decomposition algorithm in [4] to compute bases of $\mathcal{F}_{\text{col}}$ and $\mathcal{F}_{\text{row}}$ in randomized polynomial time over finite fields. However, the module decomposition algorithm in [4] does not work in general over $\mathbb{Q}$ without moving to an extension field.

A comparison with [12, 13]: In [12] and Section 4.9 of [13], a DET over $\mathbb{C}$ was given by reducing it to the Lie algebra conjugacy problem. It was also suggested there that the approach can be made to work over finite field by reduction to the problem of finding and diagonalizing split Cartan subalgebra and then applying Ryba's algorithm [27]. However, it is not quite clear to us how to carry out this approach in full details. Despite the similarities between these two approaches originating from the use of Lie algebra, our approach of reducing DET to FMAI does appear somewhat different from the approach suggested in [12, 13].

4 Reduction of DET to FMAI

We give a randomized polynomial time reduction from DET to the FMAI problem. Recall the FMAI problem from Definition 4: An algorithm for FMAI takes input an ordered basis $(L_1, \ldots, L_m)$ of a $\mathbb{F}$-algebra $\mathcal{A} \subseteq M_s$ such that $\mathcal{A} \cong M_n$, and outputs a $\mathbb{F}$-algebra isomorphism $\phi : \mathcal{A} \to M_n$ in the form of an ordered basis $(C_1, \ldots, C_m)$ of M_n, where $C_i = \phi(L_i)$ for $i \in [m]$.

▶ **Lemma 18** (Reduction of DET to FMAI). *Let $n \geq 2$, $|\mathbb{F}| > 10n^4$ and $\text{char}(\mathbb{F}) \nmid n(n-1)$. Then, there exists a randomized algorithm, with oracle access to FMAI, that takes input black-box access to a $f \in \mathbb{F}[\mathbf{x}]$ of degree n and solves DET for f over $\mathbb{F}$ with high probability. The running time of the algorithm is polynomial in n and the bit length of the coefficients of f.*

The proof of this lemma follows from the proof of correctness of the following algorithm.

4.1 The algorithm

Algorithm 2 Reduction of DET to FMAI.

Input: Black-box access to $f \in \mathbb{F}[\mathbf{x}]$ of degree n, and oracle access to an algo for FMAI.
Output: $B \in \text{GL}(m, \mathbb{F})$ such that $f = \mathsf{Det}(B \cdot \mathbf{x})$, if such a B exists. Else, output "Fail".

1: Invoke Algorithm 1. Let $\{U_1, \ldots, U_r\}$ be the basis of the space $\mathcal{U}_1$ returned by Algorithm 1, where $\mathcal{U}_1 = \mathcal{F}_{\text{col}}$.
2: Generate a basis $\{L_1, \ldots, L_k\}$ of the algebra $\mathcal{A} := \mathbb{F}[U_1, \ldots, U_r]$. If $k \neq m$, output "Fail".

3: Invoke the FMAI oracle on $(L_1, \ldots, L_m)$ which returns a basis $(C_1, \ldots, C_m)$ of M_n.
4: Pick a *random* $M \in M_m$ satisfying $L_i \cdot M = M \cdot (I_n \otimes C_i)$ for every $i \in [m]$.
5: Let b be the evaluation of $f(M \cdot \mathbf{x})$ at $x_{11} = \ldots = x_{nn} = 1$ and remaining x_{ij} set to 0.
6: If $M \notin \text{GL}(m, \mathbb{F})$ or $b = 0$, output "Fail". Else, set $D = \text{diag}(b, 1, \ldots, 1) \in M_n$. Output $(I_n \otimes D) \cdot M^{-1}$.

4.2 Proof of correctness of Algorithm 2

If f is not equivalent to Det then it can be detected with high probability by checking if $f(\mathbf{a}) = b \cdot \mathsf{Det}(M^{-1}\mathbf{a})$ at a random point $\mathbf{a} \in_r S^m$, where $S \subseteq \mathbb{F}$ is sufficiently large. So, assume that $f = \mathsf{Det}(A \cdot \mathbf{x})$ for some $A \in \mathrm{GL}(m, \mathbb{F})$. The correctness of Algorithm 1 ensure that $\mathcal{U}_1 = \mathcal{F}_{\mathrm{col}}$ without loss of generality. Step 2 can be executed efficiently by checking if $U_iU_j \in \mathrm{span}_{\mathbb{F}}\{U_1, \ldots, U_r\}$ for $i, j \in [r]$. Observation 2.5 implies that $\mathcal{A} \cong M_n$, i.e., $L_i = A^{-1} \cdot (I_n \otimes B_i) \cdot A$ for every $i \in [m]$, where $\{B_1, \ldots, B_m\}$ is a basis of M_n. In Step 3, the FMAI oracle returns a $\mathbb{F}$-algebra isomorphism $\phi : \mathcal{A} \to M_n$ such that $\{C_i = \phi(L_i) : i \in [m]\}$ is a basis of M_n. The following claim ensures the existence of a matrix M, computed in Step 4. Its proof is given in Section E.1 of the Appendix of [11].

▷ Claim 19. There exists a $S \in \mathrm{GL}(n, \mathbb{F})$ such that $B_i = S^{-1} \cdot C_i \cdot S$ for every $i \in [m]$.

Consider the linear system defined by the equation $L_i \cdot M = M \cdot (I_n \otimes C_i)$, where the entries of M are taken as variables. Step 4 is executed by picking the free variables of the solution space of the system from a sufficiently large subset of $\mathbb{F}$. Finally, the correctness of Step 6 is argued in the proof of the following claim which is given in Section E.2 of the Appendix of [11].

▷ Claim 20. Suppose $f = \mathsf{Det}(A \cdot \mathbf{x})$, where $A \in \mathrm{GL}(m, \mathbb{F})$. Then, $f = \mathsf{Det}((I_n \otimes D) \cdot M^{-1} \cdot \mathbf{x})$ with high probability.

5 DET over finite fields and over $\mathbb{Q}$

The proofs of Theorem 1 and 2 are completed by replacing the FMAI oracle in Step 3 of Algorithm 2 by the following known algorithms for FMAI over finite fields and $\mathbb{Q}$.

▶ **Theorem 21** (Theorem 5.1 of [25]). *Let $\mathbb{F}$ be a finite field. Given a basis of a $\mathbb{F}$-algebra $\mathcal{A} \subseteq M_m$ such that $\mathcal{A} \cong M_n$, an isomorphism $\phi : \mathcal{A} \to M_n$ can be constructed in randomized $(m, \log |\mathbb{F}|)$ time.*

▶ **Theorem 22** (Theorem 1 of [14]). *There is a randomized algorithm with oracle access to* IntFact *that takes input a basis of a $\mathbb{Q}$-algebra $\mathcal{A} \subseteq M_m$ such that $\mathcal{A} \cong M_n$, and outputs an isomorphism $\phi : \mathcal{A} \to M_n$ with high probability. The algorithm runs in time polynomial in the bit length of the input, if n is bounded.*

▶ **Theorem 23** (Lemma 2.5 of [3]). *There is a randomized algorithm that takes input a basis of a $\mathbb{Q}$-algebra $\mathcal{A} \subseteq M_m$ such that $\mathcal{A} \cong M_n$, and outputs an isomorphism $\phi : \mathcal{A} \otimes_{\mathbb{Q}} Ł \to M_n(Ł)$ with high probability, where Ł is an extension field of $\mathbb{Q}$ satisfying $[Ł : \mathbb{Q}] \leq n$. The algorithm runs in time polynomial in the bit length of the input.*

6 Factoring hardness of DET over $\mathbb{Q}$

This section is devoted to proving Theorem 3. We show that DET in the 2×2 setting over $\mathbb{Q}$ is at least as hard as factoring square-free integers. We will need the following theorem.

▶ **Theorem 24** ([24]). *Assuming GRH, there is a randomized polynomial time reduction from the problem of factoring square-free integers to the following problem: Given non-zero $a, b \in \mathbb{Q}$, find rational numbers x, y, z (not all zero) such that $x^2 - ay^2 - bz^2 = 0$, if there exists such a solution.*

We will also need the following proposition, cited in [24], to prove the next theorem. We give a proof from [5] in Section F.1 of [11], for completeness.

▶ **Proposition 25.** *Let $a, b \in \mathbb{Q}^\times$. Then the equation $x^2 - ay^2 - bz^2 = 0$ has a non-zero rational solution if and only if the equation $x^2 - ay^2 - bz^2 + abw^2 = 0$ has a non-zero rational solution.*

We are now ready to prove integer factoring hardness of DET in the next theorem.

▶ **Theorem 26.** *Consider the polynomial $f_{a,b}(\mathbf{x}) = x_{1,1}^2 - ax_{1,2}^2 - bx_{2,1}^2 + abx_{2,2}^2$, where $a, b \in \mathbb{Q}$ are non-zero. Then $f_{a,b}(\mathbf{x}) = \mathsf{Det}_2(A \cdot \mathbf{x})$ for some $A \in \mathrm{GL}(4, \mathbb{Q})$ if and only if the equation $x^2 - ay^2 - bz^2 = 0$ has a non-zero rational solution (moreover, such a rational solution can be efficiently computed from A).*

Its proof is given in Section F.2 of [11]. Combining Theorems 24 and 26, we obtain Theorem 3.

▶ Remark 27. We want to explain how we got to the above reduction. Ronyai [24] proved that the FMAI problem over $\mathbb{Q}$ is factoring hard even for $n = 2$ via quaternion algebras. If one takes a specific quaternion algebra and tries to constructs a polynomial f whose Lie algebra is the traceless part of the quaternion algebra, then it turns out the polynomial $f_{a,b}(\mathbf{x})$ is the unique homogeneous degree 2 polynomial that comes out. But in any case, in hindsight, the polynomial $f_{a,b}(\mathbf{x})$ seems like a natural candidate to use.

7 Characterization of the determinant by its Lie algebra

In this section, we reduce FMAI to DET under mild restrictions on $\mathbb{F}$. We start with the claim that the Lie algebra of the determinant characterizes the determinant. This is well known over $\mathbb{C}$, but we give a proof in Section G.1 of [11] that works under mild restrictions on $\mathbb{F}$.

▶ **Lemma 28.** *Let $f \in \mathbb{F}[\mathbf{x}]$ be any homogeneous polynomial of degree n such that $\mathcal{L}_{col} \subseteq \mathfrak{g}_f$ (recall $\mathcal{L}_{col}$ from Section 2). Also suppose $\mathrm{char}(\mathbb{F}) \nmid n$. Then $f(\mathbf{x}) = \alpha \cdot \mathsf{Det}_n(\mathbf{x})$ for some $\alpha \in \mathbb{F}$.*

▶ Remark 29. Note that without the $\mathrm{char}(\mathbb{F}) \nmid n$ condition, Lemma 28 is not true. For example, $f(\mathbf{x}) = x_{1,1}^n + \mathsf{Det}_n(\mathbf{x})$ will have the same Lie algebra as $\mathsf{Det}_n(\mathbf{x})$ if $\mathrm{char}(\mathbb{F}) \mid n$.

▶ **Corollary 30.** *Let $f \in \mathbb{F}[\mathbf{x}]$ be a degree n homogeneous polynomial. Suppose that $A^{-1} \cdot \mathcal{L}_{col} \cdot A \subseteq \mathfrak{g}_f$ for some $A \in \mathrm{GL}(n^2, \mathbb{F})$ and $\mathrm{char}(\mathbb{F}) \nmid n$. Then $f(\mathbf{x}) = \alpha \cdot \mathsf{Det}_n(A \cdot \mathbf{x})$ for some $\alpha \in \mathbb{F}$.*

Proof. Consider $f'(\mathbf{x}) = f(A^{-1} \cdot \mathbf{x})$. By Fact 2, $\mathfrak{g}_{f'} = A \cdot \mathfrak{g}_f \cdot A^{-1}$, so $\mathcal{L}_{\mathrm{col}} \subseteq \mathfrak{g}_{f'}$. By Lemma 28, we get that $f'(\mathbf{x}) = \alpha \cdot \mathsf{Det}_n(\mathbf{x})$ for some $\alpha \in \mathbb{F}$ and hence $f(\mathbf{x}) = \alpha \cdot \mathsf{Det}_n(A \cdot \mathbf{x})$. ◀

Corollary 30 allows us to reduce FMAI to DET when n is constant (see Algorithm 3).

7.1 Proof of correctness of Algorithm 3 when $\mathrm{char}(\mathbb{F}) \nmid n$

The proof of correctness will follow from the following proposition, proved in Section G.2 of [11]. The matrices $B_{i,j}$ and $L_{i,j}$ are as defined in Step 2 of the algorithm.

▶ **Proposition 31.** *Suppose the algebra $\mathcal{A}$ spanned by $B_{1,1}, \ldots, B_{n,n}$ is isomorphic to M_n. Then there exist $K \in \mathrm{GL}(n^2, \mathbb{F})$ and $C_{1,1}, \ldots, C_{n,n} \in M_n$, s.t. $L_{i,j} = K^{-1}(I_n \otimes C_{i,j})K$ for all $i, j \in [n]$.*

Now let us proceed to the proof of correctness of Algorithm 3. First of all, it is easy to ensure that whenever the algorithm outputs an isomorphism, it is actually an isomorphism. So what we need to prove is the converse. Suppose the algebra $\mathcal{A}$ is isomorphic to M_n. Then by Proposition 31, the space spanned by $\tilde{L}_1, \ldots, \tilde{L}_{n^2-1}$ is $K^{-1} \cdot \mathcal{L}_{\text{col}} \cdot K$. Then by Corollary 30, there is a unique solution to the equations in Step 4 given by $f(\mathbf{x}) = \alpha \cdot \mathsf{Det}_n(K \cdot \mathbf{x})$, for some $\alpha \in \mathbb{F}$, and so f is equivalent to the determinant. Hence, in Step 5, we will get an $A \in \mathrm{GL}(n^2, \mathbb{F})$ s.t. $f(\mathbf{x}) = \mathsf{Det}_n(A \cdot \mathbf{x})$. Since $\tilde{L}_1, \ldots, \tilde{L}_{n^2-1}$ span a Lie algebra of dimension n^2-1 and since they lie inside the Lie algebra of $\mathsf{Det}_n(A \cdot \mathbf{x})$, we must have that $\tilde{L}_1, \ldots, \tilde{L}_{n^2-1}$ span either $A^{-1} \cdot \mathcal{L}_{\text{col}} \cdot A$ or $A^{-1} \cdot \mathcal{L}_{\text{row}} \cdot A$. From this, we get that one of the following conditions should be true:

- There exist matrices $F_{1,1}, \ldots, F_{n,n} \in M_n$ such that $A \cdot L_{i,j} \cdot A^{-1} = I_n \otimes F_{i,j}$ for all $i, j \in [n]$.
- There exist matrices $F_{1,1}, \ldots, F_{n,n} \in M_n$ such that $A \cdot L_{i,j} \cdot A^{-1} = F_{i,j} \otimes I_n$ for all $i, j \in [n]$.

The implies that the algorithm will output 1 and an isomorphism into M_n. The complexity of the reduction is dominated by Step 4 which takes $n^{O(n)}$ field operations.

Algorithm 3 Reduction of FMAI to DET.

Input: Basis $\{B_1, \ldots, B_r\}$ of a $\mathbb{F}$-algebra $\mathcal{A} \subseteq M_m$, and access to an algorithm for DET.
Output: if $\mathcal{A} \cong M_n$ for some $n \in \mathbb{N}$, then output an isomorphism, 0 otherwise.

1: If $r = \dim_{\mathbb{F}} \mathcal{A} \neq n^2$ for any $n \in \mathbb{N}$, output 0 and halt.
2: Index the basis elements by $[n] \times [n]$, i.e., rename them as $B_{1,1}, \ldots, B_{n,n}$. Compute $n^2 \times n^2$ matrices $L_{1,1}, \ldots, L_{n,n}$ as follows: $L_{i,j}$ is the matrix corresponding to the left-multiplication action of $B_{i,j}$ on $B_{1,1}, \ldots, B_{n,n}$. That is $B_{i,j} \cdot B_{i_2,j_2} = \sum_{i_1,j_1} L_{i,j}((i_1,j_1),(i_2,j_2)) \cdot B_{i_1,j_1}$.
3: Compute a basis for the traceless parts of the matrices $L_{i,j}$. That is, compute a basis $\tilde{L}_1, \ldots, \tilde{L}_s$ of the space spanned by $L_{1,1} - \frac{\mathrm{tr}(L_{1,1})}{n^2} I_{n^2}, \ldots, L_{n,n} - \frac{\mathrm{tr}(L_{n,n})}{n^2} I_{n^2}$. If $s \neq n^2 - 1$, output 0 and halt.
4: Find a non-zero homogeneous polynomial of degree n, $f(\mathbf{x})$, satisfying the equations

$$\sum_{i_1,j_1,i_2,j_2} M((i_1,j_1),(i_2,j_2)) \cdot x_{i_2,j_2} \cdot \frac{\partial f}{\partial x_{i_1,j_1}} = 0$$

for every $M \in \{\tilde{L}_1, \ldots, \tilde{L}_{n^2-1}\}$ (these give linear equations in the coefficients of f). If no such non-zero polynomial exists then output 0 and halt.
5: Run DET on f. If the output is "Fail" then output 0 and halt. If $f(\mathbf{x}) = \mathsf{Det}_n(A \cdot \mathbf{x})$ then check if there exist matrices $F_{1,1}, \ldots, F_{n,n} \in M_n$ such $A \cdot L_{i,j} \cdot A^{-1} = I_n \otimes F_{i,j}$ for all i, j. If yes, output 1 and the isomorphism $\phi(B_{i,j}) = F_{i,j}$ (extended linearly to whole of $\mathcal{A}$). If no, check if there exist matrices $F_{1,1}, \ldots, F_{n,n} \in M_n$ such that $A \cdot L_{i,j} \cdot A^{-1} = F_{i,j} \otimes I_n$ for all i, j. If yes, output 1 and the isomorphism $\phi(B_{i,j}) = F_{i,j}$ (extended linearly to whole of $\mathcal{A}$). If no, output 0.

References

1. Manindra Agrawal and Nitin Saxena. Automorphisms of Finite Rings and Applications to Complexity of Problems. In *STACS 2005, 22nd Annual Symposium on Theoretical Aspects of Computer Science, Stuttgart, Germany, February 24-26, 2005, Proceedings*, pages 1–17, 2005.
2. Manindra Agrawal and Nitin Saxena. Equivalence of F-Algebras and Cubic Forms. In *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23-25, 2006, Proceedings*, pages 115–126, 2006.
3. László Babai and Lajos Rónyai. Computing irreducible representations of finite groups. *Mathematics of Computation*, 55(192):705–722, 1990.
4. Alexander L. Chistov, Gábor Ivanyos, and Marek Karpinski. Polynomial Time Algorithms for Modules over Finite Dimensional Algebras. In *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation, ISSAC '97, Maui, Hawaii, USA, July 21-23, 1997*, pages 68–74, 1997.
5. Keith Conrad. Quaternion algebras, 2016.
6. J. E. Cremona, T. A. Fisher, C. O'Neil, D. Simon, and M. Stoll. Explicit n-descent on elliptic curves III. Algorithms. *Math. Comput.*, 84(292):895–922, 2015. `arXiv:1107.3516`.
7. W.A. de Graaf. *Algorithms for Finite-Dimensional Lie Algebras*. PhD thesis, Technical University of Eindhoven, 1997.
8. W.A. de Graaf. Calculating the structure of a semisimple Lie algebra. *Journal of Pure and Applied Algebra*, 117-118:319–329, 1997.
9. Wayne Eberly. Decompositions of algebras over R and C. *computational complexity*, 1(3):211–234, September 1991.
10. W.M. Eberly. *Computations for algebras and group representations*. PhD thesis, Department of Computer Science, University of Toronto, 1989.
11. Ankit Garg, Nikhil Gupta, Neeraj Kayal, and Chandan Saha. Determinant equivalence test over finite fields and over $\mathbf{Q}$. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:42, 2019. URL: `https://eccc.weizmann.ac.il/report/2019/042`.
12. Joshua A. Grochow. Matrix Lie algebra isomorphism. In *IEEE Conference on Computational Complexity (CCC12)*, pages 203–213, 2012.
13. Joshua Abraham Grochow. *Symmetry and equivalence relations in classical and geometric complexity theory*. PhD thesis, Department of Computer Science, The University of Chicago, Chicago, Illinois, 2012.
14. Gábor Ivanyos, Lajos Rónyai, and Josef Schicho. Splitting full matrix algebras over algebraic number fields. *Journal of Algebra*, 354:211–223, 2012. `arXiv:1106.6191`.
15. Neeraj Kayal. Efficient algorithms for some special cases of the polynomial equivalence problem. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1409–1421, 2011.
16. Neeraj Kayal. Affine projections of polynomials: extended abstract. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 643–662, 2012.
17. Neeraj Kayal, Vineet Nair, and Chandan Saha. Average-case linear matrix factorization and reconstruction of low width Algebraic Branching Programs. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:29, 2018.
18. Neeraj Kayal, Vineet Nair, Chandan Saha, and Sébastien Tavenas. Reconstruction of Full Rank Algebraic Branching Programs. In *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, pages 21:1–21:61, 2017.
19. Falko Lorenz. *Algebra Volumne 2: Fields with structures, Algebras and advanced topics*. Springer, 2008.
20. Meena Mahajan and V. Vinay. A Combinatorial Algorithm for the Determinant. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 5-7 January 1997, New Orleans, Louisiana, USA.*, pages 730–738, 1997.

21 Meena Mahajan and V. Vinay. Determinant: Combinatorics, Algorithms, and Complexity. *Chicago J. Theor. Comput. Sci.*, 1997, 1997.

22 Ketan Mulmuley and Milind A. Sohoni. Geometric Complexity Theory I: An Approach to the P vs. NP and Related Problems. *SIAM J. Comput.*, 31(2):496–526, 2001.

23 Jacques Patarin. Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, pages 33–48, 1996.

24 Lajos Rónyai. Simple Algebras Are Difficult. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC), 1987, New York, New York, USA*, pages 398–408, 1987.

25 Lajos Rónyai. Computing the Structure of Finite Algebras. *J. Symb. Comput.*, 9(3):355–373, 1990.

26 Lajos Rónyai. A Deterministic Method for Computing Splitting Elements in Simple Algebras over Q. *Journal of Algorithms*, 16:24–32, 1994.

27 Alexander J.E. Ryba. Computer construction of split Cartan subalgebras. *J. Algebra*, 309:455–483, 2007.

28 Nitin Saxena. *Morphisms of rings and applications to complexity.* PhD thesis, Indian Institute of Technology Kanpur, 2006.

29 Thomas Thierauf. The Isomorphism Problem for Read-Once Branching Programs and Arithmetic Circuits. *Chicago J. Theor. Comput. Sci.*, 1998, 1998.

30 Leslie G. Valiant. Completeness Classes in Algebra. In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 249–261, 1979.

31 Lars Ambrosius Wallenborn. Computing the Hilbert symbol, quadratic form equivalence and integer factoring. Diploma thesis, University of Bonn, 2013.

Non-Clairvoyant Precedence Constrained Scheduling

Naveen Garg
Computer Science and Engineering Department, Indian Institute of Technology, Delhi, India
naveen@cse.iitd.ac.in

Anupam Gupta
Computer Science Department, Carnegie Mellon University, USA
anupamg@cmu.edu

Amit Kumar
Computer Science and Engineering Department, Indian Institute of Technology, Delhi, India
amitk@cse.iitd.ac.in

Sahil Singla
Princeton University and Institute for Advanced Study, USA
singla@cs.princeton.edu

Abstract

We consider the online problem of scheduling jobs on identical machines, where jobs have precedence constraints. We are interested in the demanding setting where the jobs sizes are not known up-front, but are revealed only upon completion (the *non-clairvoyant* setting). Such precedence-constrained scheduling problems routinely arise in map-reduce and large-scale optimization. For minimizing the total weighted completion time, we give a constant-competitive algorithm. And for total weighted flow-time, we give an $O(1/\varepsilon^2)$-competitive algorithm under $(1+\varepsilon)$-speed augmentation and a natural "no-surprises" assumption on release dates of jobs (which we show is necessary in this context).

Our algorithm proceeds by assigning *virtual rates* to all waiting jobs, including the ones which are dependent on other uncompleted jobs. We then use these virtual rates to decide on the actual rates of minimal jobs (i.e., jobs which do not have dependencies and hence are eligible to run). Interestingly, the virtual rates are obtained by allocating time in a fair manner, using a Eisenberg-Gale-type convex program (which we can solve optimally using a primal-dual scheme). The optimality condition of this convex program allows us to show dual-fitting proofs more easily, without having to guess and hand-craft the duals. This idea of using fair virtual rates may have broader applicability in scheduling problems.

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

Keywords and phrases Online algorithms, Scheduling, Primal-Dual analysis, Nash welfare

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.63

Category Track A: Algorithms, Complexity and Games

Related Version `https://arxiv.org/pdf/1905.02133.pdf`

Funding This research was supported in part by NSF awards CCF-1536002, CCF-1540541, and CCF-1617790, and the Indo-US Joint Center for Algorithms Under Uncertainty. Sahil Singla was supported in part by the Schmidt Foundation.

1 Introduction

We consider the problem of online scheduling of jobs under precedence constraints. We seek to minimize the average weighted flow time of the jobs on multiple parallel machines, in the online *non-clairvoyant* setting. Formally, there are m identical machines, each capable

of one unit of processing per unit of time. A set of $[n]$ jobs arrive online. Each job has a processing requirement p_j and a weight w_j, and is released at some time r_j. If the job finishes at time C_j, its flow or response time is defined to be $C_j - r_j$. The goal is to give a preemptive schedule that minimizes the total (or, equivalently, the average) weighted flow-time $\sum_{j\in[n]} w_j \cdot (C_j - r_j)$. The main constraints of our model are the following: (i) the scheduling is done *online*, so the scheduler does not know of the jobs before they are released; (ii) the scheduler is *non-clairvoyant* – when a job arrives, the scheduler knows its weight but *not its processing time* p_j. (It is only when the job finishes its processing that the scheduler knows the job is done, and hence knows p_j.); And (iii) there are *precedence constraints* between jobs given by a partial order $([n], \prec)$: $j \prec j'$ means job j' cannot be started until j is finished. Naturally, the partial order should respect release dates: if $j \prec j'$ then $r_j \le r'_j$. (We will require a stronger assumption for some of our results.)

This model for constrained parallelism is a natural one, both in theory and in practice. In theory, this precedence-constrained (and non-clairvoyant!) scheduling model (with other objective functions) goes back to Graham's work on list scheduling [8]. In practice, most languages and libraries produce parallel code that can be modeled using precedence DAGs [20, 1, 9]. Often these jobs (i.e., units of processing) are distributed among some m workstations or servers, either in server farms or on the cloud, i.e., they use identical parallel machines.

1.1 Our Results and Techniques

Weighted Completion Time. We develop our techniques on the problem of minimizing the *average weighted completion time* $\sum_j w_j C_j$. Our convex-programming approach gives us:

▶ **Theorem 1.1.** *There is a* 10*-competitive deterministic online algorithm for minimizing the average weighted completion time on parallel machines with both release dates and precedences, in the online non-clairvoyant setting.*

For this result, at each time t, the algorithm has to know only the partial order restricted to $\{j \in [n] \mid r_j \le t\}$, i.e., the jobs released by time t. The algorithmic idea is simple in hindsight: the algorithm looks at the *minimal* unfinished jobs (i.e., they do not depend on any other unfinished jobs): call them I_t. If J_t is the set of (already released and) unfinished jobs at time t, then $I_t \subseteq J_t$. To figure out how to divide our processing among the jobs in I_t, we write a convex program that fairly divides the time among all jobs in the larger set J_t, such that (a) these jobs can "donate" their allocated time to some preceding jobs in I_t, and that (b) the jobs in I_t do not get more than 1 unit of processing per time-step.

For this fair allocation, we maximize the (weighted) Nash Welfare $\sum_{j\in J_t} w_j \log R_j$, where R_j is the *virtual rate* of processing given to job $j \in J_t$, regardless of whether it can currently be run (i.e., is in I_t). This tries to fairly distribute the virtual rates among the jobs [19], and can be solved using an Eisenberg-Gale-type convex program. (We can solve this convex program in our setting using a simple primal-dual algorithm, see the full version.) The proof of Theorem 1.1 is via writing a linear-programming relaxation for the weighted completion time problem, and fitting a dual to it. Conveniently, the dual variables for the completion time LP naturally fall out of the dual (KKT) multipliers for the convex program!

Weighted Flow Time. We then turn to the *weighted flow-time minimization* problem. We first observe that the problem has no competitive algorithm if there are jobs j that depend on jobs released before r_j. Indeed, if OPT ever has an empty queue while the algorithm is processing jobs, the adversary could give a stream of tiny new jobs, and we would be sunk. Hence we make an additional *no-surprises* assumption about our instance: when a job j is

released, all the jobs having a precedence relationship to j are also released at the same time. In other words, the partial order is a collection of disjoint connected DAGs, where all jobs in each connected component have the same release date. A special case of this model has been studied in [20, 1] where each DAG is viewed as a "hyper-job" and there are no precedence constraints between different hyper-jobs. In this model, we show:

▶ **Theorem 1.2.** *There is an $O(1/\varepsilon^2)$-competitive deterministic non-clairvoyant online algorithm for the problem of minimizing the average weighted flow time on parallel machines with release dates and precedences, under the no-surprises and $(1+\varepsilon)$-speedup assumptions.*

Interestingly, the algorithm for weighted flow-time is almost the same as for weighted completion time. In fact, *exactly* the same algorithm works for both the completion time and flow time cases, if we allow a speedup of $(2+\varepsilon)$ for the latter. To get the $(1+\varepsilon)$-speedup algorithm, we give preference to the recently-arrived jobs, since they have a smaller current time-in-system and each unit of waiting proportionally hurts them more. This is along the lines of strategies like LAPS and WLAPS [7].

1.2 The Intuition

Consider the case of unit weight jobs on a single machine. Without precedence constraints, the round-robin algorithm, which runs all jobs at the same rate, is $O(1)$-competitive for the flow-time objective with a 2-speed augmentation. Now consider precedences, and let the partial order be a collection of disjoint chains: only the first remaining job from each chain can be run at each time. We generalize round-robin to this setting by running all minimal jobs simultaneously, but at rates proportional to length of the corresponding chains. We can show this algorithm is also $O(1)$-competitive with a 2-speed augmentation. While this is easy for chains and trees, let us now consider the case when the partial order is the union of general DAGs, where each DAG may have several minimal jobs. Even though the sum of the rates over all the minimal jobs in any particular DAG should be proportional to the number of jobs in this DAG, running all minimal jobs at equal rates does not work. (Indeed, if many jobs depend on one of these minimal jobs, and many fewer depend on the other minimal jobs in this DAG, we want to prioritize the former.)

Instead, we use a convex program to find rates. Our approach assigns a "virtual rate" R_j to each job in the DAG (regardless of whether it is minimal or not). This virtual rate allows us to ensure that even though this job may not run, it can help some minimal jobs to run at higher rates. This is done by an assignment problem where these virtual rates get translated into actual rates for the minimal jobs. The virtual rates are then calculated using Nash fairness, which gives us max-min properties that are crucial for our analysis.

Analysis Challenges: In typical applications of the dual-fitting technique, the dual variables for each job encode the *increase in total flow-time* caused by arrival of this job. Using this notion turns out to create problems. Indeed, consider a minimal job of low weight which is running at a high rate (because a large number of jobs depend on it). The increase in overall flow-time because of its arrival is very large. However the dual LP constraints require these dual variables to be bounded by the weights of their jobs, which now becomes difficult to ensure. To avoid this, we define the dual variables directly in terms of the virtual rates of the jobs, given by the convex program.

Having multiple machines instead of a single machine creates new problems. The *actual rates* assigned to any minimal job cannot exceed 1, and hence we have to throttle certain actual rates. Again the versatility of the convex program helps us, since we can add this as a

constraint. Arguing about the optimal solution to such a convex program requires dealing with the suitable KKT conditions, from which we can infer many useful properties. We also show in the full version that the optimal solution corresponds to a natural "water-filling" based algorithm.

Finally, we obtain matching results for the case of $(1+\varepsilon)$-speed augmentation. Im et al. [12] gave a general-purpose technique to translate a round-robin based algorithm to a LAPS-like algorithm. In our setting, it turns out that the LAPS-like policy needs to be run on the virtual rates of jobs. Analyzing this algorithm does not follow in a black-box manner (as prescribed by [12]), and we need to adapt our dual-fitting analysis suitably.

1.3 Related Work and Organization

Completion Time. Minimizing $\sum_j w_j C_j$ on parallel machines with precedence constraints has $O(1)$-approximations in the *offline* setting: Li [16] improves on [11, 18] to give a $3.387+\varepsilon$-approximation. For *related* machines, the precedence constraints make the problem much harder: there is a $O(\log m/\log\log m)$-approximation [16] improving on a prior $O(\log m)$ result [4], and a hardness of $\omega(1)$ under certain complexity assumptions [3]. In the *online* setting, any offline algorithm for (a dual problem to) $\sum_j w_j C_j$ gives an *clairvoyant* online algorithm, losing $O(1)$ factors [11]. Two caveats: it is unclear (a) how to make this algorithm non-clairvoyant, and (b) how to solve the (dual of the) weighted completion time problem with precedences in poly-time.

Flow Time without Precedence. To minimize $\sum_j w_j(C_j - r_j)$, strong lower bounds are known for the competitive ratio of any online algorithm even on a single machine [17]. Hence we use speed augmentation [14]. For the general setting of non-clairvoyant weighted flow-time on unrelated machines, Im et al. [13] showed that weighted round-robin with a suitable migration policy yields a $(2+\varepsilon)$-competitive algorithm using $(1+\varepsilon)$-speed augmentation. They gave a general purpose technique, based on the LAPS scheduling policy, to convert any such round-robin based algorithm to a $(1+\varepsilon)$-competitive algorithm while losing an extra $1/\varepsilon$ factor in the competitive ratio. Their analysis also uses a dual-fitting technique [2, 10]. However, they do not consider precedence constraints.

Flow Time with Precedence. Much less is known for flow-time problems with precedence constraints. For the offline setting on identical machines, [15] give $O(1)$-approximations with $O(1)$-speedup, even for general delay functions. In the current paper, we achieve a $\mathrm{poly}(1/\varepsilon)$-approximation with $(1+\varepsilon)$-speedup for flow-time. Interestingly, [15] show that beating a n^{1-c}-approximation for any constant $c \in [0,1)$ requires a speedup of at least the optimal approximation factor of makespan minimization in the same machine environment. However, this lower bound requires different jobs with a precedence relationship to have different release dates, which is something our model disallows. (The full version gives another lower bound showing why we disallow such precedences in the online setting.)

In the *online* setting, [20] introduced the DAG model where each *job* is a directed acyclic graph (of *tasks*) released at some time, and a job/DAG completes when all the tasks in it are finished, and we want to minimize the total *unweighted* flow-time. They gave a $(2+\varepsilon)$-speed $O(\kappa/\varepsilon)$-competitive algorithm, where κ is the largest antichain within any job/DAG. [1] show $\mathrm{poly}(1/\varepsilon)$-competitiveness with $(1+\varepsilon)$-speedup, again in the non-clairvoyant setting. The case where jobs are entire DAGs, and not individual nodes within DAGs, is captured in our weighted model by putting zero weights for all original jobs, and adding a unit-weight zero-sized job for each DAG which now depends on all jobs in the DAG. Assigning arbitrary

weights to individual nodes within DAGs makes our problem quite non-trivial – we need to take into account the structure of the DAG to assign rates to jobs. Another model to capture parallelism and precedences uses *speedup functions* [6, 5, 7]: relating our model to this setting remains an open question.

Our work is closely related to Im et al. [12] who use a Nash fairness approach for completion-time and flow-time problems with multiple resources. While our approaches are similar, to the best of our understanding their approach does not immediately extend to the setting with precedences. Hence we have to introduce new ideas of using virtual rates (and being fair with respect to them), and throttling the induced actual rates at 1. The analyses of [12] and our work are both based on dual-fitting; however, we need some new ideas for the setting with precedences.

Organization. The weighted completion time case is solved in §2. A $(2+\varepsilon)$-speedup result for weighted flow-time is in §3. In the full version we improve this to a $(1+\varepsilon)$-speedup. There we also show the need for the "no-surprises" assumption on release dates, how to solve the convex program using a "water-filling" based algorithm, and the missing proofs.

2 Minimizing Weighted Completion Time

In this section, we describe and analyze the scheduling algorithm for the problem of minimizing weighted completion time on parallel machines. Recall that the precedence constraints are given by a DAG G, and each job j has a release date r_j, processing size p_j and weight w_j.

2.1 The Scheduling Algorithm

We first assume that each of the m machines run at rate 2 (i.e., they can perform 2 units of processing in a unit time). We will show later how to remove this assumption (at a constant loss of competitive ratio). We begin with some notation. We say that a job j is *waiting* at time t (with respect to a schedule) if $r_j \le t$, but j has not been processed to completion by time t. We use J_t to denote the set of waiting jobs at time t. Note that at time t, the algorithm gets to see the subgraph G_t of G which is induced by the jobs in J_t. We say that a job j is *unfinished* at time t if it is either waiting at time t, or its release date is at least t (and hence the algorithm does not even know about this job). Let U_t denote the set of unfinished jobs at time t. Clearly, $J_t \subseteq U_t$. At time t, the algorithm can only process those jobs in J_t which do not have a predecessor in G_t – denote these *minimal* jobs by I_t: they are independent of all other current jobs. For every time t, the scheduling algorithm needs to assign a rate to each job $j \in I_t$. We now describe how it decides on these rates.

Consider a time t. The algorithm considers a bipartite graph $H_t = (I_t, J_t, E_t)$ with vertex set consisting of the minimal jobs I_t on left and the waiting jobs J_t on right. Since $I_t \subseteq J_t$, a job in I_t appears as a vertex on both sides of this bipartite graph. When there is no confusion, we slightly overload terminology by referring to a job as a vertex in H_t. The set of edges E_t are as follows: let $j_l \in I_t, j_r \in J_t$ be vertices on the left and the right side respectively. Then (j_l, j_r) is an edge in E_t if and only if there is a directed path from j_l to j_r in the DAG G_t.

The following convex program now computes the rate for each vertex in I_t. It has variables z_e^t for each edge $e \in E_t$. For each job j on the left side, i.e., for $j \in I_t$, define $L_j^t := \sum_{e \in \partial j} z_e^t$ as the sum of z_e values of edges incident to j. Similarly, define $R_j^t := \sum_{e \in \partial j} z_e^t$ for a job $j \in J_t$, i.e., on the right side. The objective function is the Nash bargaining objective function on the R_j^t values, which ensures that each waiting job gets some attention. In the full version

we give a combinatorial algorithm to efficiently solve this convex program.

$$\max \sum_{j \in J_t} w_j \ln R_j^t \tag{CP}$$

$$L_j^t = \sum_{j' \in J_t : (j,j') \in E_t} z_{jj'}^t \qquad \forall j \in I_t \tag{1}$$

$$R_j^t = \sum_{j' \in I_t : (j',j) \in E_t} z_{j'j}^t \qquad \forall j \in J_t \tag{2}$$

$$L_j^t \le 1 \qquad \forall j \in I_t \tag{3}$$

$$\sum_{j \in I_t} L_j^t \le m \tag{4}$$

$$z_e^t \ge 0 \qquad \forall e \in E_t \tag{5}$$

Let $(\bar{z}^t, \bar{L}^t, \bar{R}^t)$ be an optimal solution to the above convex program. We define the *rate of a job* $j \in I_t$ as being $\bar{L}_j^t$.

Although we have defined this as a continuous time process, it is easy to check that the rates only change if a new job arrives, or if a job completes processing. Also observe that we have effectively combined the m machines into one in this convex program. But assuming that all events happen at integer times, we can translate the rate assignment to an actual schedule as follows. For a time slot $[t, t+1]$, the total rate is at most m (using (4)), so we create m time slots $[t, t+1]_i$, one for each machine i, and iteratively assign each job j an interval of length $\bar{L}_j^t$ within these time slots. It is possible that a job may get assigned intervals in two different time slots, but the fact that $\bar{L}_j^t \le 1$ means it will not be assigned the same time in two different time slots. Further, we will never exceed the slots because of (4). Thus, we can process these jobs in the m time slots on the m parallel machines such that each job j gets processed for $\bar{L}_j^t$ amount of time and no job is processed concurrently on multiple machines. This completes the description of the algorithm; in this, we assume that we run the machines at twice the speed. Call this algorithm $\mathcal{A}$.

The final algorithm $\mathcal{B}$, which is only allowed to run the machines at speed 1, is obtained by running $\mathcal{A}$ in the background, and setting $\mathcal{B}$ to be a slowed-down version of $\mathcal{A}$. Formally, if $\mathcal{A}$ processes a job j on machine i at time $t \in \mathbb{R}_{\ge 0}$, then $\mathcal{B}$ processes this at time $2t$. This completes the description of the algorithm.

2.2 A Time-Indexed LP formulation

We use the dual-fitting approach to analyze the above algorithm. We write a time-indexed linear programming relaxation (LP) for the weighted completion time problem, and use the solutions to the convex program (CP) to obtain feasible primal and dual solutions for (LP) which differ by only a constant factor.

We divide time into integral time slots (assuming all quantities are integers). Therefore, the variable t will refer to integer times only. For every job j and time t, we have a variable $x_{j,t}$ which denotes the volume of j processed during $[t, t+1]$. Note that this is defined only for $t \ge r_j$. The LP relaxation is as follows:

$$\min \quad \textstyle\sum_{j,t} w_j \cdot \frac{t \cdot x_{j,t}}{p_j} \tag{LP}$$

$$\textstyle\sum_{t \ge r_j} \frac{x_{j,t}}{p_j} \ge 1 \quad \forall j \tag{6}$$

$$\textstyle\sum_j x_{j,t} \le m \quad \forall t \tag{7}$$

$$\textstyle\sum_{s \le t} \frac{x_{j,s}}{p_j} \ge \sum_{s \le t} \frac{x_{j',s}}{p_{j'}} \quad \forall t, j \prec j' \tag{8}$$

The following claim, whose proof is deferred to the full version, shows that it is a valid relaxation.

▷ Claim 2.1. Let *opt* denote the weighted completion time of an optimal off-line policy (which knows the processing time of all the jobs). Then the optimal value of the LP relaxation is at most *opt*.

The (LP) has a large integrality gap. Observe that the LP just imagines the m machines to be a single machine with speed m. Therefore, (LP) has a large integrality gap for two reasons: (i) a job j can be processed concurrently on multiple machines, and (ii) suppose we have a long chain of jobs of equal size in the DAG G. Then the LP allows us to process all these jobs at the same rate in parallel on multiple machines. We augment the LP lower bound with another quantity and show that the sum of these two lower bounds suffice.

A *chain* C in G is a sequence of jobs $j_1, \ldots, j_k$ such that $j_1 \prec j_2 \prec \ldots \prec j_k$. Define the processing time of C, $p(C)$, as the sum of the processing time of jobs in C. For a job j, define chain_j as the maximum over all chains C ending in j of $p(C)$. It is easy to see that $\sum_j w_j \cdot (r_j + \mathsf{chain}_j)$ is a lower bound (up to a factor 2) on the objective of an optimal schedule.

We now write down the dual of the LP relaxation above. We have dual variables α_j for every job j, and β_t for every time t, and $\gamma_{s,j\to j'}$

$$\max \quad \sum_j \alpha_j - m \sum_t \beta_t \tag{DLP}$$

$$\alpha_j - w_j \cdot t + \sum_{s \geq t} \Big(\sum_{j \prec j'} \gamma_{s,j\to j'} - \sum_{j' \prec j} \gamma_{s,j'\to j} \Big) \leq p_j \cdot \beta_t \quad \forall j, t \geq r_j \tag{9}$$

$$\alpha_j, \beta_t \geq 0$$

We write the dual constraint (9) in a more readable manner. For a job j and time s, let $\gamma^{\text{in}}_{s,j}$ denote $\sum_{j' \prec j} \gamma_{s,j'\to j}$, and define $\gamma^{\text{out}}_{s,j}$ similarly. We now write the dual constraint (9) as

$$\alpha_j - w_j \cdot t + \sum_{s \geq t} \left(\gamma^{\text{out}}_{s,j} - \gamma^{\text{in}}_{s,j} \right) \leq p_j \cdot \beta_t \quad \forall j, t \geq r_j \tag{10}$$

2.3 Properties of the Convex Program

We now prove certain properties of an optimal solution $(\bar{z}^t, \bar{L}^t, \bar{R}^t)$ to the convex program (CP). The first property, whose proof is deferred to the full version, is easy to see:

▷ Claim 2.2. If $\sum_{j \in I_t} \bar{L}^t_j < m$, then $\bar{L}^t_j = 1$ for all $j \in I_t$.

We now write down the KKT conditions for the convex program. (In fact, we can use (1) and (2) to replace $\bar{L}^t_j$ and $\bar{R}^t_j$ in the objective and the other constraints.) Then letting $\theta^t_j \geq 0, \eta^t \geq 0, \nu^t_e \geq 0$ be the Lagrange multipliers corresponding to constraints (3), (4) and (5), we get

$$\frac{w_j}{\bar{R}^t_j} = \theta^t_{j'} + \eta^t - \nu^t_e \qquad \forall e = (j', j), j' \in I_t, j \in J_t \tag{11}$$

$$\theta^t_j \, (\bar{L}^t_j - 1) = 0 \qquad \forall j \in I_t \tag{12}$$

$$\eta^t \, (\textstyle\sum_{j \in I_t} \bar{L}^t_j - m) = 0 \tag{13}$$

$$\nu^t_e \cdot \bar{z}^t_e = 0 \qquad \forall e \in E_t \tag{14}$$

We derive a few consequences of these conditions, the proofs are deferred to the full version.

▷ Claim 2.3. Consider a job $j \in J_t$ on the right side of H_t. Then $w_j \geq \bar{R}^t_j \cdot \eta^t$.

▷ Claim 2.4. Consider a job $j \in J_t$ on the right side of H_t. Suppose j has a neighbor $j' \in I_t$ such that $\bar{L}^t_{j'} < 1$ and $\bar{z}^t_{j'j} > 0$. Then $w_j = \bar{R}^t_j \cdot \eta^t$.

A crucial notion is that of an *active* job:

▶ **Definition 2.5** (Active Jobs). *A job $j \in J_t$ is* active *at time t if it has at least one neighbor in I_t (in the graph H_t) running at rate strictly less than 1.*

Let J^{act}_t denote the set of active jobs at time t. We can strengthen the above claim as follows.

▶ **Corollary 2.6.** *Consider an active job j at time t. Then $w_j = \bar{R}^t_j \cdot \eta^t$.*

▷ Claim 2.7. $w(J^{\text{act}}_t)/m \leq \eta^t \leq w(J_t)/m$.

2.4 Analysis via Dual Fitting

We analyze the algorithm $\mathcal{A}$ first. We define feasible dual variables for (DLP) such that the value of the dual objective function (along with the chain_j values that capture the maximum processing time over all chains ending in j) forms a lower bound on the weighted completion time of our algorithm. Intuitively, α_j would be the weighted completion time of j, and β_t would be $1/2m$ times the total weight of unfinished jobs at time t. Thus, $\sum_j \alpha_j - m \sum_t \beta_t$ would be at $1/2$ times the total weighted completion time. This idea works as long as all the machines are busy at any point of time, the reason being that the primal LP essentially views the m machines as a single speed-m machine. Therefore, we can generate enough dual lower bound if the rate of processing in each time slot is m. If all machines are not busy, we need to appeal to the lower bound given by the chain_j values.

We use the notation used in the description of the algorithm. In the graph H_t, we had assigned rates $\bar{L}^t_j$ to all the nodes j in I_t. Recall that a vertex $j \in J_t$ on the right side of H_t is said to be *active* at time t if it has a neighbor $j' \in I_t$ for which $\bar{L}^t_{j'} < 1$. Otherwise, we say that j is inactive at time t. We say that an edge $e = (j_l, j_r) \in E_t$, where $j_l \in I_t, j_r \in J_t$ is *active* at time t if the vertex j_r is active. Let A_t denote the set of active edges in E_t. Let $e = (j_l, j_r)$ be an edge in E_t. By definition, there is a path from j_l to j_r in G_t – we fix such a path P_e. As before, let C_j denote the completion time of job j. The dual variables are defined as follows:

- For each job j and time t, we define quantities $\alpha_{j,t}$. The dual variable α_j would be equal to $\sum_{t \geq 0} \alpha_{j,t}$. Fix a job j. If $t \notin [r_j, C_j]$ we set $\alpha_{j,t}$ to 0. Now, suppose $j \in J_t$. Consider the job j as a vertex in J_t (i.e., right side) in the bipartite graph H_t. We set $\alpha_{j,t}$ to w_j if j is active at time t, otherwise it is inactive.
- For each time t, we set β to $1/2m \cdot w(U_t)$ (Recall that U_t is the set of unfinished jobs at time t).
- We now need to define $\gamma_{t,j' \to j}$, where $j' \prec j$. If j or j' does not belong to J_t, we set this variable to 0. So assume that $j, j' \in J_t$ (and so the edge (j', j) lies in G_t). We define

$$\gamma_{t,j' \to j} := \eta^t \cdot \sum_{e : e \in A_t, (j' \to j) \in P_e} \bar{z}^t_e.$$

  In other words, we consider all the active edges e in the graph H_t for which the corresponding path P_e contains (j', j). We add up the fractional assignment $\bar{z}^t_e$ for all such edges.

This completes the description of the dual variables.

We first show that the objective function for (DLP) is close to the weighted completion time incurred by the algorithm. The proof is deferred to the full version.

▷ Claim 2.8. The total weighted completion time of the jobs in $\mathcal{A}$ is at most $2(\sum_j \alpha_j - m \cdot \sum_t \beta_t) + \sum_j w_j \cdot (\mathsf{chain}_j + 2r_j)$.

We now argue about feasibility of the dual constraint (9). Consider a job j and time $t \geq r_j$. Since $\alpha_{j,s} \leq w_j$ for all time s, $\sum_{s \leq t} \alpha_{j,s} \leq w_j \cdot t$. Therefore, it suffices to show:

$$\sum_{s \geq t} \alpha_{j,s} + \sum_{s \geq t} \left(\gamma^{\text{out}}_{s,j} - \gamma^{\text{in}}_{s,j} \right) \leq p_j \cdot \beta_t \tag{15}$$

Let $t^\star_j$ be the first time t when the job j appears in the set I_t. This would also be the first time when the algorithm starts processing j because a job that enters I_t does not leave I_t before completion.

▷ Claim 2.9. For any time s lying in the range $[r_j, t^\star_j)$, $\alpha_{j,s} + \gamma^{\text{out}}_{s,j} - \gamma^{\text{in}}_{s,j} = 0$.

Proof. Fix such a time s. Note that $j \notin I_s$. Thus j appears as a vertex on the right side in the bipartite graph H_s, but does not appear on the left side. Let e be in active edge in H_s such that the corresponding path P_e contains j as an internal vertex. Then $\bar{z}^s_e$ gets counted in both $\gamma^{\text{out}}_{s,j}$ and $\gamma^{\text{in}}_{s,j}$. There cannot be such a path P_e which starts with j, because then j will need to be on the left side of the bipartite graph. There could be paths P_e which end with j – these will correspond to active edges e incident with j in the graph H_t (this happens only if j itself is active). Let $\Gamma(j)$ denote the edges incident with j. We have shown that

$$\gamma^{\text{out}}_{s,j} - \gamma^{\text{in}}_{s,j} = -\eta^t \cdot \sum_{e \in \Gamma(j) \cap A_s} \tilde{z}^s_e. \tag{16}$$

If j is not active, the RHS is 0, and so is $\alpha_{j,s}$. So we are done. Therefore, assume that j is active. Now, $A(s)$ contains all the edges incident with j, and so, the RHS is same as $-\eta^t \cdot \bar{R}^t_j$. But then, Corollary 2.6 implies that $-\eta^t \cdot \bar{R}^t_j = -w_j$. Since $\alpha_{j,s} = w_j$, we are done again. ◁

Coming back to inequality (15), we can assume that $t \geq t^\star_j$. To see this, suppose $t < t^\star_j$. Then by Claim 2.9 the LHS of this constraint is same as

$$\sum_{s \geq t^\star_j} \alpha_{j,s} + \sum_{s \geq t^\star_j} \left(\gamma^{\text{out}}_{s,j} - \gamma^{\text{in}}_{s,j} \right).$$

Since $\beta_t \geq \beta_{t^\star_j}$ (the set of unfinished jobs can only diminish as time goes on), (15) for time t follows from the corresponding statement for time $t^\star_j$. Therefore, we assume that $t \geq t^\star_j$. We can also assume that $t \leq C_j$, otherwise the LHS of this constraint is 0.

▷ Claim 2.10. Let $s \in [t^\star_j, C_j]$ be such that j is inactive at time s. Then $\alpha_{j,s} + \gamma^{\text{out}}_{s,j} - \gamma^{\text{in}}_{s,j} \leq \eta^s \cdot \bar{L}^s_j$.

Proof. We know that $\alpha_{j,s} = 0$. As in the proof of Claim 2.9, we only need to worry about those active edges e in H_s for which P_e either ends at j or begins with j. Since any edge incident with j as a vertex on the right side is inactive, we get (let $\Gamma(j)$ denote the edges incident with j, where we consider j on the left side)

$$\alpha_{j,s} + \gamma^{\text{out}}_{s,j} - \gamma^{\text{in}}_{s,j} = \eta^s \cdot \sum_{e \in \Gamma(j) \cap A(s)} \bar{z}^s_e \leq \eta^s \cdot \bar{L}^s_j,$$

because $\eta^s \geq 0$ and $\bar{L}^s_j = \sum_{e \in \Gamma(j)} \bar{z}^s_e$. ◁

▷ Claim 2.11. Let $s \in [t^\star_j, C_j]$ be such that j is active at time s. Then $\alpha_{j,s} + \gamma^{\text{out}}_{s,j} - \gamma^{\text{in}}_{s,j} \leq \eta^s \cdot \bar{L}^s_j$.

Proof. The argument is very similar to the one in the previous claim. Since j is active, $\alpha_{j,s} = w_j$. As before we only need to worry about the active edges e for which P_e either ends or begins with j. Any edge which is incident with j on the right side (note that there will only one such edge – one the one joining j to its copy on the left side of H_t) is active. The following inequality now follows as in the proof of Claim 2.10:

$$\alpha_{j,s} + \gamma^{\text{out}}_{s,j} - \gamma^{\text{in}}_{s,j} \leq w_j + \eta^s \cdot \bar{L}^s_j - \eta^s \cdot \bar{R}^s_j.$$

The result now follows from Corollary 2.6. ◁

We are now ready to show that (15) holds. The above two claims show that the LHS of (15) is at most $\sum_{s=t}^{C_j} \eta^s \cdot \bar{L}^s_j$. Note that for any such time s, the rate assigned to j is $\bar{L}^j_s$, and so, we perform $2 \cdot \bar{L}^j_s$ amount of processing on j during this time slot. It follows that $\sum_{s=t}^{C_j} \bar{L}^s_j \leq p_j/2$. Now Claim 2.7 shows that $\eta^s \leq w(U_s)/m \leq w(U_t)/m$, and so we get

$$\sum_{s=t}^{C_j} \eta^s \cdot \bar{L}^s_j \leq \frac{p_j \cdot w(U_t)}{2m} = p_j \cdot \beta_t.$$

This shows that (15) is satisfied. We can now prove our algorithm is constant competitive.

▶ **Theorem 2.12.** *The algorithm $\mathcal{B}$ is 10-competitive.*

Proof. We first argue about $\mathcal{A}$. We have shown that the dual variables are feasible to (DLP), and so, Claim 2.8 shows that the total completion time of $\mathcal{A}$ is at most $2opt + \sum_j w_j(\mathsf{chain}_j + 2r_j)$, where opt denotes the optimal off-line objective value. Clearly, $opt \geq \sum_j w_j \cdot r_j$ and $opt \geq \sum_j w_j \cdot \mathsf{chain}_j$. This implies that $\mathcal{A}$ is 5-competitive. While going from $\mathcal{A}$ to $\mathcal{B}$ the completion time of each job doubles. ◀

3 Minimizing Weighted Flow Time

We now consider the setting of minimizing the total weighted flow time, again in the non-clairvoyant setting. The setting is almost the same as in the completion-time case: the major change is that all jobs which depend on each other (i.e., belong to the same DAG in the "collection of DAGs view" have the same release date). In the full version we show that if related jobs can be released over time then no competitive online algorithms are possible.

As before, let J_t denote the jobs which are *waiting* at time t, i.e., which have been released but not yet finished, and let G_t be the union of all the DAGs induced by the jobs in J_t. Again, let I_t denote the *minimal* set of jobs in J_t, i.e., which do not have a predecessor in G_t and hence can be scheduled.

▶ **Theorem 3.1.** *There exists an $O(1/\varepsilon)$-approximation algorithm for non-clairvoyant DAG scheduling to minimize the weighted flow time on m parallel machines, when there is a speedup of $2 + \varepsilon$.*

The rest of this section gives the proof of Theorem 3.1. The algorithm remains unchanged from §2 (we do not need the algorithm $\mathcal{B}$ now): we write the convex program (CP) as before, which assign rates $\bar{L}^t_j$ to each job $j \in I_t$. The analysis again proceeds by writing a linear

programming relaxation, and showing a feasible dual solution. The LP is almost the same as (LP), just the objective is now (with changes in dashed box):

$$\sum_{j,t} w_j \cdot \frac{\boxed{(t - r_j)} \cdot x_{j,t}}{p_j}.$$

Hence, the dual is also almost the same as (DLP): the new dual constraint requires that for every job j and time $t \geq r_j$:

$$\alpha_j + \sum_{s \geq t} \left(\gamma^{\text{out}}_{s,j} - \gamma^{\text{in}}_{s,j}\right) \leq \beta_t \cdot p_j + w_j \boxed{(t - r_j)}. \tag{17}$$

3.1 Defining the Dual Variables

In order to set the dual variables, define a total order $\prec$ on the jobs as follows: First arrange the DAGs in order of release dates, breaking ties arbitrarily. Let this order be $D_1, D_2, \ldots, D_\ell$. All jobs in D_i appear before those in D_{i+1} in the order $\prec$. Now for each dag D_i, arrange its jobs in the order they complete processing by our algorithm. Note that this order is consistent with the partial order given by the DAG. This also ensures that at any time t, the set of waiting jobs in any DAG D_i form a suffix in this total order (restricted to D_i).

For a time t and $j \in J_t$, let $\mathbf{I}[j \in J^{\text{act}}_t]$ denote the indicator variable which is 1 exactly if j is active at time t. The dual variables are defined as follows:

- For a job $j \in J_t$, we set $\alpha_j := \sum_{t=r_j}^{C_j} \alpha_{j,t}$, where the quantity $\alpha_{j,t}$ as defined as:

$$\alpha_{j,t} := \frac{1}{m}\Big[w_j \cdot \mathbf{I}[j \in J^{\text{act}}_t] \cdot \Big(\sum_{j' \in J_t : j' \preceq j} \bar{R}^t_{j'}\Big) + \bar{R}^t_j \cdot \Big(\sum_{j' \in J^{\text{act}}_t : j' \prec j} w_{j'}\Big)\Big].$$

- The variable $\beta_t := \frac{w(J_t)}{(1+\varepsilon)m}$. Recall that the machines are allowed $2(1+\varepsilon)$-speedup.
- The definition of the γ variables changes as follows. Let $(j' \to j)$ be an edge in the DAG G_t. Earlier we had considered paths P_e containing $(j' \to j)$ only for the active edges e. But now we include *all* edges. Moreover, we replace the multiplier η^t by η^t_j, where $\eta^t_j := \frac{1}{m} \cdot \Big(\sum_{j' \in J_t : j' \preceq j} w_{j'}\Big)$. In other words, we define

$$\gamma_{t,j' \to j} := \eta^t_j \cdot \sum_{e : e \in H_t, (j' \to j) \in P_e} \bar{z}^t_e.$$

In the following sections, we show that these dual settings are enough to "pay for" the flow time of our solution (i.e., have large objective function value), and also give a feasible lower bound (i.e., are feasible for the dual linear program).

3.2 The Dual Objective Function

We first show that $\sum_j \alpha_j - m \sum_t \beta_t$ is close to the total weighted flow-time of the jobs. The quantity chain_j is defined as before. Notice that chain_j is still a lower bound on the flow-time of job j in the optimal schedule because all jobs of a DAG are simultaneously released. The following claim, whose result is deferred to the full version, shows that the dual objective value is close to the weighted flow time of the algorithm.

▷ Claim 3.2. The total weighted flow-time is at most $\frac{2}{\varepsilon}\Big(\sum_j \alpha_j - m \sum_t \beta_t + \sum_j w_j \cdot \mathsf{chain}_j\Big)$.

3.3 Checking Dual Feasibility

Now we need to check the feasibility of the dual constraint (17). In fact, we will show the following weaker version of that constraint:

$$\alpha_j + \left\lceil \frac{2}{} \right\rceil \sum_{s \geq t} \left(\gamma^{\text{out}}_{s,j} - \gamma^{\text{in}}_{s,j}\right) \leq \beta_t \cdot p_j + \left\lceil \frac{2}{} \right\rceil w_j(t - r_j). \tag{18}$$

This suffices to within another factor of 2: indeed, scaling down the α and β variables by another factor of 2 then gives dual feasibility, and loses only another factor of 2 in the objective function. We begin by bounding $\alpha_{j,s}$ in two different ways.

▶ **Lemma 3.3.** *For any time $s \geq r_j$, we have $\alpha_{j,s} \leq 2w_j$.*

Proof. Consider the second term in the definition of $\alpha_{j,s}$. This term contains $\sum_{j' \in J^{\text{act}}_s : j' \prec j} w_{j'}$. By Corollary 2.6, for any $j' \in J^{\text{act}}_s$ we have $w_{j'} = \bar{R}^s_{j'} \cdot \eta^s$. Therefore,

$$\sum_{j' \in J^{\text{act}}_s : j' \prec j} w_{j'} \quad \leq \quad \eta^s \cdot \sum_{j' \in J^{\text{act}}_s : j' \prec j} \bar{R}^s_{j'} \quad \leq \quad \eta^s \cdot \sum_{j' \in J_s} \bar{R}^s_{j'}.$$

Now we can bound $\alpha_{j,s}$ by dropping the indicator on the first term to get

$$\frac{1}{m} \cdot \Big[\Big(w_j \cdot \sum_{j' \in J_s : j' \preceq j} \bar{R}^s_{j'} \Big) + \bar{R}^s_j \cdot \Big(\eta^s \cdot \sum_{j' \in J^{\text{act}}_s : j' \prec j} \bar{R}^s_{j'} \Big) \Big] \;\leq\; \frac{1}{m} w_j \Big[\sum_{j' \in J_s} \bar{R}^s_{j'} + \sum_{j' \in J_s} \bar{R}^s_{j'} \Big],$$

the last inequality using Claim 2.3. Simplifying, $\alpha_{j,s} \leq \frac{2}{m} \cdot w_j \cdot \sum_{j'' \in I_s} \bar{L}^s_{j''} = 2w_j$. ◀

Here is a slightly different upper bound on $\alpha_{j,s}$.

▶ **Lemma 3.4.** *For any time $s \geq r_j$, we have $\alpha_{j,s} \leq 2\eta^s_j \cdot \bar{R}^s_j$.*

Proof. The second term in the definition of $\alpha_{j,s}$ is at most $\eta^s_j \cdot \bar{R}^s_j$, directly using the definition of η^s_j. For the first term, assume j is active at time s, otherwise this term is 0. Now Corollary 2.6 shows that $w_j = \eta^s \cdot \bar{R}^s_j$, so the first term can be bounded as follows:

$$\frac{w_j}{m} \cdot \sum_{j' \in J_s : j' \preceq j} \bar{R}^s_{j'} \quad = \quad \frac{\bar{R}^s_j \cdot \eta^s}{m} \cdot \sum_{j' \in J_s : j' \preceq j} \bar{R}^s_{j'} \quad \overset{\text{(Claim 2.3)}}{\leq} \quad \frac{\bar{R}^s_j}{m} \cdot \sum_{j' \in J_s : j' \preceq j} w_{j'} \quad = \quad \bar{R}^s_j \cdot \eta^s_j,$$

which completes the proof. ◀

To prove (18), we write $\alpha_j = \sum_{s=r_j}^{t-1} \alpha_{j,s} + \sum_{s \geq t} \alpha_{j,s}$, and use Lemma 3.3 to cancel the first summation with the term $2w_j(t - r_j)$. Hence, it remains to prove

$$\sum_{s \geq t} \alpha_{j,s} + 2 \sum_{s \geq t} \left(\gamma^{\text{out}}_{s,j} - \gamma^{\text{in}}_{s,j}\right) \leq \beta_t \cdot p_j. \tag{19}$$

Let $t^\star_j$ be the time at which the algorithm starts processing j. We first argue why we can ignore times $s < t^\star_j$ on the LHS of (19).

▷ Claim 3.5. Let s be a time satisfying $r_j \leq s < t^\star_j$. Then $\alpha_{j,s} + 2(\gamma^{\text{out}}_{s,j} - \gamma^{\text{in}}_{s,j}) \leq 0$.

Proof. While computing $\gamma^{\text{out}}_{s,j} - \gamma^{\text{in}}_{s,j}$, we only need to consider paths P_e for edges e in H_s which have j as end-point. Since j does not appear on the left side of H_s, this quantity is equal to $-\eta^s_j \cdot \bar{R}^s_j$. The result now follows from Lemma 3.4. ◁

So using Claim 3.5 in (19), it suffices to show

$$\sum_{s \geq \max\{t, t_j^\star\}} \alpha_{j,s} + 2 \sum_{s \geq \max\{t, t_j^\star\}} \left(\gamma_{s,j}^{\text{out}} - \gamma_{s,j}^{\text{in}}\right) \leq \beta_t \cdot p_j. \tag{20}$$

Note that we still have β_t on the right hand side, even though the summation on the left is over times $s \geq \max\{t, t_j^\star\}$. The proof of the following claim is deferred to the full version.

▷ Claim 3.6. Let s be a time satisfying $s \geq \max\{t, t_j^\star\}$. Then $\alpha_{j,s} + 2(\gamma_{s,j}^{\text{out}} - \gamma_{s,j}^{\text{in}}) \leq 2(1+\varepsilon)\beta_t \cdot \bar{L}_j^s$.

Hence, the left-hand side of (20) is at most $2(1+\varepsilon)\beta_t \cdot \sum_{s \geq \max\{t, t_j^\star\}} \bar{L}_j^s$. However, since job j is assigned a rate of $\bar{L}_j^s$ and the machines run at speed $2(1+\varepsilon)$, we get that this expression is at most $p_j \cdot \beta_t$, which is the right-hand side of (20). This proves the feasibility of the dual constraint (18).

Proof of Theorem 3.1. In the preceding §3.3 we proved that the variables $\alpha_j/2$, $\beta_t/2$ and $\gamma_{t,j' \to j}$ satisfy the dual constraint for the flow-time relaxation. Since $\sum_j (\alpha_j/2) - m \sum_t (\beta_t/2)$ is a feasible dual, it gives a lower bound on the cost of the optimal solution. Moreover, $\sum_j w_j \cdot \mathsf{chain}_j$ is another lower bound on the cost of the optimal schedule. Now using the bound on the weighted flow-time of our schedule given by Claim 3.2, this shows that we have an $O(1/\varepsilon)$-approximation with $2(1+\varepsilon)$-speedup. ◀

In the full version we show how to use a slightly different scheduling policy that prioritizes the last arriving jobs to reduce the speedup to $(1+\varepsilon)$.

References

1. Kunal Agrawal, Jing Li, Kefu Lu, and Benjamin Moseley. Scheduling Parallel DAG Jobs Online to Minimize Average Flow Time. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 176–189, 2016. doi:10.1137/1.9781611974331.ch14.
2. S. Anand, Naveen Garg, and Amit Kumar. Resource augmentation for weighted flow-time explained by dual fitting. In *SODA'12*, pages 1228–1241. ACM, New York, 2012.
3. Abbas Bazzi and Ashkan Norouzi-Fard. Towards Tight Lower Bounds for Scheduling Problems. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 118–129, 2015. doi:10.1007/978-3-662-48350-3_11.
4. Fabián A. Chudak and David B. Shmoys. Approximation Algorithms for Precedence-Constrained Scheduling Problems on Parallel Machines that Run at Different Speeds. *J. Algorithms*, 30(2):323–343, 1999. doi:10.1006/jagm.1998.0987.
5. Jeff Edmonds. Scheduling in the Dark. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA*, pages 179–188, 1999. doi:10.1145/301250.301299.
6. Jeff Edmonds, Donald D. Chinn, Tim Brecht, and Xiaotie Deng. Non-clairvoyant Multiprocessor Scheduling of Jobs with Changing Execution Characteristics (Extended Abstract). In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 120–129, 1997. doi:10.1145/258533.258565.
7. Jeff Edmonds and Kirk Pruhs. Scalably scheduling processes with arbitrary speedup curves. *ACM Trans. Algorithms*, 8(3):Art. 28, 10, 2012. doi:10.1145/2229163.2229172.
8. R. L. Graham. Bounds for Certain Multiprocessing Anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966. doi:10.1002/j.1538-7305.1966.tb01709.x.

9 Robert Grandl, Srikanth Kandula, Sriram Rao, Aditya Akella, and Janardhan Kulkarni. GRAPHENE: packing and dependency-aware scheduling for data-parallel clusters. In *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016.*, pages 81–97, 2016. URL: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/grandl_graphene.

10 Anupam Gupta, Ravishankar Krishnaswamy, and Kirk Pruhs. Online Primal-Dual for Non-linear Optimization with Applications to Speed Scaling. In *Approximation and Online Algorithms - 10th International Workshop, WAOA 2012, Ljubljana, Slovenia, September 13-14, 2012, Revised Selected Papers*, pages 173–186, 2012. doi:10.1007/978-3-642-38016-7_15.

11 Leslie A. Hall, Andreas S. Schulz, David B. Shmoys, and Joel Wein. Scheduling to minimize average completion time: off-line and on-line approximation algorithms. *Math. Oper. Res.*, 22(3):513–544, 1997. doi:10.1287/moor.22.3.513.

12 Sungjin Im, Janardhan Kulkarni, and Kamesh Munagala. Competitive Algorithms from Competitive Equilibria: Non-Clairvoyant Scheduling under Polyhedral Constraints. *J. ACM*, 65(1):3:1–3:33, 2018. doi:10.1145/3136754.

13 Sungjin Im, Janardhan Kulkarni, Kamesh Munagala, and Kirk Pruhs. SelfishMigrate: A Scalable Algorithm for Non-clairvoyantly Scheduling Heterogeneous Processors. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 531–540, 2014.

14 Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *J. ACM*, 47(4):617–643, 2000.

15 Janardhan Kulkarni and Shi Li. Flow-time Optimization for Concurrent Open-Shop and Precedence Constrained Scheduling Models. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20-22, 2018 - Princeton, NJ, USA*, pages 16:1–16:21, 2018. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.16.

16 Shi Li. Scheduling to minimize total weighted completion time via time-indexed linear programming relaxations. In *58th Annual IEEE Symposium on Foundations of Computer Science—FOCS 2017*, pages 283–294. IEEE Computer Soc., Los Alamitos, CA, 2017.

17 Rajeev Motwani, Steven Phillips, and Eric Torng. Nonclairvoyant scheduling. *Theorertical Computer Science*, 130(1):17–47, 1994.

18 Alix Munier, Maurice Queyranne, and Andreas S. Schulz. Approximation bounds for a general class of precedence constrained parallel machine scheduling problems. In *Integer programming and combinatorial optimization (Houston, TX, 1998)*, volume 1412 of *Lecture Notes in Comput. Sci.*, pages 367–382. Springer, Berlin, 1998. doi:10.1007/3-540-69346-7_28.

19 John F. Nash. The Bargaining Problem. *Econometrica*, 18(2):155–162, 1950. URL: http://www.jstor.org/stable/1907266.

20 Julien Robert and Nicolas Schabanel. Non-clairvoyant scheduling with precedence constraints. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 491–500, 2008. URL: http://dl.acm.org/citation.cfm?id=1347082.1347136.

A Composition Theorem for Randomized Query Complexity via Max-Conflict Complexity*

Dmitry Gavinsky
Institute of Mathematics, Czech Academy of Sciences, 115 67 Žitna 25, Praha 1, Czech Republic

Troy Lee
Centre for Quantum Software and Information, Faculty of Engineering and Information Technology, University of Technology Sydney, Australia
troyjlee@gmail.com

Miklos Santha
CNRS, IRIF, Université de Paris, 75205 Paris, France
Centre for Quantum Technologies, National University of Singapore, Singapore 117543
MajuLab, UMI 3654, Singapore
santha@irif.fr

Swagato Sanyal
Indian Institute of Technology Kharagpur, India
swagato@cse.iitkgp.ac.in

Abstract

For any relation $f \subseteq \{0,1\}^n \times S$ and any partial Boolean function $g : \{0,1\}^m \to \{0,1,*\}$, we show that

$$\mathsf{R}_{1/3}(f \circ g^n) \in \Omega(\mathsf{R}_{4/9}(f) \cdot \sqrt{\mathsf{R}_{1/3}(g)}),$$

where $\mathsf{R}_\epsilon(\cdot)$ stands for the *bounded-error randomized query complexity* with error at most ϵ, and $f \circ g^n \subseteq (\{0,1\}^m)^n \times S$ denotes the composition of f with n instances of g.

The new composition theorem is *optimal*, at least, for the general case of relational problems: A relation f_0 and a partial Boolean function g_0 are constructed, such that $\mathsf{R}_{4/9}(f_0) \in \Theta\left(\sqrt{n}\right)$, $\mathsf{R}_{1/3}(g_0) \in \Theta(n)$ and $\mathsf{R}_{1/3}(f_0 \circ g_0^n) \in \Theta(n)$.

The theorem is proved via introducing a new complexity measure, *max-conflict complexity*, denoted by $\bar{\chi}(\cdot)$. Its investigation shows that $\bar{\chi}(g) \in \Omega(\sqrt{\mathsf{R}_{1/3}(g)})$ for any partial Boolean function g and $\mathsf{R}_{1/3}(f \circ g^n) \in \Omega(\mathsf{R}_{4/9}(f) \cdot \bar{\chi}(g))$ for any relation f, which readily implies the composition statement. It is further shown that $\bar{\chi}(g)$ is always at least as large as the *sabotage complexity* of g.

2012 ACM Subject Classification Theory of computation → Oracles and decision trees

Keywords and phrases query complexity, lower bounds

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.64

Category Track A: Algorithms, Complexity and Games

Related Version https://arxiv.org/abs/1811.10752 [6]

Funding Part of this work was conducted while T.L. and S.S. were at the Nanyang Technological University and the Centre for Quantum Technologies, supported by the Singapore National Research Foundation under NRF RF Award No. NRF-NRFF2013-13. This work was additionally supported by the Singapore National Research Foundation, the Prime Minister's Office, Singapore and the Ministry of Education, Singapore under the Research Centres of Excellence programme under research grant R 710-000-012-135. This research was supported in part by the QuantERA ERA-NET Cofund project QuantAlgo. D.G. is partially funded by the grant 19-27871X of GA ČR.

* This paper is a merger of [5] and [12], together with some new results.

Acknowledgements We thank Rahul Jain for useful discussions. We thank Srijita Kundu and Jevgēnijs Vihrovs for their helpful comments on the manuscript. We thank Yuval Filmus for suggesting to look at the min-max version of conflict complexity, which led to the development of max-conflict complexity. We thank the anonymous reviewers for their helpful comments.

1 Introduction

For a relational problem $f \subseteq \{0,1\}^n \times S$ and a partial Boolean function $g : \{0,1\}^m \to \{0,1,*\}$, their *composition* $f \circ g^n \subseteq (\{0,1\}^m)^n \times S$ is defined as

$$f \circ g^n(x) = \begin{cases} S & \text{if } * \in \{g(x_i) | i \in [n]\}; \\ f(g(x_1), \ldots, g(x_n)) & \text{otherwise.} \end{cases}$$

Relating the complexity of $f \circ g^n$ to the complexities of f and g is a natural research problem.

A *query algorithm* for computing h is allowed to query *individual bits* of the input x, with the goal of outputting $h(x)$ (or an element of $h(x)$ if the problem is a relational one). The *query complexity of an algorithm* is the maximum possible number of queries that it makes.

Query algorithms can be *deterministic*, *randomized* or *quantum*, where the latter two classes allow for (bounded) errors. The corresponding *query complexity of a function* – denoted, respectively, by $\mathsf{D}(h)$, $\mathsf{R}(h)$ or $\mathsf{Q}(h)$ – is the minimal query complexity of an algorithm that belongs to the corresponding class and computes h with error $1/3$[1]. Section 2 contains formal definitions of various query complexity measures.

It is easy to see that $\mathsf{D}(f \circ g^n) \leq \mathsf{D}(f) \cdot \mathsf{D}(g)$. [2] For the cases of randomized and quantum query complexity the argument is slightly more subtle, though very similar conceptually; in particular, both $\mathsf{R}(f \circ g^n) \in O(\mathsf{R}(f) \cdot \mathsf{R}(g) \cdot \log n)$ and $\mathsf{Q}(f \circ g^n) \in O(\mathsf{Q}(f) \cdot \mathsf{Q}(g))$ hold. [3]

Showing a strong *lower bounds* on the query complexity of $f \circ g^n$ (preferably, matching the trivial upper bound) is often more interesting, the corresponding statements are sometimes called *composition theorems*. Such results can lead to further theoretical developments (e.g., separating complexity measures, as well as different classes in structural complexity).

For deterministic query complexity it has been shown [10, 13] that

$$\mathsf{D}(f \circ g^n) = \mathsf{D}(f) \cdot \mathsf{D}(g),$$

which means that *the trivial query algorithm for $f \circ g^n$ described above is optimal*. Similarly, for bounded-error quantum query complexity it has been shown [8, 11] that

$$\mathsf{Q}(f \circ g^n) \in \Theta(\mathsf{Q}(f) \cdot \mathsf{Q}(g)).$$

Prior to this work, the randomized query complexity of composition has remained an open problem. We partially solve it for the most general case of composition: namely, letting f be a *relational problem* and g be a *partial Boolean function*. [4]

---

[1] In general, $\mathsf{R}_\epsilon(h)$ (resp. $\mathsf{Q}_\epsilon(h)$) stands for the randomized (resp. quantum) query complexity with respect to error ϵ.

[2] To compute $f \circ g^n$, one can simulate an optimal query algorithm for f, serving every query of this algorithm by running an optimal query algorithm for g.

[3] The multiplicative factor of $\log n$ in the case of $\mathsf{R}(h)$ is due to the need to reduce the error in computing each instance of g to $O(1/n)$; in the quantum case this can be handled in a more elegant, "lossless" way.

[4] Letting g be a relation seems to result in a rather awkward definition of $f \circ g^n$. Letting g be a non-Boolean promise function doesn't seem to lead to any interesting development (the original version of this work [5] has demonstrated the same composition result for an arbitrary partial g; switching to the Boolean case in the current version has allowed somewhat clearer presentation).

▶ **Theorem 1.** *For any relation* $f \subseteq \{0,1\}^n \times S$ *and any partial Boolean function* $g : \{0,1\}^m \to \{0,1,*\}$,

$$\mathsf{R}_{1/3}(f \circ g^n) \in \Omega\Big(\mathsf{R}_{4/9}(f) \cdot \sqrt{\mathsf{R}_{1/3}(g)}\Big).$$

Note that the above lower bound *does not match the trivial upper bound*, so its optimality has to be addressed separately. That is done via constructing an example where the above bound is tight: in other words, while some *incomparable* lower bounds on $\mathsf{R}_{1/3}(f \circ g^n)$ are conceivable, the statement of Theorem 1 is *a strongest possible in general*. [5]

▶ **Theorem 2.** *There exists a relation* $f_0 \subseteq \{0,1\}^n \times \{0,1\}^n$ *and a partial Boolean function* $g_0 : \{0,1\}^n \to \{0,1,*\}$, *such that*

$$\mathsf{R}_{4/9}(f_0) \in \Theta\big(\sqrt{n}\big),\ \mathsf{R}_{1/3}(g_0) \in \Theta(n) \text{ and } \mathsf{R}_{1/3}(f_0 \circ g_0^n) \in \Theta(n).$$

Our approach

We introduce a new complexity measure of Boolean functions, the *max-conflict complexity*, denoted by $\bar{\chi}(g)$. We show that $\bar{\chi}(g)$ is a quadratically tight lower bound on randomized query complexity of a (partial) function g.

▶ **Theorem 3.** *For any partial Boolean function* $g : \{0,1\}^m \to \{0,1,*\}$,

$$\bar{\chi}(g) \in \Omega\Big(\sqrt{\mathsf{R}_{1/3}(g)}\Big).$$

The main technical ingredient of this work is the following composition statement for the max-conflict complexity.

▶ **Theorem 4.** *For any relation* $f \subseteq \{0,1\}^n \times S$ *and any partial Boolean function* $g : \{0,1\}^m \to \{0,1,*\}$,

$$\mathsf{R}_{1/3}(f \circ g^n) \in \Omega\big(\mathsf{R}_{4/9}(f) \cdot \bar{\chi}(g)\big).$$

Combining Theorem 3 with Theorem 4 implies Theorem 1.

Previous work

In the special case of f being a partial function and g being a total one, a significant progress has been made by Ben-David and Kothari [4], who showed recently that

$$R_{1/3}(f \circ g^n) \in \Omega\left(R_{1/3}(f) \cdot \sqrt{\frac{R_0(g)}{\log R_0(g)}}\right). \tag{1}$$

To prove the above statement, the authors have introduced and investigated a new complexity measure of Boolean functions, *sabotage complexity*, denoted by $\mathrm{RS}(g)$. This notion has a very natural definition and is of independent interest. In this work we show that max-conflict complexity is always lower-bounded by the sabotage complexity of the same function.

▶ **Theorem 5.** *For any partial Boolean function* $g : \{0,1\}^m \to \{0,1,*\}$,

$$\bar{\chi}(g) \geq \mathrm{RS}(g).$$

Theorem 5 along with Theorem 4 imply (1).

[5] The following construction also witnesses the possibility of $\mathsf{R}(f \circ g^n) \in O(\mathsf{R}(g))$ when $\mathsf{R}(f) \in \Omega\big(\sqrt{n}\big)$ – in other words, *it is not, in general, true, that composition with a "hard" relation makes a Boolean function harder for randomized query algorithms.*

1.1 Proof Technique

At a high level, the proof of Theorem 4 follows the structure of the proof by Anshu et al. [2] and Ben-David and Kothari [4]. We show that for every probability distribution η over the input space $\{0,1\}^n$ of f, there exists a deterministic query algorithm $\mathcal{A}$ that makes $O(\mathsf{R}_{1/3}(f \circ g^n)/\sqrt{\mathsf{R}_{1/3}(g)})$ queries in the worst case, and computes f with high probability, $\Pr_{z\sim\eta}[(z, \mathcal{A}(z)) \in f] \geq 5/9$. By the minimax principle (Fact 4) this implies Theorem 4.

We do this by using a query algorithm for $f \circ g^n$ to construct a query algorithm for f. We define a sampling procedure that for any $z \in \{0,1\}^n$ samples $x = (x_1, \ldots, x_n)$ such that $(z, s) \in f$ if and only if $(x, s) \in f \circ g^n$. This procedure is defined in terms of $\mathcal{Q}$, which is a probability distribution over pairs of distributions (μ_0, μ_1), where μ_0 is supported on $g^{-1}(0)$ and μ_1 is supported on $g^{-1}(1)$. We define a distribution γ_η over $(\{0,1\}^m)^n$ in terms of this sampling process as follows:

1. Sample $z = (z_1, \ldots, z_n)$ from $\{0,1\}^n$ according to η.
2. Independently sample $(\mu_0^{(i)}, \mu_1^{(i)})$ from $\mathcal{Q}$ for $i = 1, \ldots, n$.
3. Sample $x_i = (x_i^{(1)}, \ldots, x_i^{(m)})$ according to $\mu_{z_i}^{(i)}$ for $i = 1, \ldots, n$. Return $x = (x_1, \ldots, x_n)$.

Notice that steps (1) and (2) are independent and the order in which they are performed does not matter. For future reference, for a fixed z let $\gamma_z(\mathcal{Q})$ be the probability distribution defined by the last two steps.

Now γ_η is simply a probability distribution over $(\{0,1\}^m)^n$. Thus by the minimax principle (Fact 4 below), there is a deterministic query algorithm $\mathcal{A}'$ of worst-case complexity at most $\mathsf{R}_{1/3}(f \circ g^n)$ such that $\Pr_{x\sim\gamma_\eta}[(x, \mathcal{A}'(x)) \in f \circ g^n] \geq 2/3$. We first use $\mathcal{A}'$ to construct a randomized query algorithm T for f with bounded *expected* query complexity and error at most $1/3$. The final algorithm $\mathcal{A}$ will be a truncation of T which has bounded worst-case complexity and error at most $4/9$.

On input z, the algorithm T seeks to sample a string x from $\gamma_z(\mathcal{Q})$, and run $\mathcal{A}'$ on x. Put another way, $\gamma_z(\mathcal{Q})$ induces a probability distribution over the leaves of $\mathcal{A}'$, and the goal of T is to sample a leaf of $\mathcal{A}'$ according to this distribution. Since for each $s \in S$, $(x, s) \in f \circ g^n$ if and only if $(z, s) \in f$, and $\Pr_{x\sim\gamma_\eta}[(x, \mathcal{A}'(x)) \in f \circ g^n] \geq 2/3$, we have that $\Pr_{z\sim\eta}[(z, T(z)) \in f] \geq 2/3$. Thus T meets the accuracy requirement.

The catch, of course, is to specify how T samples from $\gamma_z(\mathcal{Q})$ *without making too many queries to* z. To sample x_i from $\mu_{z_i}^{(i)}$ seems to require knowledge of z_i, and thus T would have to query all of z.

To bypass this problem, we remember that $\mathcal{A}'$, being an efficient algorithm, will query only a few bits of x. This allows us to sample x bit by bit as and when they are queried by $\mathcal{A}'$. To see this more clearly, consider a run of T where the pairs of distributions $(\mu_0^{(1)}, \mu_1^{(1)}), \ldots, (\mu_0^{(n)}, \mu_1^{(n)})$ were chosen in step (2) of the sampling procedure. Suppose that T is trying to simulate $\mathcal{A}'$ at a vertex v where $x_i^{(j)}$ is queried. To respond to this query, T will sample $x_i^{(j)}$ from its marginal distribution according to $\mu_{z_i}^{(i)}$ conditioned on the event $x \in v$. Let the following be the marginal distributions of $x_i^{(j)}$ for the two possible values of z_i.

	$\Pr_{x_i\sim\mu_{z_i}^{(i)}}[x_i^{(j)} = 0 \mid x \in v]$	$\Pr_{x_i\sim\mu_{z_i}^{(i)}}[x_i^{(j)} = 1 \mid x \in v]$
$z_i = 0$	p_0	$1 - p_0$
$z_i = 1$	p_1	$1 - p_1$

Without loss of generality, assume that $p_0 \leq p_1$. T answers the query by the procedure BITSAMPLER given in Algorithm 1. Note that the bit returned by BITSAMPLER has the desired distribution. The step in which BITSAMPLER returns the bit depends on the value

Algorithm 1: BITSAMPLER (suppose $p_0 \leq p_1$).

```
1  Sample r ∼ [0,1] uniformly at random.
2  if r < p_0 then
3      return 0.
4
5  else if r > p_1 then
6      return 1.
7
8  else
9      query z_i.
10     if r ≤ p_{z_i} then
11         return 0.
12     else
13         return 1.
```

of r sampled in step 1. In particular, z_i is queried if and only if $r \in [p_0, p_1]$, and the bit is returned in step 11 or 13. Such a query to z_i contributes to the query complexity of T. Thus the probability that T makes a query when the underlying simulation of $\mathcal{A}'$ is at vertex v is $(p_1 - p_0)$. We refer to this quantity as $\Delta(v)$. It plays an important role in our analysis (in particular, in the proof of Theorem 6 that can be found in [6]).

Our sampling procedure and the tools we use to bound its cost is reminiscent of work of Barak et al. [3] in communication complexity. They look at a communication analog of our setting where two players are trying to sample a leaf in a communication protocol while communicating as little as possible.

1.1.1 Conflict complexity and max-conflict complexity

Bounding the query complexity of T naturally suggests the quantities that we define in this work: the conflict complexity $\chi(g)$ and the max-conflict complexity $\bar{\chi}(g)$ of a partial Boolean function g. A formal definition can be found in Section 4; here we give the high-level idea and motivation behind these quantities.

Forget about T for a moment and just consider a deterministic query algorithm $\mathcal{B}$ computing the partial function $g \subseteq \{0,1\}^m \times \{0,1\}$. Let μ_0, μ_1 be distributions with support on $g^{-1}(0), g^{-1}(1)$, respectively. For each vertex $v \in \mathcal{B}$ let $p_0(v)$ (respectively $p_1(v)$) be the probability that the answer to the query at v is 0 on input $x \sim \mu_0$ (respectively $x \sim \mu_1$), conditioned on x reaching v. Now we can imagine a process $\mathcal{P}(\mathcal{B}, \mu_0, \mu_1)$ that runs BITSAMPLER on the tree $\mathcal{B}$: $\mathcal{P}(\mathcal{B}, \mu_0, \mu_1)$ begins at the root, and at a vertex v in $\mathcal{B}$ it uniformly chooses a random real number $r \in [0,1]$. If $r < \min\{p_0(v), p_1(v)\}$ then the query is "answered" 0 and it moves to the left child. If $r > \max\{p_0(v), p_1(v)\}$ then the query is "answered" 1 and it moves to the right child. If $r \in [\min\{p_0(v), p_1(v)\}, \max\{p_0(v), p_1(v)\}]$ then the process halts. The conflict complexity $\chi(\mathcal{B}, (\mu_0, \mu_1))$ is the expected number of vertices this process visits before halting. The conflict complexity of g is defined to be

$$\chi(g) = \max_{(\mu_0,\mu_1)} \min_{T} \chi(T, (\mu_0, \mu_1)) \ ,$$

where the minimum is taken over trees T that compute g. For *max-conflict complexity* we enlarge the set over which we maximize. Let $\mathcal{Q}$ be a distribution over pairs of distributions (μ_0, μ_1), where $\mathrm{supp}(\mu_0) \subseteq g^{-1}(0), \mathrm{supp}(\mu_1) \subseteq g^{-1}(1)$ for each pair (μ_0, μ_1) in the support of $\mathcal{Q}$. Let $\chi(\mathcal{B}, \mathcal{Q}) = \mathsf{E}_{(\mu_0,\mu_1)\sim\mathcal{Q}}\,[\chi(\mathcal{B}, (\mu_0, \mu_1))]$. The max-conflict complexity $\bar{\chi}(g)$ is defined as

$$\bar{\chi}(g) = \max_{\mathcal{Q}} \min_{T} \chi(T, \mathcal{Q}) \ ,$$

where the minimum is taken over trees T that compute g. Clearly, the max-conflict complexity is at least as large as the conflict complexity.

To motivate the max-conflict complexity, note that the query complexity of T is the number of times step 9 in BITSAMPLER is executed, i.e. when the random number $r \in [p_0, p_1]$. In the definition of T we will choose $\mathcal{Q}$ to achieve the optimal value in the definition of $\bar{\chi}(g)$. Then intuitively one expects that for each i, T queries z_i only after $\mathcal{A}'$ makes about $\bar{\chi}(g)$ queries into x_i. By means of a direct sum theorem for max-conflict complexity we make this intuition rigorous and prove that the expected query complexity of T is at most $\mathsf{R}_{1/3}(f \circ g^n)/\bar{\chi}(g)$. We refer the reader to [6] for a formal proof.

1.1.2 $\bar{\chi}(g)$ and $\mathsf{R}(g)$

Note that applying Theorem 4 with the outer function $f(z) = z_1$ shows that $\mathsf{R}_{1/3}(g) \in \Omega(\bar{\chi}(g))$. We complete the proof of Theorem 1 by showing that max-conflict complexity is a quadratically tight lower bound on randomized query complexity, even for partial functions g. In fact, we show the stronger result that this is true even for the conflict complexity.

▶ **Theorem 6.** *For any partial Boolean function $g \subseteq \{0,1\}^m \times \{0,1\}$,*

$$\chi(g) \in \Omega\Big(\sqrt{\mathsf{R}_{1/3}(g)}\Big).$$

A proof of Theorem 6 can be found in [6]. At a high level, our proof is reminiscent of the result of [3] on compressing communication protocols in that both look at a random sampling process to navigate a tree, and relate the probability of this process needing to query or communicate at a node to the amount of information that is learned at the node.

To prove $\mathsf{R}(g) \in O(\chi(g)^2)$, we again resort to the minimax principle; we show that for each probability distribution μ over the valid inputs to g, there is an accurate and efficient distributional query algorithm for g. For $b \in \{0,1\}$, let μ_b be the distribution obtained by conditioning μ on the event $g(x) = b$. By the definition of $\chi(g)$, there is a query algorithm $\mathcal{B}$ such that the following is true: if its queries are served by BITSAMPLER, step 9 is executed within expected $\chi(\mathcal{B}, \mu_0, \mu_1) \leq \chi(g)$ queries. Note that at a vertex v which queries i, the probability that step 9 is executed is $\Delta(v) = |\Pr_{\mu_0}[x_i = 0 \mid x \text{ at } v] - \Pr_{\mu_1}[x_i = 0 \mid x \text{ at } v]|$. This roughly implies that for a typical vertex v of $\mathcal{B}$, $\Delta(v)$ is at least about $\frac{1}{\chi(g)}$. By a technical claim this implies that the query outcome at v carries about $\frac{1}{\chi(g)^2}$ bits of information about $g(x)$. Using the *chain rule of mutual information*, we can show that the mutual information between $g(x)$ and the outcomes of first $O(\chi(g))^2$ queries by $\mathcal{B}$ is $\Omega(1)$. This enables us to conclude that we can infer the value of $g(x)$ with success probability $1/2 + \Omega(1)$ from the transcript of $\mathcal{B}$ restricted to the first $O(\chi(g)^2)$ queries. The distributional algorithm of g for μ is simply the algorithm $\mathcal{B}$ terminated after $O(\chi(g)^2)$ queries.

1.1.3 $\bar{\chi}(g)$ and $\mathrm{RS}(g)$

To see why $\bar{\chi}(g) \geq \mathrm{RS}(g)$, we first give an alternative characterization of $\mathrm{RS}(g)$. For a deterministic tree T computing g and strings x, y such that $g(x) \neq g(y)$, let $\mathrm{sep}_T(x, y)$ be the depth of the node v in T such that x and y both reach v yet $x_{q(v)} \neq y_{q(v)}$, where $q(v)$

is the index queried at v. Let $\mathcal{T}$ be a zero-error randomized protocol for g, i.e. $\mathcal{T}$ is a probability distribution supported on deterministic trees that compute g. Then we have (for a proof see [6])

$$\mathrm{RS}(g) = \min_{\mathcal{T}} \max_{\substack{x,y \\ g(x)\neq g(y)}} \mathsf{E}_{T\sim\mathcal{T}}[\mathrm{sep}_T(x,y)] \ .$$

By von Neumann's minimax theorem [14], this is equal to

$$\mathrm{RS}(g) = \max_{p} \min_{T} \mathsf{E}_{(x,y)\sim p}[\mathrm{sep}_T(x,y)] \ .$$

Here, the max is taken over distributions p on pairs (x,y) where $g(x)\neq g(y)$, and the min is taken over deterministic trees T computing g.

We have seen that the definition of $\bar{\chi}(g)$ is

$$\bar{\chi}(g) = \max_{\mathcal{Q}} \min_{T} \mathsf{E}_{(\mu_0,\mu_1)\sim\mathcal{Q}} \ [\chi(T,(\mu_0,\mu_1))] \ ,$$

where $\mathcal{Q}$ is a distribution over pairs (μ_0,μ_1) and T is a deterministic tree computing g. When (μ_0,μ_1) are taken to be singleton distributions, i.e. μ_0 puts all its weight on a single x with $g(x)=0$, and μ_1 puts all its weight on a single y with $g(y)=1$, it can be shown that $\chi(T,(\mu_0,\mu_1)) = \mathrm{sep}_T(x,y)$ (see [6] for details). Thus $\bar{\chi}(g)$ is at least as large as the sabotage complexity of g as $\mathcal{Q}$ is allowed to be a distribution over general (μ_0,μ_1), not just singleton distributions.

2 Preliminaries

Let $g \subseteq \{0,1\}^m \times \{0,1\}$ be a partial Boolean function. For $b\in\{0,1\}$, $g^{-1}(b)$ is defined to be the set of strings x in $\{0,1\}^m$ for which $(x,b)\in g$ and $(x,\bar{b})\notin g$. We refer to $g^{-1}(0)\cup g^{-1}(1)$ as the set of valid inputs to g. We assume that for all strings $y\notin g^{-1}(0)\cup g^{-1}(1)$, both $(y,0)$ and $(y,1)$ are in g. For a string $x\in g^{-1}(0)\cup g^{-1}(1)$, $g(x)$ refers to the unique bit b such that $(x,b)\in g$. All the probability distributions μ over the domain of a partial Boolean function g in this paper are assumed to have support on $g^{-1}(0)\cup g^{-1}(1)$. Thus $g(x)$ is well-defined for any x in the support of μ.

Let S be any set. Let $h\subseteq\{0,1\}^k\times S$ be any relation. Consider query algorithms $\mathcal{A}$ that accept a string $x\in\{0,1\}^k$ as input, query various bits of x, and produce an element of S as output. We denote the output by $\mathcal{A}(x)$.

▶ **Definition 1** (Deterministic query complexity). *A deterministic query algorithm $\mathcal{A}$ is said to compute h if $(x,\mathcal{A}(x))\in h$ for all $x\in\{0,1\}^k$. The deterministic query complexity $\mathsf{D}(h)$ of h is the minimum over all deterministic query algorithms $\mathcal{A}$ computing h of the maximum number of queries made by $\mathcal{A}$ over $x\in\{0,1\}^k$.*

▶ **Definition 2** (Bounded-error randomized query complexity). *Let $\epsilon\in[0,1/2)$. We say that a randomized query algorithm $\mathcal{A}$ computes h with error ϵ if $\Pr[(x,\mathcal{A}(x))\in h]\geq 1-\epsilon$ for all $x\in\{0,1\}^k$. The bounded-error randomized query complexity $\mathsf{R}_\epsilon(h)$ of h is the minimum over all randomized query algorithms $\mathcal{A}$ computing h with error ϵ of the maximum number of queries made by $\mathcal{A}$ over all $x\in\{0,1\}^k$ and the internal randomness of $\mathcal{A}$.*

▶ **Definition 3** (Distributional query complexity). *Let μ a distribution on the input space $\{0,1\}^k$ of h, and $\epsilon\in[0,1/2)$. We say that a deterministic query algorithm $\mathcal{A}$ computes h with distributional error ϵ on μ if $\Pr_{x\sim\mu}[(x,\mathcal{A}(x))\in h]\geq 1-\epsilon$. The distributional query complexity $\mathsf{D}^\mu_\epsilon(h)$ of h is the minimum over deterministic algorithms $\mathcal{A}$ computing h with distributional error ϵ on μ of the maximum over $x\in\{0,1\}^k$ of the number of queries made by $\mathcal{A}$ on x.*

We will use the minimax principle in our proofs to go between distributional and randomized query complexity.

▶ **Fact 4** (Minimax principle). *For any integer $k > 0$, set S, and relation $h \subseteq \{0,1\}^k \times \mathcal{S}$,*

$$\mathsf{R}_\epsilon(h) = \max_\mu \mathsf{D}^\mu_\epsilon(h).$$

A proof of Fact 4 can be found in [6].

Let μ be a probability distribution over $\{0,1\}^k$. We use $\mathrm{supp}(\mu)$ to denote the support of μ. By $x \sim \mu$ we mean that x is a random string drawn from μ. Let $C \subseteq \{0,1\}^k$ be an arbitrary set such that $\Pr_{x\sim\mu}[x \in C] = \sum_{y\in C} \mu(y) > 0$. Then $\mu \mid C$ is defined to be the probability distribution obtained by conditioning μ on the event that the sampled string belongs to C, i.e.,

$$(\mu \mid C)(x) = \begin{cases} 0 & \text{if } x \notin C \\ \frac{\mu(x)}{\sum_{y\in C}\mu(y)} & \text{if } x \in C \end{cases}$$

For a distribution $\mathcal{Q}$ over pairs of distributions (μ_0, μ_1), let $\mathrm{supp}_0(\mathcal{Q}) = \cup\{\mathrm{supp}(\mu_0) : \exists \mu_1, (\mu_0,\mu_1) \in \mathrm{supp}(\mathcal{Q})\}$. Similarly let $\mathrm{supp}_1(\mathcal{Q}) = \cup\{\mathrm{supp}(\mu_1) : \exists \mu_0, (\mu_0,\mu_1) \in \mathrm{supp}(\mathcal{Q})\}$. We say that $\mathcal{Q}$ is *consistent* if $\mathrm{supp}_0(\mathcal{Q})$ and $\mathrm{supp}_1(\mathcal{Q})$ are disjoint sets. We say that $\mathcal{Q}$ is consistent with a (partial) function g if $\mathrm{supp}_0(\mathcal{Q}) \subseteq g^{-1}(0)$ and $\mathrm{supp}_1(\mathcal{Q}) \subseteq g^{-1}(1)$.

▶ **Definition 5** (Subcube, co-dimension). *A subset $\mathsf{C} \subseteq \{0,1\}^m$ is called a subcube if there exists a set $S \subseteq \{1,\ldots,m\}$ of indices and an* assignment function *$\sigma : S \to \{0,1\}$ such that $\mathsf{C} = \{x \in \{0,1\}^m : \forall i \in S, x_i = \sigma(i)\}$. The co-dimension $\mathsf{codim}(C)$ of C is defined to be $|S|$.*

Now we define the composition of a relation and a partial Boolean function.

▶ **Definition 6** (Composition of a relation and a partial Boolean function). *Let $f \subseteq \{0,1\}^n \times S$ and $g \subseteq \{0,1\}^m \times \{0,1\}$ be a relation and a partial Boolean function respectively. The composed relation $f \circ g^n \subseteq (\{0,1\}^m)^n \times S$ is defined as follows: For $x = (x^{(1)},\ldots,x^{(n)}) \in (\{0,1\}^m)^n$ and $s \in S$, $(x,s) \in f \circ g^n$ if and only if one of the following holds:*

- *$x_i \notin g^{-1}(0) \cup g^{-1}(1)$ for some $i \in \{1,\ldots,n\}$.*
- *$x_i \in g^{-1}(0) \cup g^{-1}(1)$ for each $i \in \{1,\ldots,n\}$ and $((g(x_1),\ldots,g(x_n)), s) \in f$.*

We will often view a deterministic query algorithm as a binary decision tree. In each vertex v of the tree, an input variable is queried. Depending on the outcome of the query, the computation goes to a child of v. The child of v corresponding to outcome b of the query is denoted by v_b.

The set of inputs that lead the computation of a decision tree to a certain vertex is a subcube. We will use the same symbol (e.g. v) to refer to a vertex as well as the subcube associated with it.

The execution of a decision tree terminates at some leaf. If the tree computes some relation $h \subseteq \{0,1\}^k \times S$, the leaves are labelled by elements of S, and the tree outputs the label of the leaf at which it terminates. We will also consider decision tree with unlabelled leaves (see Section 4).

3 Conflict Complexity

In this section, we define the conflict complexity and max-conflict complexity of a partial Boolean function g on m bits. For this, we will need to introduce some notation related to a deterministic decision tree T. For a node $v \in T$, let $\pi(v) = \perp$ if v is the root and $\pi(v)$ be the parent of v otherwise. Let $q(v)$ be the index that is queried at v in T, and let $d_T(v)$ be the number of vertices on the unique path in T from the root to v. The depth of the root is 1.

Now fix a partial function $g \subseteq \{0,1\}^m \times \{0,1\}$ and probability distributions μ_0, μ_1 over $g^{-1}(0), g^{-1}(1)$, respectively. Let T be a tree that computes g. For a node $v \in T$ let $p_0(v) = \Pr_{\mu_0}[x_{q(v)} = 0 | x \text{ at } v]$ and $p_1(v) = \Pr_{\mu_1}[x_{q(v)} = 0 | x \text{ at } v]$, and

$$R(v) = \begin{cases} 1 & \text{if } v \text{ is the root} \\ R(\pi(v)) \cdot \min\{\Pr_{\mu_0}[x \to v | x \text{ at } \pi(v)], \Pr_{\mu_1}[x \to v | x \text{ at } \pi(v)]\} & \text{otherwise .} \end{cases}$$

Also define

$$\Delta(v) = |p_0(v) - p_1(v)| \ .$$

To gather intuition about these quantities, imagine a random walk on T that begins at the root. At a node v, this walk moves to the left child with probability $\min\{p_0(v), p_1(v)\}$, and it moves to the right child with probability $1 - \max\{p_0(v), p_1(v)\}$. With the remaining probability, $\Delta(v)$, it terminates at v. Note that for any tree T computing g we have $\sum_{v \in T} \Delta(v) R(v) = 1$. This is because the walk always terminates before it reaches a leaf of T. In particular, this means that $\sum_{v \in T} d_T(v) \Delta(v) R(v)$ – the expected number of steps the walk takes before it terminates – is always at most the depth of the tree T.

▶ **Definition 7** (Conflict complexity and max-conflict complexity)**.** *Let g be a partial function. For distributions μ_0, μ_1 with $\mathrm{supp}(\mu_b) \subseteq g^{-1}(b)$ for $b \in \{0,1\}$, and a deterministic decision tree T computing g, define*

$$\chi(T, (\mu_0, \mu_1)) = \sum_{v \in T} d_T(v) \Delta(v) R(v) \ .$$

The conflict complexity of g is

$$\chi(g) = \max_{\mu_0, \mu_1} \min_T \chi(T, (\mu_0, \mu_1)) \ ,$$

where the maximum is over all pairs of distributions (μ_0, μ_1) supported on $g^{-1}(0)$ and $g^{-1}(1)$ respectively, and the minimum is taken over all deterministic trees T computing g. For $\mathcal{Q}$ a distribution over pairs satisfying $\mathrm{supp}_b(\mathcal{Q}) \subseteq g^{-1}(b)$ for $b \in \{0,1\}$, and T a deterministic tree computing g, let $\chi(T, \mathcal{Q}) = \mathsf{E}_{(\mu_0,\mu_1)\sim\mathcal{Q}}[\chi(T, (\mu_0, \mu_1))]$. Finally, the max-conflict complexity of g is

$$\bar{\chi}(g) = \max_{\mathcal{Q}} \min_T \chi(T, \mathcal{Q}) \ ,$$

where the maximum is taken over $\mathcal{Q}$ with $\mathrm{supp}_b(\mathcal{Q}) \subseteq g^{-1}(b)$ for $b \in \{0,1\}$, and the minimum is taken over deterministic trees T computing g.

We can extend the definition of conflict complexity and max-conflict complexity to more general query processes that do not necessarily compute a function. We first need the notion of FULL.

▶ **Definition 8.** *For a deterministic tree T and pair of distributions (μ_0, μ_1) with disjoint support, we say that $(T, (\mu_0, \mu_1))$ is* FULL *if $\sum_{v \in T} \Delta(v) R(v) = 1$, i.e. if the random walk described above terminates with probability 1. We say that $(T, \mathcal{Q})$ is* FULL *if $(T, (\mu_0, \mu_1))$ is* FULL *for each $(\mu_0, \mu_1) \in \mathrm{supp}(\mathcal{Q})$.*

▶ **Definition 9.** *For a deterministic tree T and pair of distributions (μ_0, μ_1) such that $(T, (\mu_0, \mu_1))$ is* FULL*, define $\chi(T, (\mu_0, \mu_1)) = \sum_{v \in T} d_T(v) \Delta(v) R(v)$. For a distribution $\mathcal{Q}$ such that $(T, \mathcal{Q})$ is* FULL*, define $\chi(T, \mathcal{Q}) = \mathsf{E}_{(\mu_0,\mu_1)\sim\mathcal{Q}}[\chi(T, (\mu_0, \mu_1))]$.*

3.1 Comparison with other query measures

Li [9] shows that the conflict complexity of a total Boolean function g is at least the block sensitivity of g. As mentioned in Section 1.1.3, in this work we show that the max-conflict complexity of a (partial) function g is at least as large as the sabotage complexity of g. For a total Boolean function g, Ben-David and Kothari [4] show that the sabotage complexity of g is at least as large as the fractional block sensitivity of g [1, 13, 7], which in turn is at least as large as the block sensitivity. They also show examples where the sabotage complexity is much larger than the partition bound, quantum query complexity and approximate polynomial degree, thus the same holds for max-conflict complexity as well.

▶ **Theorem 7.** *Let $g \subseteq \{0,1\}^m \times \{0,1\}$ be a partial function. Then $\bar{\chi}(g) \geq \mathrm{RS}(g)$.*

A proof of Theorem 7 can be found in [6].

4 Query Process

We now come to the most important definition of the paper, that of the query process $\mathcal{P}(\mathcal{B}, \mathcal{Q})$. Let $t > 0$ be any integer and $\mathcal{B}$ be any deterministic query algorithm that runs on inputs in $(\{0,1\}^m)^t$. Let $x = (x_i^{(j)})_{\substack{i=1,\ldots,t \\ j=1,\ldots,m}}$ be a generic input to $\mathcal{B}$, and let x_i stand for $(x_i^{(j)})_{j=1,\ldots,m}$. For a vertex v of $\mathcal{B}$, let $v^{(i)}$ denote the subcube in v corresponding to x_i, i.e., $v = v^{(1)} \times \ldots \times v^{(t)}$. Recall from Section 2 that v_b stands for the child of v corresponding to the query outcome being b, for $b \in \{0,1\}$.

The query process $\mathcal{P}(\mathcal{B}, \mathcal{Q})$ runs on an input $z \in \{0,1\}^t$ and uses the BITSAMPLER (Algorithm 1) routine to simulate the queries of $\mathcal{B}$ to x when it can. This process is the heart of how we will transform an algorithm for $f \circ g^n$ into a query efficient algorithm for f.

▶ **Definition 10** (Query process $\mathcal{P}(\mathcal{B}, \mathcal{Q})$)**.** *Let $\mathcal{B}$ be a decision tree that runs on inputs $(\{0,1\}^m)^t$. Let $\mathcal{Q}$ be a consistent probability distribution over pairs of distributions (μ_0, μ_1). The query process $\mathcal{P}(\mathcal{B}, \mathcal{Q})$ is run on an input $z \in \{0,1\}^t$ and is defined by Algorithm 2.*

A few comments about Definition 10. First, we think of $\mathcal{B}$ and $\mathcal{P}$ as query procedures that query input variables and terminate. In particular, they do not have to produce outputs, i.e. their leaves do not have to be labeled. Also note that in Algorithm 2 the segment from line 9 to line 19 corresponds to the BITSAMPLER procedure in Algorithm 1. Queries to the input bits z_i are made in line 15, which corresponds to step 9 of BITSAMPLER.

We now present an important structural result about $\mathcal{P}(\mathcal{B}, \mathcal{Q})$. In particular, this formally proves that the procedure BITSAMPLER given in Algorithm 1 samples the bits from the right distribution.

▶ **Theorem 8.** *Let $\mathcal{B}$ be a deterministic decision tree running on inputs from $(\{0,1\}^m)^t$, and let v be a vertex in $\mathcal{B}$. Let $A_z(v, \mathcal{Q})$ be the event that $\mathcal{P}(\mathcal{B}, \mathcal{Q})$, when run on z, reaches node v. Let $B_z(v, \mathcal{Q})$ be the event that for a random input x sampled from $\gamma_z(\mathcal{Q})$, the computation of $\mathcal{B}$ reaches v. Then for every $z \in \{0,1\}^t$ and each vertex v of $\mathcal{B}$,*

$$\Pr[A_z(v, \mathcal{Q})] = \Pr[B_z(v, \mathcal{Q})] \ .$$

A proof of Theorem 8 is given in [6].

We will be interested in the number of queries $\mathcal{P}(\mathcal{B}, \mathcal{Q})$ is able to simulate before making a query to z_i. To this end, let the random variable $\mathcal{N}_i(\mathcal{B}, z, \mathcal{Q})$ stand for the value of the variable N_i in Algorithm 2 after the termination of $\mathcal{P}(\mathcal{B}, \mathcal{Q})$ on input z. Note that $\mathcal{N}_i$ depends on the randomness in the choices of r (step 9) and also on the randomness in $\mathcal{Q}$ in the choice of distributions $(\mu_0^{(k)}, \mu_1^{(k)})$ (step 4).

Algorithm 2: $\mathcal{P}(\mathcal{B}, \mathcal{Q})$.

Input: $z = (z_1, \ldots, z_t) \in \{0,1\}^t$.

for $1 \le k \le t$ **do**
  $\mathsf{QUERY}_k \leftarrow 0$.
  $\mathsf{N}_k \leftarrow 0$.
  Sample $(\mu_0^{(k)}, \mu_1^{(k)})$ from $\mathcal{Q}$.
$v \leftarrow$Root of $\mathcal{B}$
while *v is not a leaf of* $\mathcal{B}$ **do**
  Let $q(v) = (i, j)$, the j^{th} coordinate of x_i
  if $\mathsf{QUERY}_i = 0$ **then**
    Sample a fresh real number $r \sim [0,1]$ uniformly at random.
    if $r < \min_b \Pr_{x_i \sim \mu_b^{(i)}}[x_i^{(j)} = 0 \mid x_i \in v^{(i)}]$ **then**
      $v \leftarrow v_0$.
    else if $r > \max_b \Pr_{x_i \sim \mu_b^{(i)}}[x_i^{(j)} = 0 \mid x_i \in v^{(i)}]$ **then**
      $v \leftarrow v_1$.
    else
      Query z_i. $\mathsf{QUERY}_i \leftarrow 1$.
      if $r \le \Pr_{x_i \sim \mu_{z_i}^{(i)}}[x_i^{(j)} = 0 \mid x_i \in v^{(i)}]$ **then**
        $v \leftarrow v_0$.
      else
        $v \leftarrow v_1$.
    $\mathsf{N}_i \leftarrow \mathsf{N}_i + 1$.
  else
    $b \leftarrow \begin{cases} 1 & \text{with probability } \Pr_{x_i \sim \mu_{z_i}^{(i)}}[x_i^{(j)} = 1 \mid x_i \in v^{(i)}] \\ 0 & \text{with probability } \Pr_{x_i \sim \mu_{z_i}^{(i)}}[x_i^{(j)} = 0 \mid x_i \in v^{(i)}] \end{cases}$
    $v \leftarrow v_b$

4.1 Relating $\mathcal{P}(\mathcal{B}, \mathcal{Q})$ to max-conflict complexity

A key to our composition theorem will be relating the number of simulated queries made by $\mathcal{P}(\mathcal{B}, \mathcal{Q})$ to max-conflict complexity, which we do in this section. Let $\mathcal{B}$ be a query algorithm taking inputs from $\{0,1\}^m$. In this case, $\mathcal{N}_1(\mathcal{B}, 1, \mathcal{Q}) = \mathcal{N}_1(\mathcal{B}, 0, \mathcal{Q})$. This is because the behavior of $\mathcal{P}(\mathcal{B}, \mathcal{Q})$ on input 0 is exactly the same as the behavior on input 1 before a query to z is made, and after z is queried the value of N_i does not change.

▷ Claim 11. Let $\mathcal{B}$ be an algorithm taking inputs from $\{0,1\}^m$. Then $(\mathcal{B}, \mathcal{Q})$ is FULL if and only if $\mathcal{P}(\mathcal{B}, \mathcal{Q})$ queries z with probability 1. If $(\mathcal{B}, \mathcal{Q})$ is FULL then

$$\chi(\mathcal{B}, \mathcal{Q}) = \mathsf{E}[\mathcal{N}_1(T, 1, \mathcal{Q})]$$

Proof. Note that until z is queried, $\mathcal{P}(\mathcal{B}, (\mu_0, \mu_1))$ exactly executes the random walk described in Section 3, and querying z in $\mathcal{P}(\mathcal{B}, (\mu_0, \mu_1))$ corresponds to this random walk terminating. The first part of the claim then follows as $\mathcal{P}(\mathcal{B}, \mathcal{Q})$ queries z with probability 1 if and only if $\mathcal{P}(\mathcal{B}, (\mu_0, \mu_1))$ queries z with probability 1 for every $(\mu_0, \mu_1) \in \mathrm{supp}(\mathcal{Q})$.

Also because $\mathcal{P}(\mathcal{B}, (\mu_0, \mu_1))$ exactly executes the random walk described in Section 3 we see that $\chi(\mathcal{B}, (\mu_0, \mu_1)) = \mathsf{E}[\mathcal{N}_1(T, 1, (\mu_0, \mu_1))]$. The second part of the claim follows by taking the expectation of this equality over $(\mu_0, \mu_1) \sim \mathcal{Q}$. ◁

The correspondence of claim 11 prompts us to define FULL in a more general setting.

▶ **Definition 12** (FULL). *Let $\mathcal{B}$ be a query algorithm taking inputs from $(\{0,1\}^m)^t$. The pair $(\mathcal{B}, \mathcal{Q})$ is said to be* FULL *if for every $z \in \{0,1\}^t$ it holds that $\mathcal{P}(\mathcal{B}, \mathcal{Q})$ queries z_i with probability 1, for every $i = 1, \ldots, t$.*

5 The Composition Theorem

A proof of Theorem 4 is given in [6].

6 Tightness: $\mathsf{R}_{1/3}(f \circ g^n) \in O\left(\mathsf{R}_{4/9}(f) \cdot \sqrt{\mathsf{R}_{1/3}(g)}\right)$ is possible

In this section we prove Theorem 2. We construct a relation $f_0 \subseteq \{0,1\}^n \times \{0,1\}^n$ (i.e., $\mathcal{S} = \{0,1\}^n$) and a promise function $g_0 \subseteq \{0,1\}^n \times \{0,1\}$ (i.e., $m = n$), such that $\mathsf{R}_{4/9}(f_0) \in \Theta(\sqrt{n})$, $\mathsf{R}_{1/3}(g_0) \in \Theta(n)$ and $\mathsf{R}_{1/3}(f_0 \circ g_0^n) \in \Theta(n)$.

For strings $x = (x_1, \ldots, x_n), z = (z_1, \ldots, x_n)$ in $\{0,1\}^n$, let $x \oplus z$ be the string $(x_1 \oplus z_1, \ldots, x_n \oplus z_n)$ obtained by taking their bitwise XOR. Let $|x|$ stand for the *Hamming weight* $|\{i \in [n] : x_i = 1\}|$ of x. We define f_0 as follows:

$$f_0(z) \stackrel{\text{def}}{=} \left\{(a, z) \in \{0,1\}^n \times \{0,1\}^n \middle| |a \oplus z| \leq \frac{n}{2} - \sqrt{n}\right\}$$

Now we define g_0 by specifying $g_0^{-1}(0)$ and $g_0^{-1}(1)$.

$$g_0^{-1}(0) \stackrel{\text{def}}{=} \left\{(x, 0) \middle| x \in \{0,1\}^n, |x| \leq n/2 - \sqrt{n}\right\},$$
$$g_0^{-1}(1) \stackrel{\text{def}}{=} \left\{(x, 1) \middle| x \in \{0,1\}^n, |x| \geq n/2 + \sqrt{n}\right\}.$$

We now determine the randomized query complexities of f_0, g_0 and $f_0 \circ g_0^n$.

▷ Claim 13.

(i) $\mathsf{R}_{4/9}(f_0) \in \Omega(\sqrt{n})$.

(ii) $\mathsf{R}_{1/3}(g_0) \in \Omega(n)$.

(iii) $\mathsf{R}_{\varepsilon}(f_0 \circ g_0^n) \in O\left(n \cdot \sqrt{\log(1/\varepsilon)}\right)$.

A proof of Claim 13 can be found in [6]. Theorem 2 follows from Theorem 4 and Claim 13 with ε set to $1/3$.

References

1 Scott Aaronson. Quantum certificate complexity. *Journal of Computer and System Sciences*, 74(3):313–322, 2008.

2 Anurag Anshu, Dmitry Gavinsky, Rahul Jain, Srijita Kundu, Troy Lee, Priyanka Mukhopadhyay, Miklos Santha, and Swagato Sanyal. A Composition Theorem for Randomized Query Complexity. In *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017, December 11-15, 2017, Kanpur, India*, pages 10:1–10:13, 2017.

3 Boaz Barak, Mark Braverman, Xi Chen, and Anup Rao. How to Compress Interactive Communication. *SIAM J. Comput.*, 42(3):1327–1363, 2013. doi:10.1137/100811969.

4 Shalev Ben-David and Robin Kothari. Randomized Query Complexity of Sabotaged and Composed Functions. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 60:1–60:14, 2016.

5 Dmitry Gavinsky, Troy Lee, and Miklos Santha. On the randomised query complexity of composition. Technical report, arXiv, 2018. `arXiv:1801.02226`.

6 Dmitry Gavinsky, Troy Lee, Miklos Santha, and Swagato Sanyal. A composition theorem for randomized query complexity via max conflict complexity. *CoRR*, abs/1811.10752, 2018. `arXiv:1811.10752`.

7 Justin Gilmer, Michael Saks, and Srikanth Srinivasan. Composition limits and separating examples for some Boolean function complexity measures. *Combinatorica*, 36(3):265–311, 2016.

8 Peter Høyer, Troy Lee, and Robert Spalek. Negative weights make adversaries stronger. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 526–535, 2007.

9 Yaqiao Li. Conflict complexity is lower bounded by block sensitivity. Technical report, arXiv, 2018. `arXiv:1810.08873`.

10 Ashley Montanaro. A composition theorem for decision tree complexity. *Chicago J. Theor. Comput. Sci.*, 2014, 2014.

11 Ben Reichardt. Reflections for quantum query algorithms. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 560–569, 2011.

12 Swagato Sanyal. A composition theorem via conflict complexity. Technical report, arXiv, 2018. `arXiv:1801.03285`.

13 Avishay Tal. Properties and applications of boolean function composition. In *Innovations in Theoretical Computer Science, ITCS '13*, pages 441–454, 2013.

14 John von Neumann. Zur Theorie der Gessellschaftsspiele. *Math. Ann.*, 100:295–320, 1928.

The Hairy Ball Problem is PPAD-Complete

Paul W. Goldberg
Department of Computer Science, University of Oxford, United Kingdom
Paul.Goldberg@cs.ox.ac.uk

Alexandros Hollender
Department of Computer Science, University of Oxford, United Kingdom
alexandros.hollender@cs.ox.ac.uk

Abstract

The Hairy Ball Theorem states that every continuous tangent vector field on an even-dimensional sphere must have a zero. We prove that the associated computational problem of computing an approximate zero is PPAD-complete. We also give a FIXP-hardness result for the general exact computation problem.

In order to show that this problem lies in PPAD, we provide new results on multiple-source variants of End-of-Line, the canonical PPAD-complete problem. In particular, finding an approximate zero of a Hairy Ball vector field on an even-dimensional sphere reduces to a 2-source End-of-Line problem. If the domain is changed to be the torus of genus $g \geq 2$ instead (where the Hairy Ball Theorem also holds), then the problem reduces to a $2(g-1)$-source End-of-Line problem.

These multiple-source End-of-Line results are of independent interest and provide new tools for showing membership in PPAD. In particular, we use them to provide the first full proof of PPAD-completeness for the Imbalance problem defined by Beame et al. in 1998.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Computational Complexity, TFNP, PPAD, End-of-Line

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.65

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1902.07657`.

Funding *Alexandros Hollender*: Supported by an EPSRC doctoral studentship (Reference 1892947).

1 Introduction

The Hairy Ball Theorem (HBT) is a well-known topological theorem stating that there is no non-vanishing continuous tangent vector field on an even-dimensional k-sphere. It has various informal statements such as "you can't comb a hairy ball flat without creating a cowlick"[1], or "there is a point on the surface of the earth with zero horizontal wind velocity". The HBT is superficially reminiscent of the Borsuk-Ulam Theorem, stating that given any continuous mapping from the 2-sphere to the plane, there are two antipodal points that map to the same value. (Informally, "there are two antipodal points on the surface of the earth where the temperature and pressure are the same"). As we shall see, the present paper highlights a fundamental difference between the two, in terms of the complexity class naturally associated with each of them.

The HBT was first proved in 1885 by Poincaré [29] for the case $k = 2$. The theorem as stated for all even k was proved in 1912 by Brouwer [5]. Accordingly, this result is sometimes also called the Poincaré-Brouwer theorem. In fact, the result proved by Poincaré [29] is

[1] `https://en.wikipedia.org/wiki/Hairy_ball_theorem`

stronger than stated above. It follows from it that for any (sufficiently well-behaved) 2-dimensional manifold with genus $g \neq 1$, any continuous tangent vector field must have a zero. In particular, this means that the HBT also holds for the torus of genus g for $g \geq 2$, i.e. the 2-dimensional torus with g holes. It is easy to see that it does not hold for the standard single-hole torus.

Over the years, various papers in the *American Mathematical Monthly* have presented alternative proofs of the Hairy Ball Theorem and variants, for example [22, 26, 4, 14, 24, 9].

Topological existence results (such as the HBT, Borsuk-Ulam, and the Brouwer and Banach fixpoint theorems) have a very interesting relationship with complexity classes of search problems in which any instance has a guaranteed solution. Any such theorem has a corresponding computational challenge, of searching for such a solution, given a circuit that computes an appropriate function. The assumption that these complexity classes are distinct from each other (the ones of main interest here being PPAD and PPA, discussed below in more detail) provides a taxonomy of these theorems. Our results highlight a fundamental distinction between the HBT and Borsuk-Ulam, by showing that the corresponding search problem for the HBT is characterised by the complexity class PPAD, in contrast to Borsuk-Ulam, which is characterised by PPA [1]. The complexity-theoretic analysis of topological search problems provides a well-defined sense in which the HBT is "Brouwer-like" rather than "Borsuk-Ulam-like". It has previously been noted that the HBT may be used to prove Brouwer's fixed point theorem [26], but not the other way around.

1.1 Background on NP total search and PPAD

The complexity class TFNP is the set of all *total* function computation problems in NP: functions where *every* input has an efficiently-checkable solution (in Section 2.2 we give a precise definition). Many problems in TFNP appear to be computationally difficult, notably Factoring, the problem of computing a prime factorisation of a given number, also Nash, the problem of computing a Nash equilibrium of a game. However, such problems are unlikely to be NP-hard, due to the 1991 result of Megiddo and Papadimitriou [25] showing that TFNP problems cannot be NP-hard unless NP is equal to co-NP. This basic fact, that hard TFNP problems are in a very strong sense "NP-intermediate", provides TFNP's strong theoretical appeal. This has led to the classification of these problems in terms of certain syntactic subclasses of TFNP, whose problems are shown to be total due to some basic combinatorial principle. The best-known of these classes are PLS, PPP, PPAD, and PPA, identified by Papadimitriou in 1994 [28].

- PPAD consists of problems whose totality is based on the principle that given a source in a directed graph whose vertices have in-degree and out-degree at most 1, there exists another degree-1 vertex. Its canonical problem End-of-Line consists of an exponentially-large graph of this kind, presented concisely via a circuit.
- PPA differs from PPAD in that the graph need not be directed; being a more general principle, PPA is thus a superset of PPAD. Its canonical problem Leaf is similar, only the graph is undirected.

Subsequently, many TFNP problems of interest were shown PPAD-complete [10, 7, 23, 13], while more recently others were shown PPA-complete [17, 18]. Despite their similar definitions, PPAD and PPA are usually conjectured to be different, and (along with other syntactic TFNP subclasses) are separated by oracles [3].

1.2 Our results and their significance

Given the long-standing interest in the Hairy Ball Theorem, it is natural to study the corresponding computational search problem. In this paper, we prove that computing an approximate zero of a Hairy Ball vector field is PPAD-complete. While many PPAD-completeness results already exist, a noteworthy novelty of our results is that we find that computing HBT solutions corresponds with *multiple-source* variants of the End-of-Line problem: given a large directed graph implicitly represented by a circuit, suppose you are shown several sources and told to find another degree-1 vertex. This is in contrast with previous PPAD-complete problems that naturally reduce to standard single-source End-of-Line.

In Section 6 we prove that these multiple-source End-of-Line variants are PPAD-complete (membership of PPAD being the tricky aspect). Our results make progress on the general question (studied in [19]) of whether there exist combinatorial principles indicating totality of search problems, that are fundamentally different from the known ones that give rise to complexity classes such as PPAD. In particular, in Section 6.3, we note that a proof of PPAD-completeness for the Imbalance problem by Beame et al. [3] is incomplete and provide a full proof using our results.

The generalisation of Poincaré's result to higher dimensions is called the Poincaré-Hopf theorem (see e.g. [20]). This theorem relates the number and types of zeros of a vector field on a manifold with its Euler characteristic, a topological invariant. In particular, if the Euler characteristic of a manifold is not 0, then any continuous tangent vector field on the surface must have a zero. The Euler characteristic of even-dimensional spheres is 2, while it is $2(1-g)$ for 2-dimensional toruses of genus $g \geq 2$. For odd-dimensional spheres it is 0.

We believe that the reduction to multiple-source End-of-Line is not an artefact of our techniques, but instead intrinsically related to the Euler characteristic of the domain. Indeed, the reduction from the HBT problem on even-dimensional spheres to End-of-Line yields 2 sources (Section 4). On the other hand, if we consider the HBT problem on the 2-dimensional torus of genus $g \geq 2$, then we obtain $2(g-1)$ sources (Section 7 of the full version). The connection between HBT and directed graph problems has previously only appeared in a proof for the 2-dimensional sphere case [22].

Finally, we note that PPAD-hardness is obtained by constructing a HBT vector field from multiple copies of a discrete Brouwer fixpoint problem. The usage of multiple copies is a new conceptual feature, closely related to the multi-source aspect. Using the same high-level idea, we also provide a FIXP-hardness result for the problem of computing an *exact* solution (Section 5.2).

1.3 Other related work

Banach's Fixed Point Theorem [2] says that a *contraction map* has a unique fixpoint. Its corresponding computational problem Contraction, is to find a fixed point of a given contraction map. Some versions of Contraction have been shown complete for CLS, a subclass of PPAD [11, 12, 16]. The search for Brouwer fixpoints (including discretised versions of Brouwer functions) is PPAD-complete for most variants of the problem [28, 7], which is why we say the HBT is "Brouwer-like". Finally – in contrast – the computational problem of searching for a Borsuk-Ulam solution is PPA-complete [1]. Other topological existence results that have PPA-complete search problems include the Hobby-Rice theorem [17] and the Ham Sandwich Theorem [18].

2 Preliminaries

Let k be a positive integer. For $x \in \mathbb{R}^k$, $\|x\|_2$, $\|x\|_1$ and $\|x\|_\infty$ denote the standard ℓ_2-norm, ℓ_1-norm and ℓ_∞-norm respectively. For $x, y \in \mathbb{R}^k$, $\langle x, y\rangle := \sum_{i=1}^m x_i y_i$ denotes the inner product.

The k-dimensional unit sphere in $\mathbb{R}^{k+1}$ (or k-sphere) is denoted $S^k = \{x \in \mathbb{R}^{k+1} : \|x\|_2 = 1\}$. A continuous tangent vector field on S^k is a continuous function $f : S^k \to \mathbb{R}^{k+1}$ such that for all $x \in S^k$ we have $\langle f(x), x\rangle = 0$. The Hairy Ball Theorem can be stated as follows:

▶ **Theorem 1** (Poincaré [29]–Brouwer [5]). *If $k \geq 2$ is even, then for any continuous tangent vector field $f : S^k \to \mathbb{R}^{k+1}$, there exists $x \in S^k$ such that $f(x) = 0$.*

2.1 Model of Computation

We work in the standard Turing machine model. All numbers appearing in computations are rational numbers where the numerator and denominator are integers represented in binary. For a rational number x, size(x) denotes the size of the representation of x, i.e. the sum of the representation length of its numerator and denominator in binary. For an arithmetic circuit F, size(F) denotes the number of gates in the circuit added to the representation length of any rational constants used by the circuit.

2.2 Formal definition of TFNP

A computational search problem is given by a binary relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$, interpreted as follows: $y \in \{0,1\}^*$ is a solution to instance $x \in \{0,1\}^*$, if and only if $(x, y) \in R$. The search problem R is in FNP (*Functions in NP*), if R is polynomial-time computable (i.e. $(x, y) \in R$ can be decided in polynomial time in $|x| + |y|$) and there exists some polynomial p such that $(x, y) \in R \implies |y| \leq p(|x|)$. Here $\{0,1\}^*$ denotes all finite length bit-strings and $|x|$ is the length of bit-string x.

The class TFNP (*Total Functions in NP* [25]) contains all search problems R that are in FNP and are *total*, i.e. every instance has at least one solution. Formally, this corresponds to requiring that for every $x \in \{0,1\}^*$ there exists $y \in \{0,1\}^*$ such that $(x, y) \in R$.

Let R and S be total search problems in TFNP. We say that R (many-one) reduces to S, if there exist polynomial-time computable functions f, g such that

$$(f(x), y) \in S \implies (x, g(x, y)) \in R.$$

Note that if S is polynomial-time solvable, then so is R. We say that two problems R and S are (polynomial-time) equivalent, if R reduces to S and S reduces to R.

To be PPAD-complete, a problem must be equivalent to End-of-Line (Definition 12); in Section 6 we show that the multiple-source version MS-EoL (Definition 13) is equivalent.

3 The Hairy-Ball Problem

3.1 The kD-Hairy-Ball problem

The Hairy Ball Theorem naturally yields a corresponding computational problem. We are given a continuous tangent vector field f on the unit sphere and have to find a point where it is zero. In trying to formalise this, some issues need to be addressed. First, one has to decide how the vector field should be represented in the input. Here we take the usual approach of assuming that it is represented as an arithmetic circuit.

Before we discuss the types of gates that we want to allow in the circuit, let us briefly handle the second issue: the vector field might not have a rational zero. Indeed, consider the following example: at $x \in S^2$ the vector field is simply the vector $(1, 1, 1)$ projected onto the tangent space of S^2 at x. In this case, the only solutions are $\pm(1/\sqrt{3}, 1/\sqrt{3}, 1/\sqrt{3})$. Thus, we cannot hope to always output an exact solution. We bypass this problem by asking for an *approximate* solution instead, i.e. a point $x \in S^2$ such that $\|f(x)\|_\infty \leq \varepsilon$ for some $\varepsilon > 0$ provided in the input. This notion of approximate solution is the standard one used when studying topological existence theorems in the context of TFNP (e.g. Brouwer's fixed point theorem or the Borsuk-Ulam theorem).

As mentioned above, the vector field will be represented as an arithmetic circuit. In the case of S^2, the circuit will have three input gates and three output gates. The arithmetic circuit will be allowed to use gates $\{+, -, \times\zeta, \max, \min\}$ and rational constants. All the gates have fan-in 2, except $\times\zeta$ which has fan-in 1 and corresponds to multiplication by a rational constant ζ. Note that such a circuit is polynomially equivalent to a circuit only using gates $\{+, \times\zeta, \max\}$ and rational constants, since the other gates can be efficiently simulated using these. These circuits correspond to LINEAR-FIXP-type circuits that are known to be sufficient to obtain PPAD-hardness of Brouwer [15]. A discussion about why we don't use more powerful gates in our definition can be found in Section 3.2 of the full version.

This type of circuit yields piece-wise affine functions that are continuous. Furthermore, it has the following nice property: for any such arithmetic circuit F, and any rational x, we can compute $F(x)$ exactly in polynomial time in size(F) and size(x). One potential issue is that F might not be *tangent* to the sphere, but this is easy to fix by simply considering the vector field given by the projection of F onto the corresponding tangent space to the sphere. Thus, we define the computational problem as follows:

▶ **Definition 2** (kD-Hairy-Ball)**.** *Let $k \geq 2$ be even. The* kD-Hairy-Ball *problem is defined as: given $\varepsilon > 0$ and an arithmetic circuit F with $k + 1$ inputs and outputs, using gates $\{+, \times\zeta, \max\}$ and rational constants, find $x \in S^k$ such that $\|P_x[F(x)]\|_\infty \leq \varepsilon$.*

Here $P_x[\cdot]$ denotes the projection onto the tangent space to the sphere S^k at $x \in S^k$. Note that for any $v \in \mathbb{R}^{k+1}$, we have $P_x[v] = v - \langle v, x\rangle x$, because $\|x\|_2 = 1$. Thus, the projection of any rational vector v onto the tangent space at rational $x \in S^k$ can be computed exactly in polynomial time in size(v) and size(x). Note that we are looking for a solution with respect to the ℓ_∞-norm, but we could also have used the ℓ_2- or ℓ_1-norm, since all these versions are computationally equivalent.

kD-Hairy-Ball lies in TFNP. Clearly, any solution can be checked in polynomial time. Totality of kD-Hairy-Ball will immediately follow when we prove that it lies in PPAD (Corollary 8). The following Lemma is proved in Section 3.1 of the full version.

▶ **Lemma 3.** *Let $k \geq 2$ be even. Let F be an arithmetic circuit with $k + 1$ inputs and outputs, using gates $\{+, \times\zeta, \max\}$ and rational constants. Then, the function $S^k \to \mathbb{R}^{k+1}$, $x \mapsto P_x[F(x)]$ is Lipschitz-continuous with Lipschitz constant $L = k \cdot 2^{\text{size}(F)^2+3}$ (w.r.t. ℓ_∞-norm).*

Our main result is Theorem 4. Containment in PPAD, which turns out to be the most challenging part of this result, is presented in Section 4 (using the multiple-source End-of-Line results of Section 6). PPAD-hardness is presented in Section 5.

▶ **Theorem 4.** *For all even $k \geq 2$, kD-Hairy-Ball is* PPAD*-complete.*

4 The Hairy-Ball Problem is in PPAD

In this section we present our main result: the problem of computing an approximate Hairy Ball solution reduces to End-of-Line, the canonical PPAD-complete problem.

From a purely mathematical standpoint, our proof can be used to provide a (fairly cumbersome) proof of the Hairy Ball theorem by using Brouwer's fixed point theorem. Indeed, it is known [28] that End-of-Line reduces to Brouwer (in fact, even 2D-Brouwer [7]). Thus, given a Hairy Ball function f, using our reduction and Brouwer's fixed point theorem, one can prove the existence of a point x_k such that $\|f(x_k)\| \leq 1/2^k$ for any k (using the fact that f must be uniformly continuous since the sphere is compact). Then, since any sequence in a compact set must have a converging subsequence it follows that there must exist x such that $f(x) = 0$. Finding a more direct way to deduce the Hairy Ball theorem from Brouwer's fixed point theorem is an interesting open question.

Our goal is to prove that the Hairy Ball problem lies in PPAD in a setting that is as general and encompassing as possible. The way the function is represented, as a circuit or otherwise, should not play a role. Thus, we are only going to make two assumptions about the tangent vector field: that it can be evaluated in polynomial time and that it is polynomially continuous in some well-defined sense. The first assumption is very natural: if we are given a Hairy Ball function, we expect to be able to evaluate it efficiently. The motivation for the second assumption is that if we omit it, then there is no guarantee that there will exist an approximate solution with representation size that is polynomial in the input size.

We now define these assumptions formally, following the analogous definitions by Etessami and Yannakakis [15] for Brouwer fixed point problems. Let $\mathcal{F}$ be a class of Hairy Ball functions $f : S^k \to \mathbb{R}^{k+1}$ (i.e. continuous tangent vector fields) with $k \geq 2$ even. Note that here k is not fixed for all $f \in \mathcal{F}$, but we assume that $k \leq \text{size}(f)$. For any $f \in \mathcal{F}$, $\text{size}(f)$ denotes the length of the representation of f in $\mathcal{F}$. In the case of kD-Hairy-Ball, k is fixed and $\mathcal{F}$ is the class of all such functions represented using arithmetic circuits with gates $\{+, \times\zeta, \max\}$ (with the projection onto the tangent space at the end). In that case, $\text{size}(f)$ is the size of the circuit representing f. Recall that for rational vector x, $\text{size}(x)$ is the length of the representation of x.

▶ **Definition 5** ([15]). *Let $\mathcal{F}$ be a class of Hairy Ball functions.*

- *$\mathcal{F}$ is* polynomially computable*, if there exists some polynomial p such that for any $f \in \mathcal{F}$ and any rational input $x \in S^k$, $f(x)$ can be computed in time $p(\text{size}(f) + \text{size}(x))$.*
- *$\mathcal{F}$ is* polynomially continuous*, if there exists some polynomial q such that for any $f \in \mathcal{F}$ and any rational $\varepsilon > 0$, there exists a rational $\delta > 0$ with $\text{size}(\delta) \leq q(\text{size}(f) + \text{size}(\varepsilon))$ such that for all $x, y \in S^k$ we have $\|x - y\|_\infty \leq \delta \implies \|f(x) - f(y)\|_\infty \leq \varepsilon$.*

Note that kD-Hairy-Ball yields a class $\mathcal{F}$ that is both polynomially computable and polynomially continuous (by Lemma 3).

▶ **Definition 6.** *Let $\mathcal{F}$ be a class of Hairy Ball functions. The problem* Hairy-Ball($\mathcal{F}$) *is defined as: given $f \in \mathcal{F}$ and $\varepsilon > 0$, find $x \in S^k$ such that $\|f(x)\|_\infty \leq \varepsilon$.*

For simplicity we assume that we can recognise whether some string in $\{0,1\}^*$ represents an element $f \in \mathcal{F}$ in polynomial time. If this does not hold, then Hairy-Ball($\mathcal{F}$) has to be studied as a *promise* problem. The reduction to End-of-Line given in the proof below still holds. However, this does not imply that the problem lies in PPAD, because TFNP requires the problem to be total without any promise.

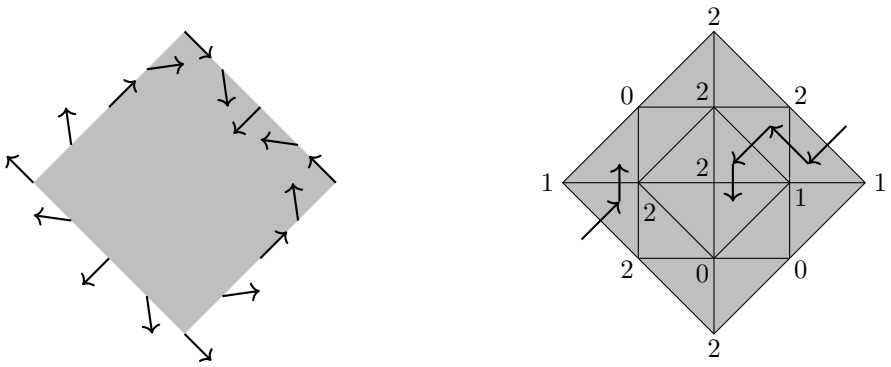
▶ **Theorem 7.** *Let $\mathcal{F}$ be a class of Hairy Ball functions that is polynomially computable and polynomially continuous. Then,* Hairy-Ball($\mathcal{F}$) *lies in* PPAD.

▶ **Corollary 8.** *For all even $k \geq 2$, k*D-Hairy-Ball *lies in* PPAD.

Proof Overview for Theorem 7. The proof can be subdivided into two parts. In the first part, we reduce kD-Hairy-Ball to a *2-source* End-of-Line problem. In the second part, we show that the 2-source version reduces to the standard version of End-of-Line, where a single source is known. Surprisingly, the reduction from 2 sources to 1 is non-trivial. The proof for this is presented separately in Section 6. In fact, we prove the more general result: as long as the number of known sources in an End-of-Line instance is polynomial, we can reduce to standard End-of-Line. Various implications of this result are also presented in Section 6.

We now give some details about the first part of the proof, in which we reduce kD-Hairy-Ball to 2-source End-of-Line through a Sperner argument. The inspiration for this comes from a proof of the 2-dimensional Hairy Ball Theorem via a version of Sperner's Lemma, given by Jarvis and Tanton [22]. Our contribution here is two-fold: we extend their proof to any higher (even) dimension and we turn it into a polynomial-time reduction.

In order to obtain a polynomial-time reduction, instead of working directly on the sphere, we use a stereographic projection to "unfold" the sphere S^k ($\subset \mathbb{R}^{k+1}$) into the space $\mathbb{R}^k$, along with the vector field. Then, we consider a sufficiently large cross-polytope C of $\mathbb{R}^k$ and prove that the "unfolded" vector field satisfies certain boundary conditions. In the case $k = 2$, this corresponds to the vector field making two full rotations when we move along the boundary of C (see Figure 1a). Next, we pick an efficient triangulation of C and a suitable colouring of its nodes. The last step then requires us to prove that this colouring yields exactly two starting points (on the boundary) for Sperner paths (see Figure 1b) that lead to *panchromatic* simplices. Using standard Sperner-arguments this yields a 2-source End-of-Line instance. The full proof for any even k can be found in Section 4.2 of the full version. Note that, as expected, the proof does not work if k is odd. Indeed, the construction then yields a starting point and an ending point on the boundary, instead of two starting points. ◀



(a) Boundary conditions after "unfolding" ... **(b)** ... yielding a Sperner instance with two sources.

Figure 1 An example for the proof of Theorem 7 in the case $k = 2$. The region C is represented in grey.

5 Computational Hardness for Hairy Ball Problems

It is possible to prove Brouwer's fixed point theorem using the Hairy Ball Theorem as follows (see [26] for the full details). Let $B_k \subset \mathbb{R}^k$ be the unit ball. If we assume that a function $f : B_k \to B_k$ does not have any fixed point, then we can construct a Hairy Ball function $g : S^k \to \mathbb{R}^k$ that does not have a zero. Brouwer's theorem follows by contradiction. The main idea for the construction of g is the following. Consider $f'(x) = f(x) - x$ and assume that it points directly inward on the boundary of B_k. Take one copy of B_k with the vector field f' and one copy with the vector field $-f'$, and glue their boundaries together. The resulting object can be deformed to yield the sphere S^k and the vector fields will perfectly match on the glued region. Thus, assuming that f' has no zero, g will have no zero either.

By making this idea fully constructive and efficient, we obtain reductions from Brouwer problems to Hairy Ball problems. Thus, existing PPAD- and FIXP-hardness results for Brouwer also hold for the corresponding Hairy Ball problems. We note that these reductions always involve using *two* copies of a Brouwer instance to obtain a single Hairy Ball instance. This further supports our claim that the fact that we obtain two sources when reducing kD-Hairy-Ball to End-of-Line (Section 4) is not an artefact of our reduction, but intrinsic to the problem.

5.1 PPAD-hardness

▶ **Theorem 9.** *For all even $k \geq 2$, k*D-Hairy-Ball *is* PPAD*-hard.*

Note that this result is particularly strong, because the type of circuit allowed in the definition of kD-Hairy-Ball (Definition 2) is particularly weak. Furthermore, the hardness is proved for inversely exponential ε/L (where L is the Lipschitz constant of the function), which is the best we can hope for in the fixed dimension case. Indeed, if ε/L is inversely polynomial, then the following is a polynomial time algorithm that solves the problem: divide the domain into a sufficiently small (but polynomial) number of regions and check an arbitrary point in each region. In the case where the dimension is not fixed, it seems likely that the problem should be hard even for constant ε/L, but we do not investigate this in this paper.

Proof Overview. One way to prove this result is to take an instance F of a 2D-Brouwer problem, which is known to be PPAD-complete [7], and embed a copy F and a copy $-F$ on the south and north hemisphere of S^2 respectively. However, since our 2D-Hairy-Ball circuit can only use gates in $\{+, \times\zeta, \max\}$, we first shrink the domain of F so that we embed it in a small region around the south pole. This ensures that even after projection onto the tangent space, no bogus solutions will appear. We do the same for $-F$ in the north pole and then define G on the rest of the sphere in such a way that no solution appears there.

In spirit, we follow this general proof idea, but take a slight detour, because it gives us the chance to define and study a discrete analog to 2D-Hairy-Ball: the 2D-Hairy-Cube problem. Intuitively, this problem is obtained from 2D-Hairy-Ball the same way that discrete 2D-Brouwer is obtained from continuous 2D-Brouwer. The domain is discretised by a grid and a circuit computes the local direction of the function in every grid-square. The natural way to discretise the sphere S^2 is to replace it by a cube with a grid on each face. The advantage of 2D-Hairy-Cube is that PPAD-hardness is easy to prove: just put a (slightly modified) discrete 2D-Brouwer instance on one face, and the inverse instance on the opposite face. Defining the instance on the remaining faces is trivial in this case. In Section 6 of the full version we define the problem and prove PPAD-hardness. An illustration of a 2D-Hairy-Cube instance is given in Figure 2.

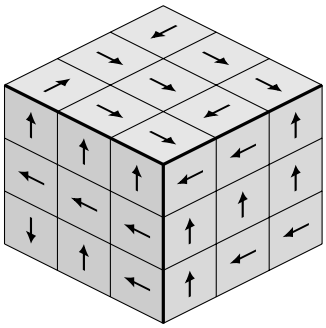


Figure 2 Partial view of a 2D-Hairy-Cube instance.

The next step is reducing 2D-Hairy-Cube to 2D-Hairy-Ball. Even though it seems natural that this should hold, the reduction is technically tedious. In particular, we have to simulate a Boolean circuit using an arithmetic circuit, but the input bits cannot always be computed exactly. Thus, we need to use an averaging trick (see [10, 8]). The details are in Appendix A of the full version.

This yields PPAD-hardness for 2D-Hairy-Ball. The final step is to extend this to kD-Hairy-Ball, by showing that kD-Hairy-Ball reduces to $(k+2)$D-Hairy-Ball. The proof can be found in Appendix B of the full version. ◀

5.2 FIXP-hardness

Up to this point we have only considered the problem of computing an *approximate* Hairy Ball solution. However, there are other computational problems one could consider, e.g. computing a point that is close to an *exact* solution, or computing the first n bits of an *exact* solution.

The corresponding problems for Brouwer fixed points have been studied by Etessami and Yannakakis [15]. In particular, they define the class FIXP that captures the complexity of computing an exact fixed point of a function given by an arithmetic circuit and mapping the unit cube into itself. They prove that computing an exact Nash equilibrium of a 3-player game is FIXP-complete. In doing so, they use a special type of reduction, called an SL-reduction, that ensures that the reduction also holds for three problems that can be studied in the standard Turing machine model: the "strong approximation problem" (i.e. find a point close to an exact solution), the "partial computation" problem (i.e. compute the first n bits of an exact solution) and various decision problems. This means that computing a strong approximation for 3-player Nash is as hard as computing a strong approximation of a Brouwer function given by an arithmetic circuit. We prove that the corresponding problems for the Hairy Ball Theorem are at least as hard as their Brouwer counterparts.

▶ **Definition 10.** *The* Exact-Hairy-Ball *problem is defined as: given an arithmetic circuit F (with gates $\{+,-,\times,/,\max,\min\}$, rational constants and that never divides by 0) that computes a tangent vector field $S^k \to \mathbb{R}^{k+1}$, k even, find $x \in S^k$ such that $F(x) = 0$.*

Note that the vector field will be continuous since we never divide by 0. The vector field can be syntactically forced to be tangent to the sphere because we can compute the projection exactly using this kind of circuit.

▶ **Theorem 11.** Exact-Hairy-Ball *is* FIXP*-hard. Furthermore, the corresponding strong approximation, partial computation and decision problems are also hard for the corresponding versions of* FIXP *(as defined in [15]).*

Proof Overview. We embed two copies of a Brouwer instance (after some preprocessing) on the sphere such that any exact Hairy Ball solution yields an exact Brouwer fixed point. The details can be found in Section 5.2 of the full version. Note that this reduction makes use of the $\times$ and / gates and cannot be used to prove PPAD-hardness of kD-Hairy-Ball. ◀

Following Etessami and Yannakakis, we could define a class HB that captures the complexity of computing an exact Hairy Ball solution (and corresponding versions of the class for the related problems) by taking the set of all problems that reduce to Exact-Hairy-Ball. Then, Theorem 11 is saying that FIXP $\subseteq$ HB. Note that the three discrete problems that can be studied in the Turing machine model lie in PSPACE, by using the same technique as in [15, Proposition 4.2].

6 End-of-Line: One Source to Rule Them All

End-of-Line is the canonical problem used to define PPAD. Investigating variants of the problem is of independent interest, in particular in order to gain a better understanding of PPAD and how it relates to other similar subclasses of TFNP. An additional motivation for studying these variants is given by this paper, since a multiple-source variant of End-of-Line is used to prove that finding an approximate Hairy Ball solution lies in PPAD (Section 4).

This section is an improved version of the corresponding content in the technical report [21]. In [21] we use these results to show that a computational problem related to the Mutilated Chessboard puzzle is PPAD-complete.

6.1 The End-of-Line problem

The End-of-Line problem is informally defined as follows: given a directed graph where each vertex has in- and out-degree at most 1 and given a known source of this graph, find a sink or another source. The problem is computationally challenging, because the graph is not given explicitly in the input. Instead, we are given an implicit concise representation of the graph through circuits that compute the predecessor and successor of a vertex in the graph. In what follows, we sometimes interpret the input and output of the circuits, which are elements in $\{0,1\}^n$, as the numbers $\{0,1,\ldots,2^n-1\}$.

▶ **Definition 12** (End-of-Line [10]). *The* End-of-Line *problem is defined as: given Boolean circuits S, P with n input bits and n output bits and such that $P(0) = 0 \neq S(0)$, find x such that $P(S(x)) \neq x$ or $S(P(x)) \neq x \neq 0$.*

The circuits define a graph as follows. There is a directed edge from vertex x to y ($x \neq y$), if and only if $S(x) = y$ and $P(y) = x$. Note that any badly defined edge, i.e. $S(x) = y$ and $P(y) \neq x$, or $P(y) = x$ and $S(x) \neq y$, qualifies as a solution of End-of-Line as defined above (because $P(S(x)) \neq x$ or $S(P(x)) \neq x$ respectively). Note that 0 is a source of the graph, unless $P(S(0)) \neq 0$, in which case 0 is a valid solution to the problem as stated above.

It is easy to check that this formal definition of the problem is computationally equivalent to the informal description given above. By definition, End-of-Line is PPAD-complete [28]. Furthermore, reduction from End-of-Line is a very common technique to show PPAD-hardness (e.g. [10, 7]).

6.2 Multiple-Source End-of-Line

What if, instead of just one, we already know *two* sources of an End-of-Line instance? We are still interested in finding any sink or any *other* source. Intuitively, the problem might seem easier, because the existence of two sources implies the existence of at least two sinks, hence more potential solutions. In fact, it is easy to see that this problem is actually at least as hard as End-of-Line: just duplicate the whole End-of-Line instance.

The other direction, however, is not trivial. Indeed, if we interpret our 2-source End-of-Line instance as a standard End-of-Line instance (and pick one of the two sources as the standard source), then the other known source is a valid solution to End-of-Line, but not a valid solution to our original problem. In other words, it is not clear how to solve this problem if we are given access to an oracle solving End-of-Line, because the oracle could just return the other known source. We consider the following more general problem, where we are given an End-of-Line graph and an explicit list of known sources.

▶ **Definition 13** (MS-EoL). *The* Multiple-Source End-of-Line *problem, abbreviated* MS-EoL*, is defined as: given circuits S, P with n inputs and n outputs and $s_1, \dots, s_k \in \{0,1\}^n$ such that $P(s_i) = s_i \neq S(s_i)$ for all i, find $x \in \{0,1\}^n$ such that $P(S(x)) \neq x$ or $x \notin \{s_1, \dots, s_k\}$ such that $S(P(x)) \neq x$.*

In passing, let us note that in the undirected case this kind of generalisation is trivial. The undirected analogue of End-of-Line is Leaf: given an undirected graph where every vertex has degree at most 2 and given a vertex of degree 1, find another vertex of degree 1, i.e. another leaf. Assume that we know k leaves instead of just one. If k is even, then the problem is not even in TFNP. If k is odd, then we can add edges between known leaves until exactly one is left. Thus, the problem is equivalent to Leaf. This kind of reduction does not work for the directed case. Nevertheless, we obtain[2]:

▶ **Theorem 14.** Multiple-Source End-of-Line *is equivalent to* End-of-Line.

Proof Overview. We inductively construct a vertex set containing various combinations of subsets of the original graph. The successor and predecessor circuits are carefully constructed so as to ensure that any solution yields a solution to the original problem. The full proof can be found in Section 8.2 of the full version. ◀

▶ Remark (Multiple Known Sources and Sinks). A natural generalisation of Multiple-Source End-of-Line is the following problem: given an End-of-Line graph and a list of k known sources and m known sinks, find another source or sink. Note that for this problem to be in TFNP, we must require $k \neq m$. Using Theorem 14, it is easy to see that this problem is equivalent to End-of-Line. If $k > m$, then we add an edge from each of the m known sinks to some corresponding known source and obtain an instance with $k - m$ known sources and no known sinks. Similarly, if $k < m$, then we first reverse all directed edges and then apply the same trick.

The next two sections present additional consequences of Theorem 14.

[2] This problem was discussed in an online thread (`https://cstheory.stackexchange.com/q/37481`). E. Jeřábek proved membership in PPADS and PPA-p for every prime p (but not membership in PPAD).

6.3 The Imbalance problem

Up to this point, we have only considered graphs where every vertex has in- and out-degree at most 1. However, the principle that guarantees the existence of a solution in an End-of-Line graph can be generalised to higher degree graphs. If we are given a directed graph and an *unbalanced* vertex, i.e. a vertex with in-degree $\neq$ out-degree, then there must exist another unbalanced vertex.

Beame et al. [3] defined the corresponding problem Imbalance, which is seemingly more general than End-of-Line. In this problem, every vertex is not constrained to have in- and out-degree at most 1. Instead, in- and out-degree are bounded by some polynomial of the input length[3]. We are given a vertex that is unbalanced and have to find another unbalanced vertex (which is guaranteed to exist). The problem can be informally defined as follows:

▶ **Definition 15** (Imbalance [3], informal)**.** *The* Imbalance *problem is defined as: given a directed graph (represented concisely by predecessor and successor functions) and a vertex z that has in-degree $\neq$ out-degree, find a vertex $x \neq z$ that also has in-degree $\neq$ out-degree.*

Beame et al. [3] claim that Imbalance reduces to End-of-Line, using the same argument as for the corresponding problems on undirected graphs. However, if the graph is directed, a complication arises (that is not an issue in the undirected case). Indeed, their proof idea is incomplete, because they overlook the fact that their reduction yields an End-of-Line instance with *multiple* known sources. Using Theorem 14 we can provide a full proof of their claim.

▶ **Theorem 16.** Imbalance *is* PPAD*-complete.*

A formal definition of the problem and a full proof can be found in Section 8.3 of the full version.

6.4 Looking for multiple solutions

If we are given an End-of-Line instance with k known sources, then we can ask for k sinks or k unknown sources. The problem is total, because at least k sinks are guaranteed to exist.

▶ **Definition 17** (k-EoL)**.** *Let $k \in \mathbb{N}$. The* k-Ends-of-Line *problem, abbreviated* k-EoL*, is defined as: given circuits S, P with n inputs and n outputs and such that $P(z) = z \neq S(z)$ for all $z < k$, find distinct $x_1, \ldots, x_k$ such that $P(S(x_i)) \neq x_i$ for all i or $S(P(x_i)) \neq x_i \geq k$ for all i.*

Intuitively, this problem seems harder than End-of-Line or MS-Eol, because we are now looking for more than one solution. However, using Theorem 14 we can show:

▶ **Theorem 18.** *For any $k \in \mathbb{N}$,* k-Ends-of-Line *is* PPAD*-complete.*

Proof Overview. The reduction to End-of-Line is obtained by constructing a Turing reduction (using Theorem 14) and then applying a result by Buss and Johnson [6] that PPAD is closed under Turing reductions. The details are in Section 8.4 of the full version. ◀

▶ Remark. The same proof also yields PPAD-completeness for the following problem. Fix some polynomial p. The problem is: given k sources, find k sinks or $p(k)$ sources. This seems quite surprising, as one might have expected this problem to be closer to PPADS.

[3] Note that this trivially holds, if the input consists of circuits that explicitly output the predecessor and successor list.

Some analogous results for the class PPADS and its canonical complete problem Sink [27, 3] are presented in Section 8.4 of the full version. Sink is identical to End-of-Line, except that we only accept a sink as a solution and are not interested in other sources. In this case the results are easier to obtain, because there is no need for an analogue of Theorem 14.

7 Conclusion and Further Work

We have obtained a satisfying answer to the question of the computational complexity of the Hairy Ball Theorem, if we are looking for an approximate solution. For other solution concepts related to exact solutions, we have provided a FIXP-hardness result. This leaves open the question of whether the problem is FIXP-complete in this case. Indeed, our reduction from Hairy Ball to Brouwer only works for approximate solutions. A first step would be to try to reduce Hairy Ball to Borsuk-Ulam, even though no such (fully constructive) mathematical proof seems to be known.

Our results on multiple-source variants of the End-of-Line problem open the way for two new research directions. First, they provide a new tool for showing membership of PPAD, which can be used to put further problems in this class. It seems very unlikely that the Hairy Ball Theorem should be the only "natural" application of these results. Furthermore, a second interesting research direction is investigating the complexity of End-of-Line with a super-polynomial number of known sources (implicitly given in the input).

References

1 James Aisenberg, Maria Luisa Bonet, and Sam Buss. 2-D Tucker is PPA complete. Technical Report TR15-163, ECCC, 2015.

2 Stefan Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae*, 3(1):133–181, 1922.

3 Paul Beame, Stephen Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of NP search problems. *Journal of Computer and System Sciences*, 57(1):3–19, 1998.

4 W. M. Boothby. On two classical theorems of algebraic topology. *The American Mathematical Monthly*, 78(3):237–249, 1971.

5 L. E. J. Brouwer. Über Abbildung von Mannigfaltigkeiten. *Mathematische Annalen*, 71:97–115, 1912.

6 Samuel R. Buss and Alan S. Johnson. Propositional proofs and reductions between NP search problems. *Annals of Pure and Applied Logic*, 163(9):1163–1182, 2012.

7 Xi Chen and Xiaotie Deng. On the complexity of 2D discrete fixed point problem. *Theoretical Computer Science*, 410(44):4448–4456, 2009.

8 Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibria. *Journal of the ACM*, 56(3):1–57, 2009.

9 Eugene Curtin. Another Short Proof of the Hairy Ball Theorem. *The American Mathematical Monthly*, 125(5):462–463, 2018.

10 Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009.

11 Constantinos Daskalakis and Christos H. Papadimitriou. Continuous local search. In *22nd SODA*, pages 790–804. SIAM, 2011.

12 Constantinos Daskalakis, Christos Tzamos, and Manolis Zampetakis. A converse to Banach's fixed point theorem and its CLS-completeness. In *50th STOC*, pages 44–50. ACM, 2018.

13 Xiaotie Deng, Qi Qi, and Amin Saberi. Algorithmic solutions for envy-free cake cutting. *Operations Research*, 60(6):1461–1476, 2012.

14 Murray Eisenberg and Robert Guy. A Proof of the Hairy Ball Theorem. *The American Mathematical Monthly*, 86(7):571–574, 1979.

15 Kousha Etessami and Mihalis Yannakakis. On the complexity of Nash equilibria and other fixed points. *SIAM Journal on Computing*, 39(6):2531–2597, 2010.

16 John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. End of Potential Line. *arXiv preprint*, 2018. `arXiv:1804.03450`.

17 Aris Filos-Ratsikas and Paul W. Goldberg. Consensus Halving is PPA-complete. In *50th STOC*, pages 51–64. ACM, 2018.

18 Aris Filos-Ratsikas and Paul W. Goldberg. The Complexity of Splitting Necklaces and Bisecting Ham Sandwiches. *arXiv preprint*, 2018. (to appear at STOC 2019). `arXiv:1805.12559`.

19 Paul W. Goldberg and Christos H. Papadimitriou. Towards a unified complexity theory of total functions. *Journal of Computer and System Sciences*, 94:167–192, 2018.

20 Victor Guillemin and Alan Pollack. *Differential topology*. Prentice-Hall, 1974.

21 Alexandros Hollender and Paul W. Goldberg. The Complexity of Multi-source Variants of the End-of-Line Problem, and the Concise Mutilated Chessboard. Technical Report TR18-120, ECCC, 2018.

22 Tyler Jarvis and James Tanton. The Hairy Ball Theorem via Sperner's Lemma. *American Mathematical Monthly*, pages 599–603, 2004.

23 Shiva Kintali, Laura J. Poplawski, Rajmohan Rajaraman, Ravi Sundaram, and Shang-Hua Teng. Reducibility among Fractional Stability Problems. *SIAM Journal on Computing*, 42(6):2063–2113, 2013.

24 Peter McGrath. An extremely short proof of the hairy ball theorem. *The American Mathematical Monthly*, 123(5):502–503, 2016.

25 Nimrod Megiddo and Christos H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, 1991.

26 John Milnor. Analytic proofs of the "hairy ball theorem" and the Brouwer fixed point theorem. *The American Mathematical Monthly*, 85(7):521–524, 1978.

27 Christos H. Papadimitriou. On graph-theoretic lemmata and complexity classes. In *31st FOCS*, pages 794–801. IEEE, 1990.

28 Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994.

29 Henri Poincaré. Sur les courbes définies par les équations différentielles. *Journal de Mathématiques Pures et Appliquées*, 4:167–244, 1885.

$\mathrm{AC}^0[p]$ Lower Bounds Against MCSP via the Coin Problem

Alexander Golovnev
Harvard University, Cambridge, USA
alexgolovnev@gmail.com

Rahul Ilango
Rutgers University, New Brunswick, USA
rahul.ilango@rutgers.edu

Russell Impagliazzo
University of California San Diego, USA
russell@cs.ucsd.edu

Valentine Kabanets
Simon Fraser University, Burnaby, Canada
kabanets@cs.sfu.ca

Antonina Kolokolova
Memorial University of Newfoundland, St. John's, Canada
kol@mun.ca

Avishay Tal
Stanford University, USA
avishay.tal@gmail.com

Abstract

Minimum Circuit Size Problem (MCSP) asks to decide if a given truth table of an n-variate boolean function has circuit complexity less than a given parameter s. We prove that MCSP is hard for constant-depth circuits with mod p gates, for any prime $p \geq 2$ (the circuit class $\mathsf{AC}^0[p]$). Namely, we show that MCSP requires d-depth $\mathsf{AC}^0[p]$ circuits of size at least $\exp(N^{0.49/d})$, where $N = 2^n$ is the size of an input truth table of an n-variate boolean function. Our circuit lower bound proof shows that MCSP can solve the coin problem: distinguish uniformly random N-bit strings from those generated using independent samples from a biased random coin which is 1 with probability $1/2 + N^{-0.49}$, and 0 otherwise. Solving the coin problem with such parameters is known to require exponentially large $\mathsf{AC}^0[p]$ circuits. Moreover, this also implies that MAJORITY is computable by a non-uniform AC^0 circuit of polynomial size that also has MCSP-oracle gates. The latter has a few other consequences for the complexity of MCSP, e.g., we get that any boolean function in NC^1 (i.e., computable by a polynomial-size formula) can also be computed by a non-uniform polynomial-size AC^0 circuit with MCSP-oracle gates.

2012 ACM Subject Classification Theory of computation → Circuit complexity; Theory of computation → Problems, reductions and completeness

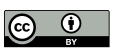
Keywords and phrases Minimum Circuit Size Problem (MCSP), circuit lower bounds, AC0[p], coin problem, hybrid argument, MKTP, biased random boolean functions

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.66

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://eccc.weizmann.ac.il/report/2019/018/`.

Funding *Alexander Golovnev*: Supported by a Rabin Postdoctoral Fellowship.
Russell Impagliazzo: Work supported by a Simons Investigator Award from the Simons Foundation.
Valentine Kabanets: Supported in part by an NSERC Discovery grant.



46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 66; pp. 66:1–66:15

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

Antonina Kolokolova: Supported in part by an NSERC Discovery grant.
Avishay Tal: Supported by a Motwani Postdoctoral Fellowship and by NSF grant CCF-1763299. Part of this work was done while the last four authors were visiting Simons Institute for the Theory of Computing.

Acknowledgements This work was partly carried out while many of the authors were visiting the Simons Institute for the Theory of Computing in association with the DIMACS/Simons Collaboration on Lower Bounds in Computational Complexity, which is conducted with support from the National Science Foundation. We also thank Chris Umans and Ronen Shaltiel for helpful discussions in the early stages of this project (during the Dagstuhl 2018 workshop on "Algebraic Methods in Complexity"). We thank Eric Allender and Shuichi Hirahara for their comments, and special thanks to Eric for pointing us to the paper of Dančik [9] and the discussion of various circuit and formula complexity measures for constant-depth circuit models. We are grateful to our anonymous reviewers for helpful comments on this paper.

1 Introduction

Minimum Circuit Size Problem (MCSP) asks to decide if a given boolean function $f\colon \{0,1\}^n \to \{0,1\}$ (presented by its truth table of length $N = 2^n$) can be computed by a boolean circuit of size at most s, for a given parameter $0 \leq s \leq 2^n$. There is no nontrivial algorithm currently known for MCSP other than the "brute force" enumeration of all circuits of size up to s and checking if any one of them computes f. On the other hand, while MCSP is obviously in NP, it is a major open question to decide if MCSP is NP-complete (and there is a growing list of research papers providing arguments for and against various NP-completeness reductions to MCSP [17, 2, 4, 3, 6, 13, 12, 24]).

Another natural question is to prove circuit lower bounds (for restricted circuit models) for MCSP. Here some results are known. Allender et al. [2] showed that MCSP requires super-polynomial-size AC^0 circuits (constant-depth circuits with AND, OR, and NOT gates). Hirahara and Santhanam [11] proved that MCSP requires almost quadratic-size De Morgan formulas.

It was an open question [5, 25] to prove that MCSP requires super-polynomial $\mathsf{AC}^0[p]$ circuits (constant-depth circuits with AND, OR, NOT and mod p counting gates), for a prime $p > 0$. We resolve this question in the present paper. Our main result is that MCSP requires d-depth $\mathsf{AC}^0[p]$ circuits of size at least $\exp(N^{0.49/d})$, where $N = 2^n$ is the size of an input truth table of an n-variate boolean function.

Previous proof methods of circuit lower bounds for MCSP

The lack of NP-completeness reductions to MCSP and the scarcity of circuit lower bounds for MCSP underscore the general phenomenon that there are very few known reductions to MCSP. The main (if not the only one) use of MCSP inside known reductions is to "break" pseudorandom function generators, an idea going back to the celebrated paper of Razborov and Rudich [28] on "natural proofs". The point is that known candidate constructions of pseudorandom function generators produce pseudorandom functions that do have "small" circuit complexity, whereas truly random functions are known to require "high" circuit complexity. Thus, an assumed efficient MCSP algorithm can distinguish between the truth tables of such pseudorandom functions and those of truly random functions, thereby "breaking" the pseudorandom function generator.

Both previously known circuit lower bounds for MCSP by Allender et al. [2] and by Hirahara and Santhanam [11] used MCSP's ability to break pseudorandom function generators together with the existence of known pseudorandom (function) generators that are provably secure against AC^0 and quadratic-size De Morgan formulas, respectively. The same approach cannot be applied to the case of $\mathsf{AC}^0[p]$ circuits as we currently do not have any strong enough pseudorandom generators secure against $\mathsf{AC}^0[p]$!

Our approach

Our approach instead is to reduce the Majority function to MCSP, and use the known $\mathsf{AC}^0[p]$ lower bounds against Majority [27, 31]. In fact, we give a reduction to MCSP from the coin problem where one is asked to distinguish between an N-bit uniformly random string, and an N-bit string sampled so that each bit is independently set to 1 with probability $1/2 - \epsilon$ (and to 0 otherwise), for some parameter $\epsilon > 0$. We then use the result of Shaltiel and Viola [29] showing that any algorithm solving such a coin problem yields an efficient algorithm for computing the Majority function on inputs of length $1/\epsilon$. To conclude super-polynomial-size $\mathsf{AC}^0[p]$ circuit lower bounds for MCSP from the known lower bounds for the Majority function, we need to be able to solve the coin problem for N-bit strings with the parameter $\epsilon < 1/\mathrm{poly}(\log N)$.

Here is some intuition why MCSP could be useful for solving the coin problem. For $N = 2^n$, an N-bit random string has binary entropy N. On the other hand, N-bit strings sampled using a biased coin with probability $p = 1/2 - \epsilon$ of being 1 would likely have close to $pN < N/2$ ones only, and so come from a smaller set of about $\binom{N}{pN} \approx 2^{H(p)\cdot N}$ strings of size N, where H is the binary entropy function. Information-theoretically, we can describe each string with at most pN ones using about $H(p) \cdot N$ bits. For $p \ll 1/2$, we have that $H(p) \cdot N \ll N$, and so most biased functions have a description of bit complexity much less than N. If somehow we could extend this information-theoretic argument to show that most biased functions will have *circuit complexity* noticeably smaller than that of random functions, we'd be done because MCSP would be able to distinguish between random functions (of higher circuit complexity) and random biased functions (of lower circuit complexity).

Lupanov in 1965 [21] proved that, indeed, biased random functions have circuit complexity smaller than that of random boolean functions. However, Lupanov's result applies only to the case of bias probability $p = 1/2 - \epsilon$ for large (close to constant) ϵ only, and doesn't give anything useful for our case of $\epsilon < 1/\mathrm{poly}(\log N)$. To circumvent the lack of tighter circuit upper bounds for slightly biased random functions, we employ two new ideas.

First, we show that the circuit complexity of q-random functions is very *tightly concentrated* around its expectation, for every probability q. This can be proved using a simple martingale argument (McDiarmid's Inequality [22]). The point is that the circuit complexity of a given n-variate boolean function f changes by at most $O(n)$ when we change the value of f on exactly one n-bit input (which can be simply hard-wired into the new circuit for the modified function).

Secondly, we use a *hybrid argument*. By Lupanov's result [21], one can show that for $p = 0.01$, almost all p-biased random functions will have circuit complexity noticeably smaller that $2^n/n$, and in particular, the *expected* circuit complexity of a p-biased random function is at most $0.1 \cdot 2^n/n$. On the other hand, for $p' = 1/2$, well-known counting arguments show that almost all such random functions have circuit complexity very close to $2^n/n$, and in particular, the expected circuit complexity of a random function is at least $0.9 \cdot 2^n/n$. Imagine we have t equally spaced probabilities between $p = 0.01$ and $p' = 1/2$, for some number t to be chosen. Then by the hybrid argument, there will exist two successive probabilities q and $q' \approx q + 1/t$, where the *expected circuit sizes* for q-random and q'-random functions differ by at least $\Omega(2^n/n)/t$.

By the "concentration around the expected circuit size" result mentioned above, we conclude that MCSP is able to distinguish between most q-random boolean functions and most $q' \approx q + 1/t$-random ones. By re-scaling, we conclude that MCSP is able to distinguish between $1/2 - 1/t$-random function and $1/2$-random ones, i.e., that MCSP solves the coin problem for the bias $\epsilon = 1/t$. Finally, our concentration result is strong enough to allow us to choose $t = 2^{\Omega(n)}$, which yields an exponential $\mathsf{AC}^0[p]$ circuit lower bound for MCSP. (To get the quantitatively strongest circuit lower bound for MCSP, we use the recent $\mathsf{AC}^0[p]$ lower bounds for the coin problem by [18], rather than apply the reduction from the Majority function to the coin problem from [29].)

Other results

We are able to generalize our $\mathsf{AC}^0[p]$ circuit lower bounds to several natural variants of MCSP. For a circuit class $\mathcal{C}$, let $\mathcal{C}$-MCSP denote the MCSP problem asking about the $\mathcal{C}$-type circuit complexity of a given truth table. For example, $\mathcal{C}$ can be the class AC^0, where we ask about the gate complexity of a smallest AC^0 circuit computing a given boolean function, or $\mathcal{C}$ can be the class of boolean formulas, where we ask about the formula (leaf) complexity of a smallest formula. We show that for both such cases of $\mathcal{C}$, the problem $\mathcal{C}$-MCSP requires exponential-size $\mathsf{AC}^0[p]$ circuits. This generalization requires us to re-visit Lupanov's general circuit upper bounds for biased random functions. We provide new "hashing-based" arguments for such circuit constructions that apply to the case of formulas as well as constant-depth circuits and formulas.

As a corollary of our reduction of the coin problem to MCSP and some previous results, we obtain that every function in NC^1 can be computed by a non-uniform AC^0 circuit with MCSP oracle gates. We also show that at least one of the following lower bounds must be true: either $\mathsf{NEXP} \not\subset \mathsf{P/poly}$, or $\mathsf{MCSP} \notin \mathsf{ACC}^0$.

Finally, we give a new coin-problem based proof of $\mathsf{AC}^0[p]$ circuit lower bounds for MKTP, a Kolmogorov-complexity variant of MCSP, re-proving the result of [5].

The rest of the paper

We give the necessary definitions and facts in Section 2. We prove the aforementioned circuit complexity concentration result for random biased functions in Section 3, and then prove our $\mathsf{AC}^0[p]$ circuit lower bound for MCSP in Section 4. In Section 5, we give corollaries of our main result. The lower bound for MKTP is given in Section 7. We generalize our lower bounds to the case of $\mathcal{C}$-MCSP in Section 6. Section 8 lists some open questions.

2 Preliminaries

2.1 Complexity basics

For a boolean function $f\colon \{0,1\}^n \to \{0,1\}$, let $size(f)$ denote the size of a smallest circuit (using AND, OR, and NOT gates) that computes f. Let $size(n)$ be the maximum of $size(f)$, over all n-variate boolean functions f. Finally, define $s_n = \mathbf{E}[size(f)]$ to be the average of circuit complexities over all n-variate boolean functions.

Below, all logarithms are base 2, unless stated otherwise.

Building on the work by Shannon [30], Lupanov [19] proved the following lower and upper bounds for $size(n)$.

▶ **Theorem 2.1** (Lupanov [19]). *For all sufficiently large $n \in \mathbb{N}$,*

$$size(n) \leq \frac{2^n}{n} + O\left(\frac{2^n \log n}{n^2}\right).$$

Moreover, all but $o(1)$ fraction of uniformly random n-variate boolean functions f require

$$size(f) \geq \frac{2^n}{n} + \Omega\left(\frac{2^n \log n}{n^2}\right).$$

For the case of n-variate functions with a fixed fraction of 1 inputs, Lupanov [21] proved the following generalization of his earlier bounds.

▶ **Theorem 2.2** (Lupanov [21]). *Let $f\colon \{0,1\}^n \to \{0,1\}$ be any boolean function that has the value 1 on $k \leq 2^{n-1}$ inputs, where $k \in \Omega(2^n)$. Then, for all sufficiently large $n \in \mathbb{N}$,*

$$size(f) \leq \frac{\log \binom{2^n}{k}}{\log \log \binom{2^n}{k}} + O\left(\frac{2^n \log n}{n^2}\right).$$

Moreover, all but $o(1)$ fraction of random such functions f require

$$size(f) \geq \frac{\log \binom{2^n}{k}}{\log \log \binom{2^n}{k}} + \Omega\left(\frac{2^n \log n}{n^2}\right).$$

2.2 Probability basics

▶ **Theorem 2.3** (McDiarmid's Inequality [22]). *Let $X_1, \ldots, X_N \in \{0,1\}$ be independent random variables. Let $f\colon \{0,1\}^n \to \mathbb{R}$ be any function such that, for some function $c = c(N)$, for all $1 \leq i \leq N$ and for all $b_1, \ldots, b_N, \widetilde{b}_i \in \{0,1\}$, it holds that*

$$\left| f(b_1, \ldots, b_N) - f(b_1, \ldots, b_{i-1}, \widetilde{b}_i, b_{i+1}, \ldots, b_N) \right| \leq c.$$

Then, for any $\lambda > 0$,

$$\mathbf{Pr}\left[|f(X_1, \ldots, X_N) - \mathbf{E}\left[f(X_1, \ldots, X_N)\right]| \geq \lambda\right] \leq 2 \cdot \exp\left(\frac{2\lambda^2}{Nc^2}\right).$$

Roughly speaking, the inequality states that with high probability the value of f will be of distance at most $O(\sqrt{N} \cdot c)$ around its mean.

2.3 Coin problem

A coin problem is the problem to distinguish between two coins (boolean-valued random variables), where one coin has probability p of being 1, and the other coin probability $q > p$. Usually, $p = 1/2 - \epsilon$ and $q = 1/2 + \epsilon$, for some $\epsilon > 0$; or, $p = 1/2 - \epsilon$ and $q = 1/2$. By a simple "translation argument", it is possible to show that all of these problems are essentially equivalent. For completeness, we state this argument next.

▷ Claim 2.4 (Translation Argument). Let $0 < p \leq q < 1$, $\varepsilon > 0$. Suppose C is a circuit of size S that solves the $p(1-\varepsilon)$ versus p coin problem on inputs of length N with advantage α. Then, there exists a circuit $\widetilde{C}$ of size at most S that solves the $q(1-\varepsilon)$ versus q coin problem on inputs of length N with advantage at least α.

Proof. For $p \in (0,1)$, we denote by μ_p the product distribution on $\{0,1\}^N$ where each bit is independently sampled to be 1 with probability p, and 0 otherwise. Let C be a circuit of size S that solves the $p(1-\varepsilon)$ versus p coin problem with advantage

$$\alpha := \mathop{\mathbf{Pr}}_{x\sim\mu_{p(1-\varepsilon)}}[C(x)] - \mathop{\mathbf{Pr}}_{x\sim\mu_p}[C(x)].$$

We construct a distribution over circuits C' that achieves the same advantage on the $q(1-\varepsilon)$ versus q coin problem. The randomized circuit C' is defined as follows: "For each input bit x_i, let $x'_i = x_i$ with probability p/q and $x'_i = 0$ otherwise. Apply C on $(x'_1, \ldots, x'_N)$." Note that, by design, if x is distributed according to a μ_q then x' is distributed according to μ_p, and if x is distributed according to $\mu_{q(1-\varepsilon)}$, then x' is distributed according to $\mu_{p(1-\varepsilon)}$. We get a distribution over circuits C' that solves the $q(1-\varepsilon)$ versus q coin problem with advantage at least α. By averaging, there must exist a deterministic circuit $\widetilde{C}$ that solves the $q(1-\varepsilon)$ versus q coin problem with advantage at least α. Note that $\widetilde{C}$ is obtained from C' by fixing the internal randomness that C' used to decide for each $1 \le i \le N$ whether to set $x'_i = x_i$ or $x'_i = 0$. With those choices fixed, $\widetilde{C}(x_1, \ldots, x_N)$ is just a restriction of $C(x_1, \ldots, x_N)$ where some x_i's are set to 0. Hence, the size of $\widetilde{C}$ is at most that of C, as claimed. ◁

▶ **Theorem 2.5** ([29]). *Let A be an algorithm that distinguishes, with constant distinguishing probability, between n-bit uniformly random strings, and n-bit strings sampled so that each bit is independently set to 1 with probability $1/2 - \epsilon$ (and to 0 otherwise). Then there is a non-uniform AC^0 circuit of size $\mathrm{poly}(n/\epsilon)$ that computes the majority function on binary inputs of length $1/\epsilon$, using A-oracle gates.*

Using the theorem above as well as the well-known lower bound for the majority function against $\mathsf{AC}^0[p]$ circuits, for any constant prime p, we can deduce that any algorithm solving the coin problem with bias ϵ on n-bit inputs requires $\mathsf{AC}^0[p]$ depth d circuits of size at least $\exp((1/\epsilon)^{1/O(d)})$. This lower bound has been recently sharpened.

▶ **Theorem 2.6** ([18]). *Let A be a boolean function that distinguishes, with constant distinguishing probability, between n-bit uniformly random strings, and n-bit strings sampled so that each bit is independently set to 1 with probability $1/2 - \epsilon$ (and to 0 otherwise). Then any depth d $\mathsf{AC}^0[p]$ circuit computing A must have size at least $\exp(\Omega((1/\epsilon)^{1/(d-1)}))$.*

3 Concentration of Circuit Complexity

For every $n \ge 1$, let μ be any product distribution over $\{0,1\}^N$, where $N = 2^n$. Recall that $size(f)$ is the size of a smallest circuit computing a boolean function f. For each integer $n \ge 1$, define

$$s_n^\mu = \mathbf{E}_{f\sim\mu}[size(f)],$$

the expectation of $size(f)$ over n-variate boolean functions f whose N-bit truth tables are sampled according to μ. We show that a random n-variate boolean function sampled from μ is likely to have its circuit complexity very close to the expected circuit complexity s_n^μ.

▶ **Theorem 3.1.** *For μ and s_n^μ as defined above, if a boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is chosen at random according to distribution μ, then, with probability at least $1 - 2^{-n}$,*

$$|size(f) - s_n^\mu| \le \sqrt{N} \cdot n^2.$$

Proof. Observe that for any two n-variate boolean functions f and g that differ on exactly one input $a \in \{0,1\}^n$, we have that $|size(f) - size(g)| \leq O(n)$. Indeed, suppose, without loss of generality, that $size(f) \leq size(g)$. Then we can compute g as follows: "Given $z \in \{0,1\}^n$, check if $z = a$. If so, then output the bit $b = g(a)$. Otherwise, use the circuit for f to output $f(z)$." More precisely, if C is a circuit computing f, then the circuit D for g can be described as

$$D(z) := (b \wedge (\wedge_{i=1}^{n}(z_i = a_i))) \vee (C(z) \wedge (\vee_{i=1}^{n}(z_i \neq a_i))), \tag{1}$$

where $(z_i = a_i)$ is defined to be z_i for $a_i = 1$, and the negation $\overline{z_i}$ for $a_i = 0$; and $(z_i \neq a_i)$ is the negation of $(z_i = a_i)$. Clearly, the size of D is that of C plus $O(n)$.

Let $X_1, \ldots, X_N \in \{0,1\}$ be independent random variables sampled from the product distribution μ. We apply McDiarmid's Inequality of Theorem 2.3 to the function $size\colon \{0,1\}^N \to \mathbb{N}$, which, as we just argued, differs by at most $c = O(n)$ on any two truth tables that agree in all but one coordinate. We get the desired concentration result by choosing $\lambda = \sqrt{N} \cdot n^2$. That is, all but $\exp(-n)$ fraction of μ-random n-variate boolean functions f have their circuit size $size(f)$ within $\sqrt{N} \cdot n^2$ of the expected circuit size s_n^μ. ◀

4 Main theorem

▶ **Theorem 4.1.** *Let $p \geq 2$ be any prime. For any depth $d > 0$ and large enough input size $N = 2^n$, MCSP on N-bit truth tables requires depth d $\mathsf{AC}^0[p]$ circuits of size $\exp(\Omega(N^{0.49/(d-1)}))$.*

Proof. Let $t = \lceil 2^{0.49n} \rceil$. Consider an arithmetic sequence of probabilities $(p_0, p_1, p_2, ..., p_t)$ with $p_0 = 0.01$, $p_t = 0.5$ and $p_i = p_0 + i \cdot 0.49/t$. For each $i = 0, \ldots, t$, let μ^i be the product distribution on $\{0,1\}^N$, where each bit is independently sampled to be 1 with probability p_i, and 0 with probability $1 - p_i$. Let

$$s^i = s_n^{\mu^i} = \mathbf{E}_{f \sim \mu^i}[size(f)].$$

By Lupanov's estimates of Theorem 2.1, we have

$$s^t \geq (1 - o(1)) \cdot \frac{2^n}{n} \geq 0.9 \cdot \frac{2^n}{n}.$$

By the Chernoff bound, almost all n-variate boolean functions sampled according to μ^t will assume the value 1 on at most $k = 0.011 \cdot N$ inputs. By Theorem 2.2, we have

$$s^0 \leq H(0.011) \cdot \frac{2^n}{n} + o\left(\frac{2^n}{n}\right) \leq 0.1 \cdot \frac{2^n}{n},$$

where $H()$ is the binary entropy function. It follows that

$$s^t - s^0 \geq 0.8 \cdot \frac{2^n}{n}.$$

This means that there must be an i such that $s^{i+1} \geq s^i + \Omega(2^{0.51n}/n)$. Let $s^* = (s^i + s^{i+1})/2$. By circuit complexity concentration given in Theorem 3.1, we get that, with high probability, n-variate random boolean functions sampled from μ^i have circuit size smaller than s^*, and those sampled from μ^{i+1} have circuit size larger than s^*. Hence, $\mathsf{MCSP}(x, s^*)$ can distinguish between μ^i and μ^{i+1}, with a constant distinguishing probability.

Finally, assume by contradiction that $\mathsf{MCSP} \in \mathbf{AC^0}[p]$ of size S and depth d. Let n be large enough, and let $N = 2^n$. Then, by fixing the second input of MCSP to s^*, we get an $\mathbf{AC^0}[p]$ circuit that on N-bit inputs solves the coin problem distinguishing between p_i and p_{i+1}. By Claim 2.4, there exists a circuit of size at most S and depth at most d that solves the $(1-\varepsilon)/2$ versus $1/2$ coin problem for $\varepsilon = 1 - p_i/p_{i+1} = \Theta(1/t) = \Theta(N^{-0.49})$. However, the latter implies by Theorem 2.6 that $S \geq \exp(\Omega(\varepsilon^{-1/(d-1)})) = \exp(\Omega(N^{0.49/(d-1)}))$. ◀

5 Consequences

Below, whenever we talk about circuit classes such as AC^0, ACC^0, and TC^0, we mean *non-uniform* circuit classes.

Using Theorem 2.5, we get the following corollary to Theorem 4.1 regarding the MAJORITY function, denoted MAJ.

▶ **Corollary 5.1.** $\mathrm{MAJ} \in (\mathsf{AC}^0)^{\mathsf{MCSP}}$.

Combined with the inclusion $\mathsf{NC}^1 \subseteq (\mathsf{TC}^0)^{\mathsf{MCSP}}$ of [25], Corollary 5.1 yields the following.

▶ **Corollary 5.2.** $\mathsf{NC}^1 \subseteq (\mathsf{AC}^0)^{\mathsf{MCSP}}$.

In fact, using the same techniques, we can prove something more general.

▶ **Theorem 5.3.** *Let $\mathcal{C} \subseteq \mathsf{P/poly}$ be any complexity class that has a complete problem under TC^0-computable reductions that is also random-self-reducible via a TC^0-computable reduction. Then we have*

$$\mathcal{C} \subseteq (\mathsf{AC}^0)^{\mathsf{MCSP}}.$$

Proof sketch. It follows from [7] that any function $f \in \mathsf{P/poly}$ has a non-uniform $(\mathsf{TC}^0)^{\mathsf{MCSP}}$ circuit C of polynomial size that agrees with f on all but an inverse polynomial fraction of inputs. If f is random-self-reducible via a TC^0 reduction, we can recover from C a new polynomial-size $(\mathsf{TC}^0)^{\mathsf{MCSP}}$ circuit computing f exactly (on all inputs). Applying Corollary 5.1 concludes the proof. ◀

As the class GapL has Determinant as a complete problem under AC^0 reductions (see [1] for a survey on logspace counting complexity classes), we get the following.

▶ **Corollary 5.4.** $\mathsf{GapL} \subseteq (\mathsf{AC}^0)^{\mathsf{MCSP}}$.[1]

The following is a non-uniform version of a similar "Karp-Lipton"-style "collapse" theorem from [14], which we state just for the class EXP.

▶ **Theorem 5.5.** *If* $\mathsf{EXP} \subseteq \mathsf{P/poly}$, *then* $\mathsf{EXP} \subseteq (\mathsf{AC}^0)^{\mathsf{MCSP}}$.

Proof. Using TC^0-computable locally list-decodable binary codes of [10], we get that EXP contains a complete language that is random-self-reducible via a TC^0-computable reduction. We then appeal to Theorem 5.3. ◀

[1] The potentially bigger class DET is the class of languages that are NC^1-Turing reducible to computing the determinant of an integer matrix [8]. It is not immediately clear if $\mathsf{DET} \subseteq (\mathsf{AC}^0)^{\mathsf{MCSP}}$. Perhaps, one can use the techniques of [5] who showed such a result for MKTP.

We do not know if MCSP is NP-complete. There is a line of research showing that MCSP (or its variants) can't be NP-complete under very restricted kinds of reductions (e.g., "local" reductions of [24]). One corollary of Theorem 5.5 is that it will be difficult to rule out non-uniform Turing AC^0 reductions from SAT to MCSP.

▶ **Corollary 5.6.** *If* $\mathsf{SAT} \notin (\mathsf{AC}^0)^{\mathsf{MCSP}}$*, then* $\mathsf{EXP} \not\subseteq \mathsf{P/poly}$.

Finally, while we don't know how to disprove that $\mathsf{MCSP} \in \mathsf{ACC}^0$, we get that at least some lower bound must be true.

▶ **Corollary 5.7.** *Either* $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$*, or* $\mathsf{MCSP} \notin \mathsf{ACC}^0$.

Proof. Towards a contradiction, suppose that both (1) $\mathsf{NEXP} \subseteq \mathsf{P/poly}$, and (2) $\mathsf{MCSP} \in \mathsf{ACC}^0$. By the Easy Witness Lemma of [15], (1) implies that $\mathsf{NEXP} = \mathsf{EXP}$. Then by Theorem 5.5, we get that $\mathsf{NEXP} \subseteq (\mathsf{AC}^0)^{\mathsf{MCSP}}$. Combining this with (2) yields that $\mathsf{NEXP} \subseteq \mathsf{ACC}^0$. But the latter contradicts the known lower bound of [32]. ◀

6 Generalizations

Here we show that, for a number of typical circuit classes $\mathcal{C}$, our lower bound proof (and a reduction from MAJORITY) works also for $\mathcal{C}$-MCSP. In particular, we will show that both AC^0-MCSP and Formula-MCSP require exponential $\mathsf{AC}^0[p]$ circuit lower bounds.

Our lower bound for MCSP used two main ingredients: (1) circuit size concentration for random (biased) boolean functions, and (2) a noticeable difference between most likely circuit sizes for uniformly random and biased boolean functions (where each bit of the truth table is 1 with a small constant probability, say 0.01).

For property (1), we note that the concentration argument only needs the Lipschitz property of a given circuit size measure, which comes from the fact that changing a boolean function on a single n-bit input may change the circuit size of the function by at most $O(n)$ additive term. This holds for virtually every reasonable circuit model, as the proof of Theorem 3.1 shows; there is a potential increase in depth for constant-depth circuits, but this can be avoided for the case where the circuit size is defined to be the total number of gates in the constant-depth circuit (see the proof of Corollary 6.2).

For property (2), we need a Shannon-style counting argument to show that most random n-variate functions have at least certain size S in a given circuit model $\mathcal{C}$, as well as a Lupanov-style argument that (most) boolean functions with very few (a small constant fraction α of) 1s have $\mathcal{C}$-circuit complexity at most some constant fraction δ of S, for some $0 < \delta < 1$ (dependent on α).

Property (2) is known for the case of boolean circuits, as implied by Lupanov's Theorem 2.2, and is known for formulas, by the work of Pippenger [26]. Moreover, it is possible to use the celebrated constructions of Lupanov, giving tight upper bounds for circuit complexity [19] and formula complexity [20] for all boolean functions, to show that biased random functions have relatively small circuits as well as small formulas (see the full version of this paper on ECCC for more details). However, we give a different argument (based on some hashing ideas) that will allow us to reduce the problem of showing small circuit complexity of a random biased boolean function to the known worst-case upper bounds for boolean function on fewer variables. Using such known worst-case upper bounds for the classes of circuits, formulas, and constant-depth circuits (counting the number of gates), we then obtain the required upper bounds for the circuit complexity of constant-biased random functions in these circuit models.

▶ **Theorem 6.1.** *Let $0 < \alpha < 1/2$ be any constant. For all but $o(1)$ fraction of α-biased n-variate random functions f, we have*

1. *circuit-size$(f) \leq O(\alpha \cdot \log 1/\alpha) \cdot \frac{2^n}{n}$,*
2. *(AC^0)-formula-size$(f) \leq O(\alpha \cdot \log 1/\alpha) \cdot \frac{2^n}{\log n}$, and*
3. *AC^0-circuit-size$(f) \leq O(\alpha \cdot \log 1/\alpha) \cdot 2^{n/2}$ (where consider the* gate complexity *of a given constant-depth circuit); moreover, the upper bound is for some fixed depth $d_0 > 0$ (independent of f).*

We omit the proof of Theorem 6.1 due to space constraints (see the full version of this paper on ECCC for more details).

Using Theorem 6.1, we conclude the following.

▶ **Corollary 6.2.** *Let $\mathcal{C}$ be the class of general circuits, or formulas, or constant-depth AC^0 circuits. For any prime $p \geq 2$ and any depth $d > 0$ and large enough input size $N = 2^n$, $\mathcal{C}$-MCSP on N-bit truth tables requires depth d $\mathsf{AC}^0[p]$ circuits of size at least $\exp(\Omega(N^{0.49/(d-1)}))$.*

Proof. As observed earlier, our lower bound proof for MCSP requires three ingredients: the Lipschitz property of the circuit complexity measure, a Shannon-style lower bound on the complexity measure for random n-variate boolean functions, and a $O(\alpha \log(1/\alpha))$ factor smaller upper bound on the complexity measure for random α-biased boolean functions (which can be made an arbitrary constant factor ϵ smaller than the corresponding Shannon-style upper bound by choosing the constant bias $\alpha > 0$ to be sufficiently small).

The Lipschitz property is easily seen to hold for both general circuits and formulas. For constant-depth circuits, where we count the number of gates, it also holds, provided the depth of our circuits is at least 3. We sketch the argument next.

We may assume that the circuit has alternating levels of ANDs and ORs, with negations on the bottom variables level. Without loss of generality, the top gate is an OR. (The other case is symmetric.) Case 1. We want to flip the value on $a \in \{0,1\}^n$ from 0 to 1. Add an AND of $x_i^{a_i}$'s, and feed this AND into the top OR gate. (Use just one extra gate.) Case 2: We want to flip from 1 to 0 on $a \in \{0,1\}^n$. Add an OR of $x_i^{1-a_i}$'s, and feed this OR into every AND-gate just one level below the top OR-gate. (Use just one extra gate.) Note that the depth doesn't change, if the original circuit is of depth $d \geq 3$.

The Shannon-style lower bounds for random n-variate functions are known for general circuits $\Omega(2^n/n)$, formulas $\Omega(2^n/(\log n))$, and constant-depth circuits $\Omega(2^{n/2})$ (see, e.g., [16]). Finally, Theorem 6.1 gives matching upper bounds for biased functions. The proof follows. ◀

7 Circuit lower bounds for MKTP via the coin problem

Here we show how to re-prove a known $\mathsf{AC}^0[p]$ circuit lower bound for MKTP [5], using the coin problem. This was the starting point in our attempt to prove an $\mathsf{AC}^0[p]$ lower bound for MCSP. For MKTP, we managed to show that biased random strings have a noticeably smaller KT complexity than that of uniformly random strings, where the bias $1/2 - \epsilon$ can be chosen for a sufficiently small ϵ so that we immediately get an $\mathsf{AC}^0[p]$ circuit lower bound for MKTP using Theorem 2.6. We provide the details next.

We first define the KT complexity [2]. Fix a universal random-access Turing machine U. The KT complexity of a string $x \in \{0,1\}^N$ is defined as the

$$\min\{|d| + t \mid \forall\, 0 \leq i \leq N+1 \;\; U^d(i) = x_i \text{ in at most } t \text{ steps}\},$$

and $x_{N+1} = \perp$. In other words, x can be computed at every position i in time at most t by a TM U that has random access to some binary string d, and we want to minimize the total sum of the length of such an auxiliary string d and the time bound t.

MKTP is then naturally defined as follows: Given $x \in \{0,1\}^N$ and a parameter s, decide if the KT complexity of x is at most s.

To prove a super-polynomial $\mathsf{AC}^0[p]$ circuit lower bound for MKTP via the coin problem approach (using Theorem 2.6), it would suffice to show that the MKTP oracle can distinguish between uniformly random N-bit strings and those where each bit is sampled, independently, with probability $1/2 - \epsilon$, for some $\epsilon \ll 1/\mathrm{poly}\log N$. We'll show how to do this for $\epsilon = N^{-\gamma}$, for some $\gamma > 0$ (in fact, for $\gamma \approx 1/6$).

What we need is to show that a random biased N-bit string can be compressed to have its KT complexity noticeably smaller than that of a uniformly random N-bit. Let $q = 1/2 - \epsilon$ be the probability for sampling each bit to be 1 in a random biased string. By standard concentration bounds, we know that a random biased N-bit string will have, with very high probability, the number of 1s very close to $K = qN$. For the simplicity of exposition, we will assume that our random biased strings have at most $K = qN$ ones in them. We then show that every N-bit string with (at most) K ones has its KT complexity much less than N, which is the lower bound on the KT complexity of a uniformly random N-bit string.

It is natural to think of an N-bit string with K ones as a subset of size K in the universe of size N. As there are exactly $\binom{N}{K}$ such subsets, the minimal bit complexity to represent any one of such subsets is $OPT = \log_2 \binom{N}{K}$. Such an information-theoretically optimal encoding of K-size subsets of the N-size universe is known, and is achieved by using the combinatorial number system where we represent each such subset by the unique number of the form

$$\binom{c_K}{K} + \cdots + \binom{c_2}{2} + \binom{c_1}{1}.$$

In more detail, suppose we have $x \in \{0,1\}^N$ where X has exactly K ones. For the base case, when $K = 0$, we output 0. For $K > 0$, we associate with x an integer number using the following recursive procedure: if $x_N = 1$, then output $\binom{N-1}{K} + R_1$, where R_1 is the recursively computed integer associated with $N-1$-bit prefix of x and the parameter $K-1$; if $x_N = 0$, then output R_0, which is the integer associated with the $N-1$-bit prefix of x and the parameter K.

Note that the final integer associated with a given K-size subset $x \in \{0,1\}^N$ has value at most $\binom{N}{K}$ (using Pascal's identity that $\binom{N}{K} = \binom{N-1}{K-1} + \binom{N-1}{K}$), and so has the optimal bit complexity. The encoding is efficient (as outlined above). The decoding is also efficient: given an integer B encoding some unknown K-size subset $x \in \{0,1\}^N$, if $B \geq \binom{N-1}{K}$, then set $x_N = 1$, and recursively decode $B - \binom{N-1}{K}$ for a $K-1$-size subset of the $N-1$-size universe; otherwise, set $x_N = 0$, and recursively decode B for a K-size subset of the $N-1$-size universe. The running time of such a decoding algorithm is clearly $\mathrm{poly}(N)$, and can be shown to be about $O(N^2)$.

For the KT complexity of a string $x \in \{0,1\}^N$ with K ones, we could define d to be the integer encoding this K-size subset, and then define $U^d(i)$ to be an algorithm that does the decoding of d to get all the bits of x. The problem with this is that the runtime to do a complete decoding of d into x is more than N, which is too much. However, the decoding we do is *global*, recovering all bits x_i simultaneously, whereas we just need to give a *local* decoding algorithm: given i, recover x_i.

Encoding

To get a locally decodable representation for our string $x \in \{0,1\}^N$, we partition x into blocks of size b (for b to be determined). For each block j, $1 \leq j \leq N/b$, let K_i be the number of ones in that block. Note that $\sum_{j=1}^{N/b} K_j = K$. Given K_j, encode each block j information-theoretically optimally as described above, using at most $\log_2 \binom{b}{K_j}$ bits. Write the resulting encodings of all blocks one after the other; add N/b pointers (of at most $\log N$ bits each) that point to the beginnings of the encodings of the blocks; add N/b numbers K_j's to the encoding. We get that the total bit size of the overall encoding of x is at most

$$O(N/b)(\log N + \log b) + \sum_{j=1}^{N/b} \lceil \log_2 \binom{b}{K_j} \rceil.$$

The latter sum is at most

$$\sum_{j=1}^{N/b} \log_2 \binom{b}{K_j} + N/b = \log_2 \prod_{j=1}^{N/b} \binom{b}{K_j} + N/b \leq \log_2 \binom{N}{K} + N/b,$$

since the number of sets with K_j ones in block j is at most the number of all subsets of $[N]$ with K ones. Overall, the encoding size is $OPT + O(N \log N/b)$.

Decoding

Given $i \in [N]$, we first figure out which block j it is in, and then decode that entire block (after looking up its number of ones in K_j and its compressed image). As discussed earlier, the decoding runs in time about $O(b^2)$.

Upper-bounding the KT complexity

To keep the KT complexity of x low, we choose b to be $N^{1/3}$. Then the KT complexity of x is at most $OPT + O(N^{2/3})$.

Finally, we show that $\mathsf{MKTP} \notin \mathsf{AC}^0[p]$ as follows. Recall that we consider random biased strings of length N where the bias probability is $q = 1/2 - \epsilon$. Let $K = qN$ be the expected number of ones in a typical biased string, and let's assume that most biased strings have at most K ones in them (for simplicity). We get that for such a biased string, its KT complexity is at most

$$\log_2 \binom{N}{K} + O(N^{2/3}) \approx H(q) \cdot N + O(N^{2/3}),$$

where H is the binary entropy function. For $q = 1/2 - \epsilon$, we can estimate $H(q) \approx 1 - O(\epsilon^2)$. Thus, to have the KT complexity of a typical q-biased N-bit string to be strictly less than N, we need $N \cdot O(\epsilon^2) - O(N^{2/3}) > 0$, which implies that we need $\epsilon > \Omega(N^{-1/6})$. This implies by Theorem 2.6 that MKTP on inputs of size N requires $\mathsf{AC}^0[p]$ circuits of depth d of size at least $\exp(\Omega(N^{1/6(d-1)}))$.

8 Open questions

We managed to find a work-around the lack of a tighter Lupanov-style upper bound on the circuit complexity of just slightly biased random functions where the bias probability is arbitrarily close to 1/2 (as opposed to the bias probability being a small constant bounded

away from 1/2). Our proof would be much more direct and constructive if we had such refinements of Lupanov's circuit upper bounds for biased boolean functions (since then we could have proceeded similarly to our proof for the MKTP case in the previous section). Can one prove such tighter circuit upper bounds?

Our current circuit lower bound applies to MCSP, but doesn't seem to apply for its average-case version, the Razborov-Rudich natural property [28]. Can one show such an extension?

Finally, we showed (Corollary 5.7) that either $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$ or $\mathsf{MCSP} \notin \mathsf{ACC}^0$. For the proof, we used the original Easy Witness Lemma of [15], the existence of random-self-reducible problems in EXP, plus the known lower bound that $\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0$ [32]. Given the new Easy Witness Lemma and the improved circuit lower bound that $\mathsf{NQP} = \mathsf{NTIME}[n^{\mathrm{poly}\log n}] \not\subseteq \mathsf{ACC}^0$ [23], it is natural to ask the following: Can we show that either $\mathsf{NQP} \not\subseteq \mathsf{P/poly}$ or $\mathsf{MCSP} \notin \mathsf{ACC}^0$? Our current proof techniques rely on the existence of a random-self-reducible problem complete for EXP, and no such problem is known for the class NQP.

References

1. Eric Allender. Arithmetic Circuits and Counting Complexity Classes. In Jan Krajicek, editor, *Complexity of Computations and Proofs, Quaderni di Matematica*, volume 13, pages 33–72. Seconda Universita di Napoli, 2004.
2. Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from Random Strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006. doi:10.1137/050628994.
3. Eric Allender and Bireswar Das. Zero Knowledge and Circuit Minimization. In *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, pages 25–32, 2014. doi:10.1007/978-3-662-44465-8_3.
4. Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. Minimizing Disjunctive Normal Form Formulas and AC^0 Circuits Given a Truth Table. *SIAM J. Comput.*, 38(1):63–84, 2008. doi:10.1137/060664537.
5. Eric Allender and Shuichi Hirahara. New Insights on the (Non-)Hardness of Circuit Minimization and Related Problems. In Kim G. Larsen, Hans L. Bodlaender, and Jean-François Raskin, editors, *42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017, August 21-25, 2017 - Aalborg, Denmark*, volume 83 of *LIPIcs*, pages 54:1–54:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.MFCS.2017.54.
6. Eric Allender, Dhiraj Holden, and Valentine Kabanets. The Minimum Oracle Circuit Size Problem. In *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, pages 21–33, 2015. doi:10.4230/LIPIcs.STACS.2015.21.
7. Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning Algorithms from Natural Proofs. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 10:1–10:24, 2016.
8. Stephen A. Cook. A Taxonomy of Problems with Fast Parallel Algorithms. *Information and Control*, 64(1-3):2–21, 1985. doi:10.1016/S0019-9958(85)80041-3.
9. Vladimir Dančik. Complexity of Boolean functions over bases of unbounded fan-in gates. *Information Processing Letters*, 57:31–34, 1996.
10. Shafi Goldwasser, Dan Gutfreund, Alexander Healy, Tali Kaufman, and Guy N. Rothblum. Verifying and decoding in constant depth. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 440–449. ACM, 2007. doi:10.1145/1250790.1250855.

11 Shuichi Hirahara and Rahul Santhanam. On the Average-Case Complexity of MCSP and Its Variants. In *CCC*, pages 7:1–7:20, 2017. doi:10.4230/LIPIcs.CCC.2017.7.

12 Shuichi Hirahara and Osamu Watanabe. Limits of Minimum Circuit Size Problem as Oracle. In *31st Conference on Computational Complexity, CCC*, pages 18:1–18:20, 2016. doi:10.4230/LIPIcs.CCC.2016.18.

13 John M. Hitchcock and A. Pavan. On the NP-Completeness of the Minimum Circuit Size Problem. In *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS*, pages 236–245, 2015. doi:10.4230/LIPIcs.FSTTCS.2015.236.

14 Russell Impagliazzo, Valentine Kabanets, and Ilya Volkovich. The Power of Natural Properties as Oracles. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, volume 102 of *LIPIcs*, pages 7:1–7:20. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.CCC.2018.7.

15 Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. Comput. Syst. Sci.*, 65(4):672–694, 2002. doi:10.1016/S0022-0000(02)00024-7.

16 Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-24508-4.

17 Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *STOC*, pages 73–79, 2000. ECCC:TR99-0045.

18 Nutan Limaye, Karteek Sreenivasaiah, Srikanth Srinivasan, Utkarsh Tripathi, and S. Venkitesh. The Coin Problem in Constant Depth: Sample Complexity and Parity Gates. *arXiv preprint*, 2018. arXiv:1809.04092.

19 Oleg B. Lupanov. On the synthesis of switching circuits. *Doklady Akademii Nauk SSSR*, 119(1):23–26, 1958. English translation in *Soviet Mathematics Doklady*.

20 Oleg B. Lupanov. Complexity of formula realization of functions of logical algebra. *Problemy Kibernetiki*, 3:782–811, 1962.

21 Oleg B. Lupanov. On a certain approach to the synthesis of control systems — the principle of local coding. *Problemy Kibernetiki*, 14:31–110, 1965. (in Russian).

22 Colin McDiarmid. *On the method of bounded differences*, pages 148—188. London Mathematical Society Lecture Note Series. Cambridge University Press, 1989. doi:10.1017/CBO9781107359949.008.

23 Cody Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime: an easy witness lemma for NP and NQP. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 890–901. ACM, 2018. doi:10.1145/3188745.3188910.

24 Cody D. Murray and R. Ryan Williams. On the (Non) NP-Hardness of Computing Circuit Complexity. *Theory of Computing*, 13(1):1–22, 2017. doi:10.4086/toc.2017.v013a004.

25 Igor C. Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *Proceedings of the 32nd Computational Complexity Conference*, page 18. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.

26 Nicholas Pippenger. Information theory and the complexity of boolean functions. *Mathematical systems theory*, 10:129–167, 1976.

27 Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Math. Notes*, 41(4):333–338, 1987.

28 Alexander A. Razborov and Steven Rudich. Natural Proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997. doi:10.1006/jcss.1997.1494.

29 Ronen Shaltiel and Emanuele Viola. Hardness Amplification Proofs Require Majority. *SIAM J. Comput.*, 39(7):3122–3154, 2010. doi:10.1137/080735096.

30 Claude E. Shannon. The synthesis of two-terminal switching circuits. *Bell System Tech. J.*, 28:59–98, 1949.

31 Roman Smolensky. Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 77–82, 1987.

32 Ryan Williams. Nonuniform ACC Circuit Lower Bounds. *J. ACM*, 61(1):2:1–2:32, 2014. doi:10.1145/2559903.

Stochastic Online Metric Matching

Anupam Gupta
Carnegie Mellon University, Pittsburgh, PA, USA
http://www.cs.cmu.edu/~anupamg/
anupamg@cs.cmu.edu

Guru Guruganesh[1]
Google Research, United States
gurug@google.com

Binghui Peng[2]
Tsinghua University, China
pbh15@mails.tsinghua.edu.cn

David Wajc
Carnegie Mellon University, Pittsburgh, PA, USA
http://www.cs.cmu.edu/~dwajc/
dwajc@cs.cmu.edu

Abstract

We study the minimum-cost metric perfect matching problem under online i.i.d arrivals. We are given a fixed metric with a server at each of the points, and then requests arrive online, each drawn independently from a known probability distribution over the points. Each request has to be matched to a free server, with cost equal to the distance. The goal is to minimize the expected total cost of the matching.

Such stochastic arrival models have been widely studied for the *maximization* variants of the online matching problem; however, the only known result for the *minimization* problem is a tight $O(\log n)$-competitiveness for the random-order arrival model. This is in contrast with the adversarial model, where an optimal competitive ratio of $O(\log n)$ has long been conjectured and remains a tantalizing open question.

In this paper, we show that the i.i.d model admits substantially better algorithms: our main result is an $O((\log\log\log n)^2)$-competitive algorithm in this model, implying a strict separation between the i.i.d model and the adversarial and random order models. Along the way we give a 9-competitive algorithm for the line and tree metrics – the first $O(1)$-competitive algorithm for any non-trivial arrival model for these much-studied metrics.

2012 ACM Subject Classification Theory of computation → Online algorithms; Mathematics of computing → Matchings and factors

Keywords and phrases stochastic, online, online matching, metric matching

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.67

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at https://arxiv.org/abs/1904.09284.

Funding *Anupam Gupta*: Supported in part by NSF awards CCF-1536002, CCF-1540541, and CCF-1617790, and the Indo-US Joint Center for Algorithms Under Uncertainty.
David Wajc: Supported in part by NSF grants CCF-1618280, CCF-1814603, CCF-1527110, NSF CAREER award CCF-1750808 and a Sloan Research Fellowship.

[1] Work done while the author was at Carnegie Mellon University.
[2] Work done while the author was visiting Carnegie Mellon University.

1 Introduction

We study the minimum-cost metric (perfect) matching problem under online i.i.d. arrivals. In this problem, we are given a fixed metric (S, d) with a server at each of the $n = |S|$ points. Then n requests arrive online, where each request is at a location that is drawn independently from a known probability distribution $\mathcal{D}$ over the points. Each such arriving request has to be matched immediately and irrevocably to a free server, whereupon it incurs a cost equal to distance of its location to this server. The goal is to minimize the total expected cost.

The minimization version of online matching was first considered in the standard adversarial setting by Khuller et al. [25] and Kalyanasundaram and Pruhs [22]; both papers showed $(2n-1)$-competitive deterministic algorithms, and proved that this was tight for, say, the star metric. After about a decade, a randomized algorithm with an $O(\log^3 n)$-competitiveness was given by Meyerson et al. [30]; this was improved to $O(\log^2 n)$ by Bansal et al. [4], which remains the best result known. (Recall that the maximization version of matching problems have been very widely studied, but they use mostly unrelated techniques.)

The competitive ratio model with adversarial online arrivals is often considered too pessimistic, since it assumes an all-powerful adversary. One model to level the playing field, and to make the model perhaps closer to practice, is to restrict the adversary's power. Two models have been popular here: the *random-order arrivals* (or *secretary*) model, and the *i.i.d.* model defined above. The random-order model is a *semi-random* model, in which the worst-case input is subjected to random perturbations. Specifically, the adversary chooses a *set* of requests, which are then presented to the algorithm in a uniformly random order. The min-cost online matching problem in this random-order model was studied by Raghvendra, who gave a tight $O(\log n)$-competitive algorithm [35]. The random-order model also captures the i.i.d. setting, so the natural goal is to get a better algorithm for the i.i.d. model. Indeed, our main result for the i.i.d. model gives exactly such a result:

▶ **Theorem 1.1** (Main Theorem). *There is an $O((\log\log\log n)^2)$-competitive algorithm for online minimum-cost metric perfect matching in the i.i.d. setting.*

Observe that the competitiveness here is better than the lower bounds of $\Omega(\log n)$ known for the worst-case and random-order models.

Matching on the Line and Trees. There has also been much interest in solving the problem for the line metric. However, getting better results for the line than for general metrics has been elusive: an $O(\log n)$-competitive *randomized* algorithm for line metrics (and for doubling metrics) was given by [18]. In the *deterministic* setting, recently Nayyar and Raghvendra [34] gave an $O(\log^2 n)$-competitive algorithm, whose competitive ratio was subsequently proven to be $O(\log n)$ by Raghvendra [36], improving on the $o(n)$-competitive algorithm of Antoniadis et al. [2]. To the best of our knowledge, nothing better is known for tree metrics than for general metrics in both the adversarial and the random-order models. Our second result for the i.i.d. model is a constant-competitive algorithm for tree metrics.

▶ **Theorem 1.2** (Algorithm for Trees). *There is a 9-competitive algorithm for online minimum-cost metric perfect matching on tree metrics in the i.i.d. setting.*

Max-Weight Perfect Matching. Recently, Chang et al. [7] presented a $1/2$-competitive algorithm for the *maximum*-weight perfect matching problem in the i.i.d. setting. We show that our algorithm is versatile, and that a small change to our algorithm gives us a maximization variant matching this factor of $1/2$. Our approach differs from that of [7], in that we match an arriving request based on the realization of free servers, while they do so based on the "expected realization". See the full version for details.

1.1 Our Techniques

Both theorems 1.1 and 1.2 are achieved by the same algorithm. The first observation guiding this algorithm is that we may assume that the distribution $\mathcal{D}$ of request locations is just the uniform distribution on the server locations. (In the full version we show how this assumption can be removed with a constant factor loss in the competitiveness.) Our algorithm is inspired by the following two complementary consequences of the uniformity of $\mathcal{D}$.

- Firstly, each of the $n-t+1$ free servers' locations at time t are equally likely to get a request in the future, and as such they should be left unmatched with equal probability. Put otherwise, we should match to them with equal probability of $1/(n-t+1)$. However, matching *any* arriving request to any free server with probability $1/(n-t+1)$ is easily shown to be a bad choice.
- So instead, we rely on the second observation: the t^{th} request is equally likely to arrive at each of the n server locations. This means we can couple the matching of free server locations with the location of the next request, to guarantee a marginal probability of $1/(n-t+1)$ for each free server to be matched at time t.

Indeed, the constraints that each location is matched at time t with probability $1/n$ (i.e., if it arrives) and each of the free servers are matched with marginal probability $1/(n-t+1)$ can be expressed as a *bipartite flow* instance, which guides the coupling used by the algorithm. Loosely speaking, our algorithm is fairly intuitive. It finds a min-cost fractional matching between the current open server locations and the expected arrivals, and uses that to match new requests. The challenge is to bound the competitive ratio – in contrast to previously used approaches (for the maximization version of the problem) it does not just try to match vertices using a fixed template of choices, but rather dynamically recomputes a template after each arrival.

A major advantage of this approach is that we understand the distribution of the open servers. We maintain the invariant that after t steps, the set of free servers form a uniform random $(n-t)$-subset of $[n]$ – the randomness being over our choices, and over the randomness of the input. This allows us to relate the cost of the algorithm in the t^{th} step to the expected cost of this optimal flow between the original n points and a uniformly random subset of $(n-t)$ of these points. The latter expected cost is just a statistic based on the metric, and does not depend on our algorithm's past choices. For paths and trees, we bound this quantity explicitly by considering the variance across edge-cuts in the tree – this gives us the proof of Theorem 1.2.

Since general metrics do not have any usable cut structure, we need a different idea for Theorem 1.1. We show that tree-embedding results can be used either explicitly in the algorithm or just implicitly in the proof, but both give an $O(\log n)$ loss. To avoid this loss, we use a different balls-and-bins argument to improve our algorithm's competitiveness to $O((\log\log n))^2)$. In particular, we provide better bounds on our algorithm's per-step cost in terms of $\mathbb{E}[OPT]$ and the expected load of the k most loaded bins in a balls and bins process, corresponding to the number of requests in the k most frequently-requested servers. Specifically, we show that $\mathbb{E}[OPT]$ is bounded in terms of the expected *imbalance* between the number of requests and servers in these top k server locations. Coupling this latter uniform k-tuple with the uniform k-tuple of free servers left by our algorithm, we obtain our improved bounds on the per-step cost of our algorithm in terms of $\mathbb{E}[OPT]$ and these bins' load, from which we obtain our improved $O((\log\log n)^2)$ competitive ratio. Interestingly, combining both balls and bins and tree embedding bounds for the per-step cost of step k (appealing to different bounds for different ranges of k) gives us a further improvement: we prove that our algorithm is $O((\log\log\log n)^2)$ competitive.

1.2 Further Related Work

I.i.d. stochastic arrivals have been studied for various online problems, e.g., for Steiner tree/forest [15], set cover [17], and k-server [9]. Closer to our work, stochastic arrivals have been widely studied in the online matching literature, though so far mostly for maximization variants. Much of this work was motivated by applications to online advertising, for which the worst-case optimal $(1 - 1/e)$-competitive ratios [24, 29, 1] seem particularly pessimistic, given the financial incentives involved and time-learned information about the distribution of requests. Consequently, many stochastic arrival models have been studied, and shown to admit better than $1 - 1/e$ competitive guarantees. The stochastic models studied for online matching and related problems, in increasing order of attainable competitive ratios, include random order (e.g., [16, 23, 27]), unknown i.i.d. – where the request distribution is unknown – (e.g., [10, 31]), and known i.i.d. (e.g., [13, 3, 6]). Additional work has focused on interpolating between adversarial and stochastic input (e.g., [11, 26]). See Mehta's survey [28] and recent work [8, 19, 21, 20, 14, 33] for more details. The long line of work on online matching, both under adversarial and stochastic arrivals, have yielded a slew of algorithmic design ideas, which unfortunately do not seem to carry over to minimization problems, nor to perfect matching problems.

As mentioned above, the only prior work for stochastic online matching with minimization objectives was the random order arrival result of Raghvendra [35]. We are hopeful that our work will spur further research in online minimum-cost perfect matching under stochastic arrivals, and close the gap between our upper bounds and the (trivial) lower bounds for the problem.

2 Our Algorithm

In this section we present our main algorithm, together with some of its basic properties. Throughout the paper we assume that the distribution over request locations is uniform over the n servers' locations. We show in the full version that this assumption is WLOG: it increases the competitive ratio by at most a constant. In particular, we show the following.

▶ **Lemma 2.1.** *Given an α-competitive algorithm $\mathrm{ALG}_{\mathcal{U}}$ for the* uniform *distribution over server locations, $\mathcal{U}$, we can construct a $(2\alpha + 1)$-competitive algorithm $\mathrm{ALG}_{\mathcal{D}}$ for any distribution $\mathcal{D}$.*

Focusing on the uniform distribution over server locations, our algorithm is loosely the following: in each round of the algorithm, we compute an optimal fractional matching between remaining free servers and remaining requests (in expectation). Now when a new request arrives, we just match the newly-arrived request according to this matching.

2.1 Notation

Our analysis will consider k-samples from the set $S = [n]$ both with and without replacement. We will set up the following notation to distinguish them:

- Let $\mathcal{I}_k$ be the distribution over k-sub-multisets of $S = [n]$ obtained by taking k i.i.d. samples from the uniform distribution over S. (E.g., $\mathcal{I}_n$ is the request set's distribution.)
- Let $\mathcal{U}_k$ be the distribution over k-subsets of S obtained by picking a uniformly random k-subset from $\binom{S}{k}$.

In other words, $\mathcal{I}_k$ is the distribution obtained by picking k elements from S uniformly *with* replacement, whereas $\mathcal{U}_k$ is *without* replacement.

For a sub-(multi)set $T \subseteq S$ of servers, let $M(T)$ denote the optimal fractional min-cost b-matching in the bipartite graph induced between T and the set of all locations S, with overall unit capacity on either side. That is, the capacity for each node in T is $1/|T|$ and the capacity for each node in S is $1/n$. So, if we denote by $d_{i,j}$ the distance between locations i and j, we let $M(T)$ correspond to the following linear program.

$$M(T) := \min \sum_{i \in T, j \in S} d_{i,j} \cdot x_{i,j} \qquad (M(\cdot))$$

$$\text{s.t.} \sum_{j \in S} x_{i,j} = \tfrac{1}{|T|} \qquad \forall i \in T$$

$$\sum_{i \in T} x_{i,j} = \tfrac{1}{n} \qquad \forall j \in S$$

$$x \geq 0$$

We emphasize that in the above LP, several servers in S (and likewise in T) may happen to be at the same point in the metric space, and hence there is a separate constraint for each such point j (and likewise i). Slightly abusing notation, we let $M(T)$ denote both the LP and its optimal value, when there is no scope for confusion.

2.2 Algorithm Description

The algorithm works as follows: at each time k, if $S_k \subseteq S$ is the current set of free servers, we compute the fractional assignment $M(S_k)$, and assign the next request randomly according to it. As argued above, since each free server location is equally likely to receive a request later (and therefore it is worth not matching it), it seems fair to leave each free server unmatched with equal probability. Put otherwise, it is only fair to match each of these servers with equal probability. Of course, matching any arriving request to a free server chosen uniformly at random can be a terrible strategy. In particular, it is easily shown to be $\Omega(\sqrt{n})$-competitive for n servers equally partitioned among a two-point metric. Therefore, to obtain good expected matching cost, we should bias servers' matching probability according to the arrived request, and in particular we should bias it according to $M(S_k)$. This intuition guides our algorithm FAIR-BIAS, and also inspires its name.

Algorithm 1 FAIR-BIAS.

$S_n \leftarrow S$. ▷ *S_k is the set of free servers, with $|S_k| = k$.*
for time step $k = n, n-1, \cdots, 1$ **do**
  compute optimal fractional matching $M(S_k)$, denoted by x^{S_k}.
  upon arrival of request $r_k = r$ **do**
    randomly choose server s from S_k, where s_i is chosen w/prob. $p_i = n \cdot x^{S_k}_{s_i,r}$.
    assign r to s.
  end event
  $S_{k-1} \leftarrow S_k \setminus \{s\}$.
end for

A crucial property of our algorithm is that the set S_k of free servers at each time k happens to be a uniformly random k-subset of S. Recall that FAIR-BIAS assigns each arriving request according to the assignment $M(S_k)$. This means that to analyze the algorithm, it suffices to relate the optimal assignment cost OPT to the optimal assignment costs for uniformly random subsets S_k, as follows.

▶ **Lemma 2.2** (Structure Lemma). *For each time k, the set S_k is a uniformly-drawn k-subset of S; i.e., $S_k \sim \mathcal{U}_k$. Consequently, the algorithm's cost is*

$$\mathbb{E}[ALG] = \sum_{k=1}^{n} \mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)].$$

Proof. The proof of the first claim is a simple induction from n down to 1. The base case of S_n is trivial. For any k-subset $T = \{s_1, \cdots, s_k\} \subseteq S$,

$$\begin{aligned}\Pr[S_k = T] &= \sum_{s \in S \setminus T} \Pr[S_{k+1} = T \cup \{s\}] \cdot \Pr[r_{k+1} \text{ assigns to } s \mid S_{k+1} = T \cup \{s\}] \\ &= (n-k) \cdot \frac{1}{\binom{n}{k+1}} \cdot \frac{1}{k+1} = \frac{1}{\binom{n}{k}},\end{aligned}$$

where the second equality follows from induction and the fact that

$$\Pr[r_{k+1} \text{ assigned to } s \mid S_{k+1} = T \cup \{s\}] = \sum_{r \in S} x_{s,r}^{S_{k+1}} = \frac{1}{k+1}.$$

To compute the algorithm's cost, we consider some set $S_k = T$ of k free servers. Since the request $r_k = r$ is chosen with probability $1/n$, following which we match it to some free server $s \in S_k$ with probability $n \cdot x_{s,r}^{S_k}$, we find that the next edge matched by the algorithm has expected cost

$$\mathbb{E}[d_{s,r_k} \mid S_k = T] = \sum_{r} \frac{1}{n} \cdot \sum_{s \in T} n \cdot x_{s,r}^{T} \cdot d_{s,r} = M(T).$$

Therefore, the expected cost of the algorithm is indeed

$$\begin{aligned}\mathbb{E}[ALG] &= \sum_{k=1}^{n} \mathbb{E}[d_{s,r_k}] = \sum_{k=1}^{n} \sum_{T \in \binom{S}{k}} \Pr_{S_k \sim \mathcal{U}_k}[S_k = T] \cdot \mathbb{E}[d_{s,r_k} \mid S_k = T] \\ &= \sum_{k=1}^{n} \sum_{T \in \binom{S}{k}} \Pr_{S_k \sim \mathcal{U}_k}[S_k = T] \cdot M(T) = \sum_{k=1}^{n} \mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)].\end{aligned}$$

◀

The structure lemma implies that we may assume from now on that the set of free servers S_k is drawn from $\mathcal{U}_k$. In what follows, unless stated otherwise, we have $S_k \sim \mathcal{U}_k$. More importantly, Lemma 2.2 implies that to bound our algorithm's competitive ratio by α, it suffices to show that $\sum_k \mathbb{E}[M(S_k)] \leq \alpha \cdot \mathbb{E}[\text{OPT}]$. This is exactly the approach we use in the following sections.

3 Bounds for General Metrics

In Section 4 we will show that algorithm FAIR-BIAS is $O(1)$-competitive for line metrics (and more generally tree metrics), by relying on variance bounds of the number of matches across tree edges in OPT and $M(S_k)$, our algorithm's guiding LP. For general metrics, if we first embed the metric in a low-stretch tree metric [12] (blowing up the expected cost of $\mathbb{E}[\text{OPT}]$ by $O(\log n)$) and run algorithm FAIR-BIAS on the obtained metric, we immediately obtain an $O(\log n)$-competitive algorithm. In fact, explicitly embedding the input metric in a tree metric is not necessary in order to obtain this result using our algorithm. By relying on an *implicit* tree embedding, we obtain the following lemma (mirroring the variance-based bound underlying our result for tree metrics). This lemma's proof is deferred to the full version.

▶ **Lemma 3.1.** $\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)] \leq \frac{O(\log n)}{\sqrt{nk}} \cdot \mathbb{E}[\text{OPT}]$.

Summing over all values of $k \in [n]$, we find that FAIR-BIAS is $O(\log n)$-competitive on general metrics. While this bound is no better than that of Raghvendra's t-net algorithm for random order arrival [35] (and therefore for i.i.d arrivals), the result will prove useful in our overall bound for our algorithm. In Sections 3.1 and 3.2, we use a different balls-and-bins argument to decrease our bounds on the algorithm's competitive ratio considerably, to $O((\log\log n))^2)$, by considering the imbalance between number of requests and servers in the top k most requested locations. (The former quantity corresponds to the load of the k most loaded bins in a balls and bins process – motivating our interest in this process.) Finally, in Section 3.3, we combine this improved bound with the one from Lemma 3.1, summing different bounds for different ranges of k, to prove our main result: an $O((\log\log\log n)^2)$ bound for our algorithm's competitive ratio.

3.1 Balls and Bins: The Poisson Paradigm

For our results, we need some technical facts about the classical balls-and-bins process.

The following standard lemma from [32, Theorem 5.10] allows us to use the Poisson distribution to approximate monotone functions on the bins. For $i \in [n]$, let X_i^m be a random variable denoting the number of balls that fall into the i^{th} bin, when we throw m balls into n bins. Let Y_i^m be independent draws from the Poisson distribution with mean m/n.

▶ **Lemma 3.2.** *Let $f(x_1, \cdots, x_n)$ be a non-negative function such that $\mathbb{E}[f(X_1^m, \cdots, X_n^m)]$ is either monotonically increasing or decreasing with m, then*

$$\mathbb{E}[f(X_1^m, \cdots, X_n^m)] \leq 2 \cdot \mathbb{E}[f(Y_1^m, \cdots, Y_n^m)].$$

A classic result states that for $m = n$ balls, the maximum bin load is $\Theta(\log n / \log\log n)$ w.h.p. (see e.g., [32]). The following lemma is a partial generalization of this result. Its proof, which relies on the Poisson approximation of Lemma 3.2, is deferred to the full version.

▶ **Lemma 3.3.** *Let n balls be thrown into n bins, each ball thrown independently and uniformly at random. Let L_j be the load of the j^{th} heaviest bin, and $N_k := \sum_{j \leq k} L_j$ be the number of balls in the k most loaded bins. There exists a constant $C_0 > 0$ such that for any $k \leq C_0 n$,*

$$\mathbb{E}[N_k] \geq \Omega\left(k \cdot \frac{\log(n/k)}{\log\log(n/k)}\right).$$

In the next lemma, whose proof is likewise deferred to the full version, we rely on a simple Chernoff bound to give a weaker lower bound for $\mathbb{E}[N_k]$ that holds for all $k \leq n/2$.

▶ **Lemma 3.4.** *For sufficiently large n and any $k \leq n/2$, we have $\mathbb{E}[N_k] \geq 1.5k$.*

3.2 Relating Balls and Bins to Stochastic Metric Matching

We now bound the expected cost incurred by FAIR-BIAS at time k by appealing to the above balls-and-bins argument; this will give us our stronger bound of $O((\log\log n)^2)$. Specifically, we will derive another lower bound for $\mathbb{E}[\text{OPT}]$ in terms of $\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)]$. In our bounds we will partition the probability space $\mathcal{I}_n$ (corresponding to n i.i.d. requests) into disjoint parts, based on $\mathbb{T}_k$, the top k most frequently requested locations (with ties broken uniformly at random). By symmetry, $\Pr[\mathbb{T}_k = T] = 1/\binom{n}{k}$ for all $T \in \binom{S}{k}$. By coupling $\mathbb{T}_k$ with $\mathcal{U}_k$, we will lower-bound $\mathbb{E}[OPT]$ by $\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)]$ times $\mathbb{E}[N_k] - k$, the expected imbalance between number of requests and servers in $\mathbb{T}_k$. Here $\mathbb{E}[N_k]$ is the expected occupancy of the k most loaded bins in the balls and bins process discussed in Section 3.1.

To relate $\mathbb{E}[OPT \mid \mathbb{T}_k = S_k]$ to $M(S_k)$, we will bound both these quantities by the cost of a min-cost perfect b-matching between S_k and $S \setminus S_k$; i.e., each vertex v has some (possibly fractional) demand b_v which is the extent to which it must be matched. To this end, we need the following simple lemma, which asserts that for any min-cost metric b-matching instance, there exists an optimal solution which matches co-located servers and requests maximally. We defer the lemma's proof, which follows from a local change argument and triangle inequality, to the full version.

▶ **Lemma 3.5.** *Let $\mathcal{I}$ be a fractional min-cost bipartite metric b-matching instance, with demand ℓ_i and r_i for the servers and requests at location i. Then, there exists an optimal solution x for $\mathcal{I}$ with $x_{ii} = \min\{\ell_i, r_i\}$ for every point i in the metric.*

We are now ready to prove our main technical lemma, lower-bounding $\mathbb{E}[OPT \mid \mathbb{T}_k = S_k]$ in terms of $M(S_k)$ and the imbalance between number of requests of the k most requested locations, N_k, and the number of servers in those locations.

▶ **Lemma 3.6.** *For all $k < n$ and $S_k \in \binom{S}{k}$, we have $\mathbb{E}[OPT \mid \mathbb{T}_k = S_k] \geq (\mathbb{E}[N_k] - k) \cdot M(S_k)$.*

Proof. Applying Lemma 3.5 to $M(S_k)$, we find that the optimal value of $M(S_k)$ is equal to that of a min-cost bipartite perfect b-matching instance with left vertices associated with S_k, each with demand $\frac{1}{k} - \frac{1}{n}$, and right vertices associated with $S \setminus S_k$, each with demand $\frac{1}{n}$.

We now turn to the meat of the proof – lower bounding $\mathbb{E}[OPT \mid \mathbb{T}_k = S_k]$. In particular, we will lower bound $\mathbb{E}[OPT \mid \mathbb{T}_k = S_k]$ by a min-cost bipartite perfect b-matching instance with left and right vertices as above (i.e., S_k and $S \setminus S_k$, respectively), but with uniform demands on both sides of at least $(\mathbb{E}[N_k] - k)/k$ and $(\mathbb{E}[N_k] - k)/(n - k)$, respectively. That is, the biregular min-cost bipartite b-matching whose cost C we showed lower bounds $M(S_k)$, but scaled by an $f \geq \frac{(\mathbb{E}[N_k] - k)}{k \cdot (1/k - 1/n)}$ factor. Before proving this lower bound on $\mathbb{E}[OPT \mid \mathbb{T}_k = S_k]$, we note that it implies our desired bound, as

$$\mathbb{E}[\mathrm{OPT} \mid \mathbb{T}_k = S_k] \geq \frac{(\mathbb{E}[N_k] - k)}{k \cdot (1/k - 1/n)} \cdot C > (\mathbb{E}[N_k] - k) \cdot C = (\mathbb{E}[N_k] - k) \cdot M(S_k).$$

It remains to lower bound $\mathbb{E}[OPT \mid \mathbb{T}_k = S_k]$ in terms of such a biregular b-matching instance.

For the remainder of this proof, for notational simplicity we denote by Ω the probability space induced by conditioning on the event $\mathbb{T}_k = S_k$. To lower bound $\mathbb{E}_\Omega[OPT]$, we will provide a fractional perfect matching $\vec{x}$ of the expected instance (in Ω), and show that $\mathbb{E}_\Omega[OPT] \geq \sum_{ij} d_{ij} \cdot x_{ij}$, while $\sum_{j \in S \setminus S_k} x_{ij} \geq (\mathbb{E}[N_k] - k)/k$ for all $i \in S_k$ and $\sum_{i \in S} x_{ij} \geq (\mathbb{E}[N_k] - k)/(n - k)$ for all $j \in S \setminus S_k$. Consequently, focusing on edges $(i, j) \in S_k \times (S \setminus S_k)$, we find that the min-cost biregular bipartite perfect b-matching above lower bounds $\sum_{i \in S_k, j \in S \setminus S_k} d_{ij} \cdot x_{ij} \leq \sum_{ij} d_{ij} \cdot x_{ij} \leq \mathbb{E}_\Omega[OPT]$. We now turn to producing an $\vec{x}$ satisfying our desired properties.

For any two locations $i, j \in S$, we let $(i, j) \in OPT$ indicate that a request in location i is served by the server in location j. Let $p_{ij} := \Pr_\Omega[(i, j) \in OPT]$. We will show how small modifications to $\vec{p}$ will yield a fractional perfect matching $\vec{x}$ as discussed in the previous paragraph. Let Y_i be the number of requests at server i. By Lemma 3.5, we know that $(i, i) \in OPT \iff Y_i \geq 1$. So, $p_{ii} = \Pr_\Omega[Y_i \geq 1]$. Consequently, if we let $\Delta_{in}(j) := \sum_{j' \in S \setminus \{j\}} p_{j'j}$ and $\Delta_{out}(j) := \sum_{j' \in S \setminus \{j\}} p_{jj'}$, we have by Lemma 3.5 that $\Delta_{in}(j) = \Pr[Y_i \geq 1]$ and $\Delta_{out}(i) = \mathbb{E}[(Y_i - 1)^+]$ for all $i \in S$. (As usual, $x^+ = \max\{x, 0\}$.) Consequently, $\Delta_{in}(j) = \Delta_{in}(j')$ and $\Delta_{out}(j) = \Delta_{out}(j')$ for all $j, j' \in S \setminus S_k$, as $[Y_j \mid \Omega]$ and $[Y'_j \mid \Omega]$ are identically distributed. Moreover, as $\sum_{j \in S \setminus S_k} (\Delta_{in}(j) - \Delta_{out}(j)) = N_k - k \geq 0$, we find that $\Delta_{in}(j) - \Delta_{out}(j) \geq 0$ for all $j \in S \setminus S_k$. Now, suppose $Y_i \geq 1$ for all $i \in S_k$

(conditioning on the complementary event is similar), we have by Lemma 3.5 that $p_{ji} = 0$ for all $i \in S_k$ and $j \in S \setminus \{i\}$. Moreover, by symmetry we have $\Delta_{out}(i) = (\mathbb{E}[N_k] - k)/k$ for all k locations $i \in S_k$. We now show how to obtain from $\vec{p}$ a fractional matching $\vec{x}$ between S_k and $S \setminus S_k$ of no greater cost than $\vec{p}$, such that $p_{jj'} = 0$ for all $j \neq j' \in S \setminus S_k$ and such that the values $\Delta_{in}(j) - \Delta_{out}(j)$ are unchanged for all $j \in S$. Consequently, all (simple) edges in the support of $\vec{x}$ go between S_k and $S \setminus S_k$, and $\Delta_{out}(i) = (\mathbb{E}[N_k] - k)/k$ for all $i \in S_k$ and $\Delta_{in}(j) = (\mathbb{E}[N_k] - k)/(n - k)$ for all $j \in S \setminus S_k$, yielding our desired lower bound on $\mathbb{E}_\Omega[OPT]$ in terms of a biregular bipartite b-matching instance.

We start by setting $\vec{x} \leftarrow \vec{p}$. While there exists a pair $j \neq j' \in S \setminus S_k$ with $x_{j'j} > 0$, we pick such a pair. As $\Delta_{in}(j) - \Delta_{out}(j) \geq 0$, there must also be some flow coming into j. We follow a sequence of edges $j_1 \leftarrow j_2 \leftarrow j_3 \leftarrow \dots$ with each $j_r \in S \setminus S_k$ and with $x_{j_r j_{r-1}} > 0$ until we either repeat some $j_r \in S \setminus$ or reach some j_r with $x_{ij_r} 0$ for some $i \in S$. (Note that one such case must happen, as $\Delta_{in}(j) - \Delta_{out}(j) \geq 0$ for all $j \in S \setminus S_k$.) If we repeat a vertex, j_r, we only consider the sequence of nodes given by the obtained cycle, $j_1 \leftarrow j_2 \leftarrow j_3 \cdots \leftarrow j_r = j_1$. Let $\epsilon = \min_r x_{j_r j_{r-1}}$ be the smallest $x_{jj'}$ in our trail. If we repeated a vertex, we found a cycle, and we decrease $x_{jj'}$ by ϵ for all consecutive j, j' in the cycle. If we found some $i \in S$ and $x_{ij_r} > 0$, we decrease all $x_{jj'}$ values along the path (including x_{ij_r}) by ϵ and increase x_{ij_1} by ϵ. In both cases, we only decrease the cost of $\vec{x}$ (either trivially, or by triangle inequality) and we do not change $\Delta_{in}(j) - \Delta_{out}(j)$ for any $j \in S$, while decreasing $\sum_{j \neq j' \in S \setminus S_k} x_{jj'}$. As the initial x-values are all rational, repeating the above terminates, with the above sum equal to zero, which implies a biregular fractional solution $\vec{x}$ as required. The lemma follows. ◀

Coupling the distribution of $\mathbb{T}_k$ and the set of k free servers, we obtain the following.

▶ **Lemma 3.7.** $\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)] \leq \mathbb{E}[\text{OPT}]/(\mathbb{E}[N_k] - k)$.

Proof. Taking expectations over $S_k \sim \mathcal{U}_k$, we obtain our claimed bound.

$$\begin{aligned}
\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)] &= \sum_{S_k \in \binom{S}{k}} \frac{1}{\binom{n}{k}} \cdot M(S_k) && \text{defn. of } \mathcal{U}_k \\
&\leq \sum_{S_k \in \binom{S}{k}} \frac{1}{\binom{n}{k}} \frac{1}{(\mathbb{E}[N_k] - k)} \cdot \mathbb{E}[\text{OPT} \mid \mathbb{T}_k = S_k] && \text{Lemma 3.6} \\
&= \frac{1}{(\mathbb{E}[N_k] - k)} \cdot \mathbb{E}[\text{OPT}]. && \Pr[\mathbb{T}_k = S_k] = \frac{1}{\binom{n}{k}}. \quad ◀
\end{aligned}$$

Plugging in the lower bounds of Lemmas 3.3 and 3.4 for the top k most loaded bins' loads, $\mathbb{E}[N_k]$, we obtain the following bounds on FAIR-BIAS's per-step cost in terms of $\mathbb{E}[OPT]$.

▶ **Lemma 3.8.** *For C_0 a constant as in Lemma 3.3, there exists a constant C such that*

$$\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)] \leq \begin{cases} C \cdot \frac{\log\log(n/k)}{k \log(n/k)} \cdot \mathbb{E}[\text{OPT}] & \text{if } k < C_0 n \\ \frac{2}{k} \cdot \mathbb{E}[\text{OPT}] & \text{if } C_0 n \leq k \leq n/2. \end{cases}$$

The following lemma allows us to leverage Lemma 3.8, as it allows us to focus on $\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)]$ for $k \leq n/2$. Its proof relies on our characterization of $M(S_k)$ in terms of a balanced b-matching instance between S_k and $S \setminus S_k$ as in the proof of Lemma 3.6, which implies that $M(S_k) \leq M(S_{n-k})$ for all $k \leq n/2$. Its proof is deferred to the full version.

▶ **Lemma 3.9.** $\sum_{k=1}^{n} \mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)] \leq 2 \cdot \sum_{k=1}^{n/2} \mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)]$.

Using our upper bound on $\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)]$ of Lemma 3.8 and summing the two ranges of $k \leq n/2$ in Lemma 3.9 we find that FAIR-BIAS is $O((\log\log n)^2)$ competitive. We do not elaborate on this here, as we obtain an even better bound in the following section.

3.3 Our Main Result

We are now ready to prove our main result, by combining our per-step cost bounds given by our balls and bins argument (Lemma 3.8) and our implicit tree embedding argument (Lemma 3.1).

▶ **Theorem 3.10.** *Algorithm* FAIR-BIAS *is* $O((\log\log\log n)^2)$*-competitive for the online bipartite metric matching problem under i.i.d arrivals on general metrics.*

Proof. By the structure lemma (Lemma 2.2) and Lemma 3.9, we have that

$$\mathbb{E}[ALG] = \sum_{k=1}^{n} \mathbb{E}_{S_k\sim\mathcal{U}_k}[M(S_k)] \le 2\cdot\sum_{k=1}^{n/2} \mathbb{E}_{S_k\sim\mathcal{U}_k}[M(S_k)]. \tag{1}$$

We use the three bounds from Lemma 3.1 and Lemma 3.8 for different ranges of k to bound the above sum. Specifically, by relying on Lemma 3.1 for $k \le n/\log^2 n$, we have that

$$\begin{aligned}\sum_{k=1}^{n/\log^2 n} \mathbb{E}_{S_k\sim\mathcal{U}_k}[M(S_k)] &\le \sum_{k=1}^{n/\log^2 n} \frac{O(\log n)}{\sqrt{nk}}\cdot\mathbb{E}[OPT] \\ &\le O\left(\sqrt{\frac{n}{\log^2 n}}\cdot\frac{\log n\cdot\mathbb{E}[OPT]}{\sqrt{n}}\right) = O(1)\cdot\mathbb{E}[OPT].\end{aligned}$$

Next, by the first bound of Lemma 3.8 applied to $k \in [n/\log^2 n, C_0 n]$, we have that

$$\begin{aligned}\sum_{k=n/\log^2 n}^{C_0 n} \mathbb{E}_{S_k\sim\mathcal{U}_k}[M(S_k)] &\le \sum_{k=n/\log^2 n}^{C_0 n} \frac{O(\log\log(n/k))}{k\cdot\log(n/k)}\cdot\mathbb{E}[OPT] \\ &\le O\left(-(\log\log(n/k))^2\Big|_{n/\log^2 n}^{C_0 n}\right)\cdot\mathbb{E}[OPT] \\ &= O((\log\log\log n)^2)\cdot\mathbb{E}[OPT].\end{aligned}$$

Finally, by the second bound of Lemma 3.8 applied to $k \ge C_0 n$, we have that

$$\sum_{k=C_0 n}^{n/2} \mathbb{E}_{S_k\sim\mathcal{U}_k}[M(S_k)] \le \sum_{C_0 n}^{n/2} \frac{2}{k}\cdot\mathbb{E}[OPT] \le O\left(\log\left(\frac{n/2}{C_0 n}\right)\right)\cdot\mathbb{E}[OPT] = O(1)\cdot\mathbb{E}[OPT].$$

Combining all three bounds with Equation (1), the theorem follows. ◀

4 A Simple $O(1)$ Bound for Tree Metrics

In this section we show the power of the structure lemma, by analyzing FAIR-BIAS on tree metrics. Recall that a *tree metric* is defined by shortest-path distances in a tree $T = (V, E)$, with edge lengths d_e. By adding zero-length edges, we may assume that the tree has n leaves, and that servers are on the leaves of the tree. For any edge e in the tree, deleting this edge creates two components $T_1(e)$ and $T_2(e)$; denote by $T_1(e)$ the component with fewer servers/leaves. Let n_e denote the number of leaves on this smaller side, $T_1(e)$. Hence $n_e \le n/2$ for all edges e.

We now lower bound $\mathbb{E}[\mathrm{OPT}]$, by considering the mean average deviation of the number of requests which arrive in $T_1(e)$ for each edge e.

▶ **Lemma 4.1.** *The expected optimal matching cost in a tree metric on* $n \ge 2$ *vertices is at least* $\mathbb{E}[\mathrm{OPT}] \ge \frac{1}{2}\cdot\sum_{e\in T} d_e\cdot\sqrt{n_e}$.

Proof. Let X_e denote the number of requests that arrive in the component with fewer leaves, $T_1(e)$. Every matching will match at least $|X_e - n_e| = |X_e - \mathbb{E}[X_e]|$ requests across the edge e (with the equality due to the uniform IID arrivals). Summing over all edges and taking expectations, we find that

$$\mathbb{E}[\text{OPT}] \geq \sum_e d_e \cdot \mathbb{E}\big[|X_e - n_e|\big] = \sum_e d_e \cdot \mathbb{E}\big[|X_e - \mathbb{E}[X_e]|\big]. \tag{2}$$

It remains to lower bound $\mathbb{E}[|X_e - \mathbb{E}[X_e]|]$, the mean average deviation of X_e. Observe that $X_e \sim \text{Bin}(n, n_e/n)$, with $n_e \in [1, n-1]$. The following probabilistic bound appears in [5, Theorem 1]:

▶ **Claim 4.2.** *Let $Y \sim Bin(n,p)$, with $n \geq 2$ and $p \in [1/n, 1-1/n]$. Then, we have both*

$$\mathbb{E}|Y - \mathbb{E}Y| \geq \text{std}(Y)/\sqrt{2},$$

(Note that convexity implies that $\mathbb{E}|Y - \mathbb{E}Y| \leq \text{std}(Y)$ holds for all distributions, so this is a partial converse.) Applying Claim 4.2 to our case, where $p = n_e/n \in [1/n, 1-1/n]$,

$$\mathbb{E}[|X_e - \mathbb{E}X_e|] \geq \text{std}(X_e)/\sqrt{2} = \sqrt{n_e(1-n_e/n)/2} \geq \sqrt{n_e/4},$$

where the second inequality follows from $n_e \leq n/2$. Combined with (2), the lemma follows. ◀

To upper bound $\mathbb{E}[M(S_k)]$, we again consider the mean average deviation of the number of requests in $T_1(e)$, but this time when drawing k *i.i.d.* samples. First, we need to bound the cost of $M(S_k)$ for a set S_k resulting from k draws *without replacement* by the cost for a multiset obtained by taking k i.i.d. draws *with replacement*.

▶ **Lemma 4.3.** *(Replacement Lemma) For all S and $k \in [|S|]$, we have*

$$\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)] \leq \mathbb{E}_{S_k \sim \mathcal{I}_k}[M(S_k)].$$

We defer the proof of this lemma to the full version, where we prove a more general statement regarding stochastic convex optimization with constraints and coefficients determined by elements of a set chosen uniformly with and without replacement. Armed with this lemma, it suffices to bound $\mathbb{E}_{S_k \sim \mathcal{I}_k}[M(S_k)]$ from above, which we do in the following.

▶ **Lemma 4.4.** $\mathbb{E}_{S_k \sim \mathcal{I}_k}[M(S_k)] \leq \sum_{e \in T} d_e \cdot \sqrt{n_e/(kn)}$.

Proof. Fix some edge e and let $T_1(e)$ be its smaller subtree, containing $n_e \leq n/2$ leaves. Let $X_e \sim \text{Bin}(k, n_e/n)$ be the random variable denoting the number of servers in $T_1(e)$ chosen in k i.i.d samples from S. For any given realization of S_k (and therefore of X_e) the fractional solution to $M(S_k)$ utilizes edges between the different subtrees of e by exactly $|X_e/k - n_e/n|$. Since this is a tree metric, we have

$$M(S_k) = \sum_{e \in T} d_e \cdot \left| \frac{X_e}{k} - \frac{n_e}{n} \right| = \sum_{e \in T} d_e \cdot \frac{1}{k} \cdot \left| X_e - \frac{k}{n} \cdot n_e \right| = \sum_{e \in T} d_e \cdot \frac{1}{k} \cdot |X_e - \mathbb{E}[X_e]|.$$

Taking expectations over S_k, and using the fact that the mean average deviation is always upper bounded by the standard deviation (by Jensen's inequality), we find that indeed

$$\begin{aligned} \mathbb{E}_{S_k \sim \mathcal{I}_k}[M(S_k)] &= \sum_{e \in T} d_e \cdot \frac{1}{k} \cdot \mathbb{E}[|X_e - \mathbb{E}[X_e]|] \leq \sum_{e \in T} d_e \cdot \frac{1}{k} \cdot \text{std}(X_e) \\ &= \sum_{e \in T} d_e \cdot \frac{1}{k} \cdot \sqrt{k \cdot \frac{n_e}{n}\left(1 - \frac{n_e}{n}\right)} \leq \sum_{e \in T} d_e \cdot \sqrt{\frac{n_e}{k \cdot n}}. \end{aligned}$$

◀

Combining the replacement lemma (Lemma 4.3) with Lemmas 4.4 and 4.1, we obtain the following upper bound on $\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)]$ in terms of $\mathbb{E}[OPT]$.

▶ **Lemma 4.5.** $\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)] \leq 2 \cdot \frac{\mathbb{E}[OPT]}{\sqrt{nk}}$.

We can now prove our simple result for tree metrics.

▶ **Theorem 4.6.** *(Tree Bound) Algorithm* FAIR-BIAS *is* 4*-competitive on tree metrics with* $n \geq 2$ *nodes, if the requests are drawn from the uniform distribution.*

Proof. We have by the structural lemma (Lemma 2.2) and Lemma 4.5 that

$$\begin{aligned}\mathbb{E}[\text{ALG}] &= \sum_{k=1}^{n} \mathbb{E}[M(S_k)] \leq \sum_{k=1}^{n} 2 \cdot \frac{\mathbb{E}[OPT]}{\sqrt{nk}} \\ &\leq 2 \cdot \frac{\mathbb{E}[OPT]}{\sqrt{n}} \cdot \left(1 + \int_{x=1}^{n} \frac{1}{\sqrt{x}} dx\right) \leq 4 \cdot \mathbb{E}[OPT]. \end{aligned}$$ ◀

The above bound holds for all $n \geq 2$ (for $n = 1$ any algorithm is trivially 1 competitive). For n large, however, our proof yields an improved asymptotic bound of $\sqrt{2} \cdot e + o(1) \approx (3.845 + o(1))$, by relying on the asymptotic counterpart of Claim 4.2 in [5, Corollary 2], $\mathbb{E}|Y - \mathbb{E}Y| \geq \text{std}(Y)/(e/2 + o(1))$. Combining Theorem 4.6 with our transshipment argument (Lemma 2.1), we obtain a 9-competitive algorithm under any i.i.d. distribution on tree metrics on $n \geq 2$ nodes, and even better than 9-competitive algorithms for large enough n.

5 Open Questions

In this work, we presented algorithm FAIR-BIAS and proved that it is $O((\log\log\log n)^2)$-competitive for general metrics, and 9-competitive for tree metrics. Perhaps the first question is whether our algorithm (or indeed any algorithm) is $O(1)$ competitive for (known or unknown) i.i.d arrivals for general metrics. Indeed, we do not know of any instances where Algorithm FAIR-BIAS's performance is worse than $O(1)$ competitive. However, it is not clear how to extend our proofs to establish an $O(1)$ competitive ratio.

Another question is the relationship between the known and unknown i.i.d. models and the random order model. The optimal competitive ratios for the various arrival models for online problems can be sorted as follows (see e.g. [28, Theorem 2.1])

$$C.R.(Adversarial) \geq C.R.(Random\ Order) \geq C.R.(Unknown\ IID) \geq C.R.(Known\ IID).$$

For the online metric matching problem the best bounds known for the above are, respectively, $O(\log^2 n)$ [4], $\Theta(\log n), O(\log n)$ (both [35]), and $O((\log\log\log n)^2)$ (this work). Given the lower bound of [35], our work implies that one or both of the inequalities in $C.R.(Random\ Order) \geq C.R.(Unknown\ IID) \geq C.R.(Known\ IID)$ is strict (and asymptotically so). It would be interesting to see which of these inequalities is strict, by either presenting a $o(\log n)$-competitive algorithm for unknown i.i.d or a $\omega((\log\log\log n)^2)$ lower bound for this model. For the line metric, we give the first constant-competitive algorithm for this well-studied metric under any non-trivial arrivals. Extending this result, and more generally understanding the exact relationships between these arrival models for this simple metric may prove useful in understanding the relationships between the different stochastic arrival models more broadly. Moreover, it would be interesting to study these questions for other combinatorial optimization problems with online stochastic arrivals.

References

1. Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1253–1264, 2011.
2. Antonios Antoniadis, Neal Barcelo, Michael Nugent, Kirk Pruhs, and Michele Scquizzato. A $o(n)$-Competitive Deterministic Algorithm for Online Matching on a Line. In *Proceedings of the 12th Workshop on Approximation and Online Algorithms (WAOA)*, pages 11–22, 2014.
3. Bahman Bahmani and Michael Kapralov. Improved bounds for online stochastic matching. In *Proceedings of the 18th Annual European Symposium on Algorithms (ESA)*, pages 170–181. Springer, 2010.
4. Nikhil Bansal, Niv Buchbinder, Anupam Gupta, and Joseph Seffi Naor. An $O(\log^2 k)$-competitive algorithm for metric bipartite matching. In *Proceedings of the 15th Annual European Symposium on Algorithms (ESA)*, pages 522–533, 2007.
5. Daniel Berend and Aryeh Kontorovich. A sharp estimate of the binomial mean absolute deviation with applications. *Statistics & Probability Letters*, 83(4):1254–1259, 2013.
6. Brian Brubach, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. New Algorithms, Better Bounds, and a Novel Model for Online Stochastic Matching. In *Proceedings of the 24th Annual European Symposium on Algorithms (ESA)*, pages 24:1–24:16, 2016.
7. Minjun Chang, Dorit S Hochbaum, Quico Spaen, and Mark Velednitsky. DISPATCH: an optimally-competitive algorithm for maximum online perfect bipartite matching with iid arrivals. In *Proceedings of the 16th Workshop on Approximation and Online Algorithms (WAOA)*, pages 149–164, 2018.
8. Ilan Reuven Cohen and David Wajc. Randomized Online Matching in Regular Graphs. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 960–979, 2018.
9. Sina Dehghani, Soheil Ehsani, MohammadTaghi Hajiaghayi, Vahid Liaghat, and Saeed Seddighin. Stochastic k-Server: How Should Uber Work? In *Proceedings of the 44th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 126:1–126:14, 2017.
10. Nikhil R Devanur, Balasubramanian Sivan, and Yossi Azar. Asymptotically optimal algorithm for stochastic adwords. In *Proceedings of the 13th ACM Conference on Electronic Commerce (EC)*, pages 388–404, 2012.
11. Hossein Esfandiari, Nitish Korula, and Vahab S. Mirrokni. Online Allocation with Traffic Spikes: Mixing Adversarial and Stochastic Models. In *Proceedings of the 16th ACM Conference on Economics and Computation (EC)*, pages 169–186, 2015.
12. Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004.
13. Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and S Muthukrishnan. Online stochastic matching: Beating $1-1/e$. In *Proceedings of the 50th Symposium on Foundations of Computer Science (FOCS)*, pages 117–126, 2009.
14. Buddhima Gamlath, Michael Kapralov, Andreas Maggiori, Ola Svensson, and David Wajc. Online Matching with General Arrivals. *arXiv preprint arXiv:1904.08255*, 2019.
15. Naveen Garg, Anupam Gupta, Stefano Leonardi, and Piotr Sankowski. Stochastic analyses for online combinatorial optimization problems. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 942–951, 2008.
16. Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 982–991, 2008.
17. Fabrizio Grandoni, Anupam Gupta, Stefano Leonardi, Pauli Miettinen, Piotr Sankowski, and Mohit Singh. Set covering with our eyes closed. *SIAM Journal on Computing (SICOMP)*, 42(3):808–830, 2013.

18 Anupam Gupta and Kevin Lewi. The online metric matching problem for doubling metrics. In *Proceedings of the 39th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 424–435, 2012.

19 Zhiyi Huang, Ning Kang, Zhihao Gavin Tang, Xiaowei Wu, Yuhao Zhang, and Xue Zhu. How to match when all vertices arrive online. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC)*, pages 17–29, 2018.

20 Zhiyi Huang, Binghui Peng, Zhihao Gavin Tang, Runzhou Tao, Xiaowei Wu, and Yuhao Zhang. Tight Competitive Ratios of Classic Matching Algorithms in the Fully Online Model. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2875–2886, 2019.

21 Zhiyi Huang, Zhihao Gavin Tang, Xiaowei Wu, and Yuhao Zhang. Online Vertex-Weighted Bipartite Matching: Beating 1-1/e with Random Arrivals. In *Proceedings of the 45th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 1070–1081, 2018.

22 Bala Kalyanasundaram and Kirk Pruhs. Online weighted matching. *Journal of Algorithms*, 14(3):478–488, 1993.

23 Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 587–596, 2011.

24 Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for online bipartite matching. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 352–358, 1990.

25 Samir Khuller, Stephen G Mitchell, and Vijay V Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science (TCS)*, 127(2):255–267, 1994.

26 Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Allocating online advertisement space with unreliable estimates. In *Proceedings of the 8th ACM Conference on Electronic Commerce (EC)*, pages 288–294, 2007.

27 Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 597–606, 2011.

28 Aranyak Mehta. Online matching and ad allocation. *Foundations and Trends® in Theoretical Computer Science*, 8(4):265–368, 2013.

29 Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5):22, 2007.

30 Adam Meyerson, Akash Nanavati, and Laura Poplawski. Randomized online algorithms for minimum metric bipartite matching. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 954–959, 2006.

31 Vahab S Mirrokni, Shayan Oveis Gharan, and Morteza Zadimoghaddam. Simultaneous approximations for adversarial and stochastic online budgeted allocation. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1690–1701, 2012.

32 Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge university press, 2005.

33 Joseph Seffi Naor and David Wajc. Near-optimum online ad allocation for targeted advertising. *ACM Transactions on Economics and Computation (TEAC)*, 6(3-4):16, 2018.

34 Krati Nayyar and Sharath Raghvendra. An input sensitive online algorithm for the metric bipartite matching problem. In *Proceedings of the 58th Symposium on Foundations of Computer Science (FOCS)*, pages 505–515, 2017.

35 Sharath Raghvendra. A robust and optimal online algorithm for minimum metric bipartite matching. In *Proceedings of the 19th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, volume 60, 2016.

36 Sharath Raghvendra. Optimal Analysis of an Online Algorithm for the Bipartite Matching Problem on a Line. In *Proceedings of the 34th Symposium on Computational geometry (SoCG)*, pages 67:1–67:14, 2018.

Constructions of Maximally Recoverable Local Reconstruction Codes via Function Fields

Venkatesan Guruswami
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
venkatg@cs.cmu.edu

Lingfei Jin
Shanghai Key Laboratory of Intelligent Information Processing, School of Computer Science, Fudan University, Shanghai, China
Shanghai Institute of Intelligent Electronics & Systems, Shanghai, China
Shanghai Bolckchain Engineering Research Center, Fudan University, Shanghai 200433, China
lfjin@fudan.edu.cn

Chaoping Xing
School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
xingcp@ntu.edu.sg

Abstract

Local Reconstruction Codes (LRCs) allow for recovery from a small number of erasures in a local manner based on just a few other codeword symbols. They have emerged as the codes of choice for large scale distributed storage systems due to the very efficient repair of failed storage nodes in the typical scenario of a single or few nodes failing, while also offering fault tolerance against worst-case scenarios with more erasures. A maximally recoverable (MR) LRC offers the best possible blend of such local and global fault tolerance, guaranteeing recovery from all erasure patterns which are information-theoretically correctable given the presence of local recovery groups. In an (n, r, h, a)-LRC, the n codeword symbols are partitioned into r disjoint groups each of which include a local parity checks capable of locally correcting a erasures. The codeword symbols further obey h heavy (global) parity checks. Such a code is maximally recoverable if it can correct all patterns of a erasures per local group plus up to h additional erasures anywhere in the codeword. This property amounts to linear independence of all such subsets of columns of the parity check matrix.

MR LRCs have received much attention recently, with many explicit constructions covering different regimes of parameters. Unfortunately, all known constructions require a large field size that is exponential in h or a, and it is of interest to obtain MR LRCs of minimal possible field size. In this work, we develop an approach based on function fields to construct MR LRCs. Our method recovers, and in most parameter regimes improves, the field size of previous approaches. For instance, for the case of small $r \ll \varepsilon \log n$ and large $h \geqslant \Omega(n^{1-\varepsilon})$, we improve the field size from roughly n^h to $n^{\varepsilon h}$. For the case of $a = 1$ (one local parity check), we improve the field size quadratically from $r^{h(h+1)}$ to $r^{h\lfloor (h+1)/2 \rfloor}$ for some range of r. The improvements are modest, but more importantly are obtained in a unified manner via a promising new idea.

2012 ACM Subject Classification Mathematics of computing → Coding theory

Keywords and phrases Erasure codes, Algebraic constructions, Linear algebra, Locally Repairable Codes, Explicit constructions

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.68

Category Track A: Algorithms, Complexity and Games

Related Version A full version of this paper is posted at `https://arxiv.org/abs/1808.04539`.

Funding *Venkatesan Guruswami*: This research is supported in part by NSF grants CCF-1422045, CCF-1563742 and CCF-1814603.
Lingfei Jin: This research is supported by the National Natural Science Foundation of China under Grant 11871154.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 68; pp. 68:1–68:14

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

Chaoping Xing: This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Strategic Capability Research Centres Funding Initiative; and the Singapore MoE Tier 1 grants RG25/16 and RG21/18.

1 Introduction

Interest in erasure codes has surged in recent years, with the demands of massive cloud storage systems raising hitherto unexplored, yet very natural and mathematically deep, questions concerning the parameters, robustness, and efficiency of the code. Distributed storage systems need to build in redundancy in the data stored in order to cope with the loss or inaccessibility of the data on one or more storage nodes. Traditional erasure codes offer a natural strategy for such robust data storage, with each storage node storing a small part of the codeword, so that the data is protected against multiple node failures. In particular, MDS codes such as Reed-Solomon codes can operate at the optimal storage vs. reliability trade-off – for a given amount of information to be stored and available storage space, these codes can tolerate the maximum number of erasures without losing these stored information.

Individual storage nodes in a large scale system often fail or become unresponsive. Reconstruction (repair) of the content stored on a failed node with the help of remaining active nodes is important to reinstate the system in the event of a permanent node failure, and to allow access to the data stored on a temporarily unavailable node. The use of erasure codes in large storage systems, therefore, brings to the fore a new requirement: the ability to *very efficiently* reconstruct *parts* of a codeword from the rest of the codeword.

Local Reconstruction Codes (LRCs), introduced in [7], offer an attractive way to meet this requirement. An LRC imposes local redundancies in the codewords, so that a single (or a small number of) erased symbol can be recovered locally from less than r other codeword symbols.[1] Here r is the locality parameter that is typically much smaller than the code length n. In the distributed storage context, an LRC allows for the low-latency repair of any failed node as one only needs to wait for the response from r nodes. LRCs have found spectacular practical applications with their use in the Windows Azure storage system [12].

The challenge in an LRC design is to balance the locality requirement, that allows fast recovery from a single or few erasures, with good global erasure-resilience (via traditional slower methods) for more worst-case scenarios. One simple metric for global fault tolerance is the minimum distance d of the code, which means that any pattern of fewer than d erasures can be corrected. The optimal trade-off between the distance, redundancy, and locality of an LRC was established in [8], and an elegant sub-code of Reed-Solomon codes meeting this bound was constructed in [17].

This work concerns a much stronger requirement on global fault-tolerance, called *Maximal Recoverability*. This requires that the code should simultaneously correct every erasure pattern that is information-theoretically possible to correct, given the locality conditions imposed on the codeword symbols. Let us describe it more formally in the setting of interest in this paper. Define an $(n, r, h, a)_\ell$-LRC to be a linear code over $\mathbb{F}_\ell$ of length n whose n codeword symbols are partitioned into r disjoint groups each of which includes a local parity checks capable of locally correcting a erasures. The codeword symbols further obey h heavy (global) parity checks. With this structure of parity checks, it is not hard to see that the erasure patterns one can hope to correct are precisely those which consist of up to a erasures per local group plus up to h additional erasures anywhere in the codeword.

[1] LRCs are also expanded as Locally Repairable Codes or Locally Recoverable Codes, eg. [16, 17, 10].

A *maximal recoverable* (MR) LRC is a *single* code that is capable of simultaneously correcting *all* such patterns. Thus, an MR code gives the most bang-for-the-buck for the price one pays for locality.

This notion was introduced in [2] motivated by applications to storage on solid-state devices, where it was called partial MDS codes. The terminology maximally recoverable codes was coined in [7], and the concept was more systematically studied in [7, 6]. By picking the coefficients of the heavy parity checks randomly, it is not hard to show the existence of MR LRCs over *very large* fields, of size exponential in h. An explicit construction over such large fields was also given in [7], which also proved that random codes *need* such large field sizes with high probability.[2]

Since encoding a linear code and decoding it from erasures involve performing numerous finite field arithmetic operations, it is highly desirable to have codes over small fields (preferably of characteristic 2). Obtaining MR LRCs over finite fields of minimal size has therefore emerged as a central problem in the area of codes for distributed storage. So far, no construction of MR LRCs that avoid the exponential dependence on h has been found. A recent lower bound shows that, unlike MDS codes, for certain parameter settings one cannot have MR LRCs over fields of linear size. This shows that the notion of maximal recoverability is quite subtle, and pinning down the optimal field size is likely a deep question. There remains a large gap between the upper and lower bounds on field size of MR LRCs, closing which is a challenge of theoretical and practical importance.

In this work, we develop a novel approach to construct MR LRCs based on function fields. Our framework recovers and in fact slightly improves most of the previous bounds in the literature in a unified way. We note that since there are at least three quantities of significance – the locality r, the local (intra group) erasure tolerance a, and number of global parity checks h – the landscape of parameters and different constructions in this area is quite complex. Also, depending on the motivation, the range of values of interest of these parameters might be different. For example, if extreme efficiency of local repair is important, r should be small. But on the other hand this increase the redundancy and thus storage requirement of the code, so from this perspective a modest r (say $\sqrt{n}$) might be relevant. If good global fault tolerance is required, we want larger h, but then the constructions have large field size. It is therefore of interest to study the problem treating these as independent parameters, without assumptions on their relative size. We next review the field size of previous constructions, and then turn to the parameters we achieve in different regimes.

1.1 Known field size bounds

For $a \in \{0, r-1\}$, optimal maximally recoverable local reconstruction codes (MR LRCs, for short) can be constructed by using either Reed-Solomon codes or their repetition. For $h \leqslant 1$, constructions of MR LRCs over fields of size $O(r)$ were given in [2]. For the remaining case: $1 \leqslant a \leqslant r-2$ and $h \geqslant 2$, there are quite a number of constructions in literature [1, 2, 3, 4, 5, 7, 6, 9, 11, 18].

For the cases of $h = 2$ and $h = 3$, the best known constructions of MR LRCs were given in [9] with field sizes of $O(n)$ and $O(n^3)$ respectively, uniformly for all r, a. (Their field sizes were worse by $n^{o(1)}$ factors compared to these bounds when the field is required to be of

[2] This is akin to what happens for random codes to have the MDS property. However, for MDS codes, the Vandermonde construction achieves a linear field size explicitly.

characteristic 2.) For most other parameter settings, the best constructions by [5] provide a family of MR LRCs over fields of sizes

$$\ell = O\left(r \cdot n^{(a+1)h-1}\right) \tag{1}$$

as well as

$$\ell = \max\left\{O(\frac{n}{r}),\ O(r)^{h+a}\right\}^h\ , \tag{2}$$

The bound (1) outperforms the bound (2) when $r = \Omega(n)$, while the bound (2) is better when $r \ll n$. In both the bounds, the field size grows exponentially with h and a.

Recently, by using maximum rank distance (MRD) codes, the paper [15] (specifically Corollary 14) gives a family of MR LRCs over fields of sizes

$$\ell = O\left(r^{\frac{n(r-a)}{r}}\right). \tag{3}$$

When $r = \Omega(n)$, and a is close to r or h is large, (3) is better than bounds (1) or (2). By using probabilistic arguments, the paper [15] shows existence of a family of MR LRCs over fields of sizes

$$\ell = O\left(\binom{n-1}{k-1}\right), \tag{4}$$

where $k = n\left(1 - \frac{a}{r}\right) - h$ is the dimension of the code.

On the other hand, a lower bound on the field size was presented in [9]. Stating the bound when $h \leqslant \frac{n}{r}$ for simplicity, they show that the field size ℓ of an $(n, r, h, a)_\ell$ MR LRC must obey

$$\ell = \Omega_{a,h}\left(n \cdot r^{\min\{a,h-2\}}\right). \tag{5}$$

The lower bound (5) is still quite far from the upper bounds (1) and (2). In particular, the exponent a or h is to the base growing with n in the known constructions, but only to the base r in the above lower bound. Thus, one can conjecture that there is still room to improve both the constructions and the lower bounds. We note that under more complex structural requirements on the local groups, notably grid-like topologies and product codes, the optimal field size has been pinned down to $\exp(\Theta(n))$ [13].

Several techniques have been employed in literature for constructions of MR LRCs. One prevalent idea is to use a "linearized" version of the Vandermonde matrix, where the heavy parity check part of the matrix consists of columns $(\alpha_i, \alpha_i^q, \ldots, \alpha_i^{q^{h-1}})^T$ where $\alpha_i \in \mathbb{F}_\ell$ for a sufficiently high degree extension field $\mathbb{F}_\ell$ of $\mathbb{F}_q$. This construction is combined with $2h$-wise independent spaces to get an $O(n^h)$ field size in [7], and is also employed in [5]. Another approach is based on rank-metric codes (see, for instance, [4, 15]). Various ad hoc methods have been employed for good constructions of MR LRCs for small h, for example for $h = 2, 3$ in [9].

1.2 Our results

In this work, we develop a new approach to construct MR LRCs based on algebraic function fields. We discuss the key elements underlying our strategy in Section 1.4, but for now state the field sizes of the MR LRCS we can construct for various regimes of parameters. Most of the existing results in literature can be recovered through our methods in a unified way.

In most regimes, the parameters of our codes beat the known ones. For easy reference, we summarize the different possible trade-offs we can achieve in one giant theorem statement below. Since this comprehensive statement may be overwhelming to parse, let us highlight just two of our significant improvements: item (i) for $a = 1$, where we improve r^{h+1} term in (2) quadratically to $r^{\lfloor \frac{h+1}{2} \rfloor}$, and item (vi) for sufficiently large h, where the exponent h in bounds (1) and (2) is improved to εh. Also the exponent h is replaced by $\min\{h, n/r\}$ in the bounds (i)-(iv) that improve (2). In the bounds (vii) and (viii) the factor n/r in the exponent is improved to $\min\{k, n/r\}$; this improvment is less significant as it only applies to the low-rate setting but included for completeness and also to reflect a construction approach based on generator matrices (as opposed to parity check matrices which is a more potent way to reason about MR LRCs that underlies the other parts of the theorem).

▶ **Theorem 1.** *One has a maximally recoverable $(n, r, h, a)_\ell$-local reconstruction code over a field of size ℓ with parameters satisfying any of the following conditions. (Below $\tilde{O}(f)$ denotes $f \log^{O(1)} f$.)*

(i) (see Theorem 10) $a = 1, r \geqslant h + 2$ *and*

$$\ell \leqslant \left(\max \left\{ \tilde{O}(\frac{n}{r}), (2r)^{\lfloor \frac{h+1}{2} \rfloor} \right\} \right)^{\min\{h, \frac{n}{r}\}} \text{ and } \ell \text{ is even;}$$

(ii) (see Theorem 11) $a = 1$ *and*

$$\ell \leqslant \left(\max \left\{ \tilde{O}(\frac{n}{r}), 2^r \right\} \right)^{\min\{h, \frac{n}{r}\}} \text{ and } \ell \text{ is even;}$$

(iii) (see Theorem 13) *for all settings of n, r, h, a and*

$$\ell \leqslant \left(\max \left\{ \tilde{O}(\frac{n}{r}), (2r)^{h+a} \right\} \right)^{\min\{h, \frac{n}{r}\}};$$

(iv) (see Theorem 14) *for all settings of n, r, h, a and*

$$\ell \leqslant \left(\max \left\{ \tilde{O}(\frac{n}{r}), (2r)^{r} \right\} \right)^{\min\{h, \frac{n}{r}\}};$$

(v) (see Theorem 17) $r = O\left(\frac{\log n}{\log \log n}\right)$ *and* $hr \geqslant \Omega\left(\frac{n^{\frac{2}{3}}}{\varepsilon}\right)$ *for a positive real* $\varepsilon \in (0, 0.5)$ *and*

$$\ell \leqslant O\left(n^{\frac{2h}{3}(1+\varepsilon)}\right);$$

(vi) (see Theorem 18) $r = O\left(\frac{\varepsilon \log n}{\log \log n}\right)$ *and* $hr = \Omega\left(n^{1-\varepsilon}\right)$ *for a positive real* $\varepsilon \in (0, 0.5)$ *and*

$$\ell \leqslant n^{\varepsilon h};$$

(vii) (see Theorem 5) *for all settings of n, r, h, a*

$$\ell \leqslant \begin{cases} 2^{\min\{rk, n\}} \leqslant 2^n & \text{if } r \geqslant \log n \\ 2^{\lceil \log n \rceil \min\{k, \frac{n}{r}\}} & \text{if } r \leqslant \log n \end{cases}$$

where $k = \left(1 - \frac{a}{r}\right) - h$ is the dimension of the code;

(viii) (see Theorem 7) $r - a = \Omega(\log n)$ *and*

$$\ell \leqslant 2r^{\lfloor \frac{r-a}{2} \rfloor \min\{k, \frac{n}{r}\}} \text{ and } \ell \text{ is even.}$$

The first two bounds, and the bounds in (vii) and (viii) of Theorem 1 are derived from the rational function field $\mathbb{F}_2(x)$. In addition, bounds in (i) and (viii) of Theorem 1 are obtained via a combination with binary BCH codes. Bounds in (iii) and (iv) of Theorem 1 are derived from rational function field $\mathbb{F}_q(x)$, where ℓ is a power of q. The fifth bound is obtained via Hermitian function fields, while the sixth bound is derived from the Garcia-Stichtenoth function field tower. Our codes achieving the trade-offs stated in the above theorem can in fact be explicitly specified. But we note that for MR codes even existence questions over small fields are interesting and non-trivial.

1.3 Comparison

Each of our bounds in Theorem 1 beats the known results in some parameter regimes. Let us compare them one by one.

- The bound in Theorem 1(i) outperforms the bound (2) due to the quadratically better exponent for r.
- The bound in Theorem 1(ii) outperforms even the bound in Theorem 1(i) for $\frac{r}{\log r} < \left\lfloor \frac{h+1}{2} \right\rfloor$.
- The bound in Theorem 1(iii) outperforms the bound (2) for $h > \frac{n}{r}$.
- The bound in Theorem 1(iv) even outperforms the bound in Theorem 1(iv) for $r < h + a$, and hence it beats the bound (2) for $\frac{n}{h} < r < h + a$.
- The bound in Theorem 1(v) outperforms both the bounds (1) and (2) for all parameter settings subject to $r = \widetilde{O}(\log n)$ and $hr = \Omega\left(\frac{n^{\frac{2}{3}}}{\varepsilon}\right)$. It is clear that the bound in Theorem 1(v) is better than (1). As $r = \widetilde{O}(\log n)$, then we have $\left(\frac{n}{r}\right)^h > n^{h(1-o(1))} > n^{2h(1+\varepsilon)/3}$ and hence the bound in Theorem 1(v) beats (2) in this case.
- As the bound in Theorem 1(vi) is even better than the bound in Theorem 1(v), the bound in Theorem 1(vi) beats both the bounds (1) and (2) for all parameter settings subject to $r = \widetilde{O}(\varepsilon \log n)$ and $hr = O\left(n^{1-\varepsilon}\right)$ for a positive real $\varepsilon \in (0, 0.5)$.
- When the dimension k is much smaller than n, then the probabilistic bound (4) gives the field size $O(n^k) = O(2^{k \log n})$ which is the same size as in Theorem 1(vii) for $r \leqslant \log n$. When the dimension k is proportional to n, then the probabilistic bound (4) gives the field size $2^{O(n)}$ which is the same as the bound 2^n in Theorem 1(vii) for $r \geqslant \log n$.
- Finally, the bound in Theorem 1(viii) clearly outperforms the bound (3) when $k < n/r$.

1.4 Our techniques

Note that construction of MR LRCs is equivalent to construction of certain generator or parity-check matrices with requirement of column linear independence (see Section 2.1).

Our construction idea departs from previous approaches and is based on function fields over a finite field $\mathbb{F}_q$. The key in constructing an MR LRC is the choice of the heavy parity checks. We now briefly describe our idea to pick these. We associate with each of the $g = n/r$ local groups a distinguishing (high degree) place P_i, $1 \leqslant i \leqslant g$. The degree of the place is chosen large enough to guarantee the existence of at least g such places. For each local group, we pick functions f_{ij}, $1 \leqslant j \leqslant r$, that have *exactly one pole at* P_i. The coefficients of the h heavy parity checks corresponding to the j'th symbol of i'th local group are chosen to be

$$(f_{ij}(Q), f_{ij}^q(Q), \ldots, f_{ij}^{q^{h-1}}(Q))^T \ , \tag{6}$$

where Q is a place of sufficiently high degree, so that the evaluations $f_{ij}(Q)$ belong to an extension field $\mathbb{F}_\ell$ which will be the final alphabet size of the MR LRC. By properties of the Moore determinant (Section 2.2) and the large degree of Q, the required linear independence

of columns such as (6) over $\mathbb{F}_\ell$ reduces to a certain linear independence requirement for the f_{ij}'s over $\mathbb{F}_q$. Across different local groups such linear independence follows because a function with one pole at P_i cannot cancel a function with one pole at a different place $P_{i'}$. Within a local group, the required linear independence is ensured by choosing the f_{ij}'s within a group so that any $h+a$ of them (which is the maximum number of erasures we can have within a group) are linearly independent over $\mathbb{F}_q$.

We remark that all our various guarantees of Theorem 1 except Parts (v) and (vi) are obtained using just the rational function field, and can be described in elementary language using just polynomials, as we do in Section 3.

1.5 Organization

The paper is organized as follows. In Section 2, we introduce some preliminaries such as MR LRCs (both the generator and parity check matrix viewpoints) and Moore determinants. In Section 3, we present our constructions of MR LRCs using the rational function field together with a concatenation with classical codes of good rate vs. distance trade-off. We give two constructions, using the generator matrix viewpoint in the first part (yielding Parts (vii) and (viii) of Theorem 1), and then a parity check based construction in the second part which yields Parts (i)-(iv) of Theorem 1. This section is elementary and only uses properties of polynomials. In Section 4, we generalize the construction of MR LRCs via parity-check matrix given in Section 3 by making use of arbitrary algebraic function fields. We then apply this construction to Hermitian function fields and the Garcia-Stichtenoth tower to obtain MR LRCs promised in Parts (v) and (vi) of Theorem 1 respectively.

2 Preliminaries

2.1 Maximally recoverable local reconstruction codes

Throughout this paper, $\mathbb{F}_q$ denotes the finite field of q elements for a prime power q. We use $\mathbb{F}_q^{k\times n}$ to denote the set of all $k\times n$ matrices over $\mathbb{F}_q$.

Consider a distributed storage system where there are g disjoint locality groups and each group has size r and can locally correct any a erasure errors. In addition, the system can correct any h erasure errors together with any a erasure errors in each group. This requires a class of codes called *maximally recoverable local reconstruction codes* or *partial MDS codes* for error correction of such a system. The precise definition of MR LRCs is given below.

▶ **Definition 1.** Let ℓ be a prime power and let a, g, r, h be positive integers satisfying $ga+h<gr$. Put $n=gr$ and $k=n-ga-h$. An ℓ-ary $[n,k]$-linear code with a generator matrix of the form

$$G=(B_1|B_2|\cdots|B_g)\in\mathbb{F}_\ell^{k\times n}$$

is called a maximally recoverable $(n,r,h,a)_\ell$-local reconstruction code (or an MR $(n,r,h,a)_\ell$-LRC, for short) if

- **(i)** each B_i has size $k\times r$;
- **(ii)** the row span of each B_i is an $[r,r-a,a+1]_\ell$-MDS code for $1\leqslant i\leqslant g$ (note that B_i is not a generator matrix of this MDS code in general);
- **(iii)** after puncturing a columns from each B_i, the remaining matrix of G generates an $[n-ga,k,h+1]_\ell$-MDS code.

From the definition, an MR $(n, r, h, a)_q$-LRC can correct h erasure errors at arbitrarily positions together with any a erasure errors in each of g groups. To see the recovery procedure for an MR $(n, r, h, a)_\ell$-LRC, we first recover h erasure errors (this can be done from (iii) of Definition 1). We can then correct a errors from each block by (ii) of Definition 1.

The following lemma directly follows from Definition 1.

▶ **Lemma 2.** *A matrix $G = (B_1|B_2|\cdots|B_g) \in \mathbb{F}_\ell^{k\times n}$ is a generator matrix of an MR $(n, r, h, a)_\ell$-LRC if and only if every $k \times k$ submatrix S of G with at most $r - a$ columns per block B_i is invertible.*

One can have an equivalent definition via parity-check matrix.

▶ **Definition 2.** Let ℓ be a prime power and let a, g, r, h be positive integers satisfying $ga + h < gr$. Put $n = gr$ and $k = n - ga - h$. An ℓ-ary $[n, k]$-linear code with a parity-check matrix of the form

$$H = \left(\begin{array}{c|c|c|c} A_1 & O & \cdots & O \\ \hline O & A_2 & \cdots & O \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline O & O & \cdots & A_g \\ \hline D_1 & D_2 & \cdots & D_g \end{array}\right) \in \mathbb{F}_\ell^{(n-k)\times n} \tag{7}$$

is called an MR $(n, r, h, a)_\ell$-LRC if

- **(i)** each A_i has size $a \times r$ and each D_i has size $h \times r$;
- **(ii)** each A_i generates an $[r, a, r - a + 1]_\ell$-MDS code for $1 \leqslant i \leqslant g$ (note that the nullspace of A_i is $[r, r - a, a + 1]_\ell$ code);
- **(iii)** every $ag + h$ columns consisting of any a columns in each group and other arbitrary h columns are $\mathbb{F}_\ell$-linearly independent.

▶ Remark 1.

- **(i)** To see equivalence between Definitions 1 and 2, we note that each A_i in Definition 2 is actually a parity-check matrix of the code generated by B_i given in Definition 1.
- **(ii)** In this paper, we will use both Definitions 1 and 2 for constructions of MR LRCs. However, the major results of this paper come from the constructions based one Definition 2, i.e., via parity-check matrices of the required form in (7).

2.2 Moore determinant

Let ℓ be a power of q. For elements $\alpha_1, \ldots, \alpha_h \in \mathbb{F}_\ell$, the Moore matrix is defined by

$$M = \begin{pmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_h \\ \alpha_1^q & \alpha_2^q & \cdots & \alpha_h^q \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{q^{h-1}} & \alpha_2^{q^{h-1}} & \cdots & \alpha_h^{q^{h-1}} \end{pmatrix} \in \mathbb{F}_\ell^{h\times h}.$$

The determinant $\det(M)$ is given by the following formula

$$\det(M) = \prod_{(c_1,\ldots,c_h)} (c_1\alpha_1 + \cdots + c_h\alpha_h),$$

where $(c_1, \ldots, c_h)$ runs through all non-zero direction vectors in $\mathbb{F}_q^h$. Thus, $\det(M) \neq 0$ if and only if $\alpha_1, \ldots, \alpha_h$ are $\mathbb{F}_q$-linearly independent.

3 Explicit constructions via rational function fields

In this section, we only introduce constructions of MR LRCs from rational function fields. Our description will be self-contained and elementary in terms of polynomials and we don't require any background on algebraic function fields (we have therefore deferred the background on function fields to Section 4 ahead of our more general construction in the next section).

3.1 Constructions via generator matrix

In this subsection, we present constructions of MR LRCs using Definition 1, i.e., via generator matrices of MR LRCs.

Let $N_q(d)$ denote the number of monic irreducible polynomials of degree d over $\mathbb{F}_q$. Then one has $\sum_{d|m} dN_q(d) = q^m$ for any $m \geqslant 1$ (see [14, Corollary 3.21 of Chapter 3]). This gives $\sum_{d|m} N_q(d) \geqslant \frac{q^m}{m}$. For each monic irreducible polynomial $p(x)$ of degree d with $d|m$, we get a polynomial $p(x)^{m/d}$ of degree m. Thus, for any $g \leqslant \left\lceil \frac{q^m}{m} \right\rceil$, there are g polynomials $p_1(x), p_2(x), \ldots, p_g(x)$ of degree m such that $\gcd(p_i(x), p_j(x)) = 1$ for all $1 \leqslant i \neq j \leqslant g$

Assume that (i) $m \geqslant r$; or (ii) $m < r$ and there is a q-ary $[r, r-m, \geqslant r-a+1]$-linear code, i.e. there exists a subset of $\mathbb{F}_q^m$ of size r such that any $r-a$ elements in this subset are $\mathbb{F}_q$-linearly independent.

Choose $g \leqslant \left\lceil \frac{q^m}{m} \right\rceil$ polynomials $p_1(x), p_2(x), \ldots, p_g(x)$ of degree m such that $\gcd(p_i(x), p_j(x)) = 1$ for all $1 \leqslant i \neq j \leqslant g$. Then for each $1 \leqslant i \leqslant g$, we can form an $\mathbb{F}_q$-vector space $V_i := \left\{ \frac{f(x)}{p_i(x)} : f(x) \in \mathbb{F}_q[x], \deg(f(x)) \leqslant m-1 \right\}$ of dimension m. As there is a q-ary $[r, r-m, \geqslant r-a+1]$-linear code, its parity-check matrix is an $r \times m$ matrix and any $r-a$ columns of this matrix are linearly independent. This implies that one can find r functions $g_{i1}(x), \ldots, g_{ir}(x) \in V_i$ such that any $r-a$ polynomials out of $\{g_{i1}(x), \ldots, g_{ir}(x)\}$ are $\mathbb{F}_q$-linearly independent. Choose an irreducible polynomial $Q(x) \in \mathbb{F}_q[x]$ such that $Q(x)$ is coprime with every $p_i(x)$ for $1 \leqslant i \leqslant g$. For a function $h(x) \in V_i$, we use $h(Q)$ to denote the residue class of $h(x)$ in the residue class field $\mathbb{F}_q[x]/Q(x) \simeq \mathbb{F}_{q^{\deg(Q)}}$.

▶ **Lemma 3.** *Let T be a subset $\{1, 2, \ldots, g\}$ with $|T| \leqslant \deg(Q)/m$. If $\sum_{i \in T} g_i(Q) = 0$ for some functions $g_i \in V_i$, then $g_i = 0$ for all $i \in T$.*

Proof. Write $g_i = \frac{f_i}{p_i}$ for some polynomials f_i with $\deg(f_i) \leqslant m-1$. The equality $\sum_{i \in T} g_i(Q) = 0$ implies that $\sum_{i \in T} f_i(x) \prod_{j \in T \setminus \{i\}} p_j(x)$ is divisible by $Q(x)$. As the degree of $\sum_{i \in T} f_i(x) \prod_{j \in T \setminus \{i\}} p_j(x))$ is less than $m|T|$, we must have that $\sum_{i \in T} f_i(x) \prod_{j \in T \setminus \{i\}} p_j(x)$ is the zero polynomial. Suppose that $f_t \neq 0$ for some $t \in T$, then we have

$$\sum_{i \in T \setminus \{t\}} f_i(x) \prod_{j \in T \setminus \{i\}} p_j(x) = -f_t(x) \prod_{j \in T \setminus \{t\}} p_j(x) \, .$$

The l.h.s. of the above equality is divisible by $p_t(x)$, while the r.h.s. is not divisible by $p_t(x)$. This contradiction completes the proof. ◀

Let Q be an irreducible polynomial in $\mathbb{F}_q[x]$ of degree

$$\min\{km, gm\} = \min\left\{km, \frac{nm}{r}\right\} = \min\left\{(n - \frac{an}{r} - h)m, \frac{nm}{r}\right\}.$$

Define the $k \times r$ matrix B_i as follows.

$$B_i = \begin{pmatrix} g_{i1}(Q) & g_{i2}(Q) & \cdots & g_{ir}(Q) \\ g_{i1}^{q}(Q) & g_{i2}^{q}(Q) & \cdots & g_{ir}^{q}(Q) \\ \vdots & \vdots & \vdots & \vdots \\ g_{i1}^{q^{k-1}}(Q) & g_{i2}^{q^{k-1}}(Q) & \cdots & g_{ir}^{q^{k-1}}(Q) \end{pmatrix} \in \mathbb{F}_{q^{\deg(Q)}}^{k \times r}. \tag{8}$$

The proofs of the remaining results of this Section can be found in the full version of the paper that is available at https://arxiv.org/abs/1808.04539.

▶ **Lemma 4.** *Assume that $m \geqslant r$ or there is a q-ary $[r, r-m, \geqslant r-a+1]$-linear code. Let B_i be the matrix given in (8). Put $\ell = q^{\min\left\{(n-\frac{an}{r}-h)m, \frac{nm}{r}\right\}} = q^{\min\left\{km, \frac{nm}{r}\right\}}$ and $G = (B_1|B_2|\cdots|B_g) \in \mathbb{F}_\ell^{k \times n}$. Then the ℓ-ary code C with the generator matrix G is an MR $(n, r, h, a)_\ell$-LRC.*

By taking $m = r$, we obtain the following result.

▶ **Theorem 5.** *If $r \geqslant \log n$, then there exists an MR (n, r, h, a)-LRC of dimension $k = n - \frac{na}{r} - h$ over a field of size*

$$\ell \leqslant \begin{cases} 2^{\min\{rk, n\}} \leqslant 2^n & \text{if } r \geqslant \log n \\ 2^{\min\left\{k\lceil \log n \rceil, \frac{n}{r}\lceil \log n \rceil\right\}} & \text{if } r \leqslant \log n \end{cases}$$

By considering binary BCH codes, we obtain the following binary codes.

▶ **Lemma 6.** *There exists a binary $[r, r-m, \geqslant d]$-linear code with $m = \lfloor \frac{d-1}{2} \rfloor \cdot \lceil \log_2 r \rceil + 1$.*

Combining the binary BCH codes of Lemma 6 with Lemma 4 applied with rational function field $\mathbb{F}_2(x)$ yields the following theorem.

▶ **Theorem 7.** *If $r - a = \Omega(\log n)$, then there exists an MR (n, r, h, a)-LRC of dimension $k = n - \frac{na}{r} - h$ over a field of size*

$$\ell \leqslant 2r^{\min\left\{k\lfloor \frac{r-a}{2} \rfloor, \frac{n}{r}\lfloor \frac{r-a}{2} \rfloor\right\}} \leqslant 2r^{\frac{n}{r}\lfloor \frac{r-a}{2} \rfloor}.$$

3.2 Constructions via parity-check matrix

To construct parity-check matrices of MR LRCs, we only need to construct matrices D_i given in (7). The idea of constructing matrices D_i is quite similar to that of constructing matrices B_i in the previous subsection, and leads to the following theorem.

▶ **Theorem 8.** *Let r, g, a, h, m be positive integers with $a \leqslant r$. Suppose that $q \geqslant r$ is a prime power satisfying $q^m \geqslant \frac{mn}{r}$ and there is a q-ary $[r, r-a, a+1]$-linear code. If (i) $m \geqslant r$; or (ii) $m < r$ and there exists a q-ary $[r, r-m, \geqslant h+a+1]$-linear code, then there exists an MR (n, r, h, a)-LRC with $n = rg$ over a field of size $\ell = q^{\min\{hm, \frac{nm}{r}\}}$.*

We now instantiate Theorem 8 with suitable choices of parameters to deduce the promises parts (i)–(iv) of Theorem 1.

3.2.1 The case where $a = 1$

Let $r, h \geqslant 2$ be integers. Then there is a q-ary $[r, 1, r]$-MDS code for any prime power q. Rewriting Theorem 8 for $a = 1$ gives the following lemma.

▶ **Lemma 9.** *Suppose that $q^m \geqslant \frac{mn}{r}$. If (i) $m \geqslant r$; or (ii) $m < r$ and there exists a q-ary $[r, r-m, \geqslant h+2]$-linear code, then there exists an MR $(n, r, h, 1)$-LRC over a field of size $\ell = q^{\min\{hm, \frac{nm}{r}\}}$.*

To apply Lemma 9, we need to find suitable codes and function fields as well. By taking the rational function field $\mathbb{F}_2(x)$ and applying BCH code given in Lemma 6, we obtain the following result.

▶ **Theorem 10.** *If $r \geqslant h+2$, then there exists an MR $(n, r, h, 1)$-LRC over a field of size*

$$\ell \leqslant \left(\max\left\{\tilde{O}(\frac{n}{r}), (2r)^{\lfloor \frac{h+1}{2} \rfloor}\right\}\right)^{\min\{h, \frac{n}{r}\}}.$$

Proof. Consider the rational function field $F = \mathbb{F}_2(x)$. Put

$$m = \max\left\{\left\lfloor \frac{h+1}{2} \right\rfloor \cdot \lceil \log_2 r \rceil + 1, \left\lceil \log_2\left(\frac{n}{r}\right) + 2\log_2\log_2\left(\frac{n}{r}\right)\right\rceil\right\}.$$

Then $\frac{n}{r} \leqslant \frac{1}{m}2^m$. This implies that there are $\frac{n}{r}$ places of degree m in $\mathbb{F}_2(x)$. By Lemma 6, there exists a binary $[r, r-m, \geqslant h+2]$-linear code. It follows from Lemma 9 that there exists an MR $(n, r, h, 1)$-LRC over a field of size $2^{\min\{mh, m\frac{n}{r}\}}$. By choice of our parameters, the desired result follows. ◀

▶ **Theorem 11.** *There exists an MR $(n, r, h, 1)$-LRC over a field of size*

$$\ell \leqslant \left(\max\left\{\tilde{O}(\frac{n}{r}), 2^r\right\}\right)^{\min\{h, \frac{n}{r}\}}.$$

Proof. Consider the rational function field $\mathbb{F}_2(x)$. Put $m = \max\{r, \left\lceil \log_2\left(\frac{n}{r}\right) + 2\log_2\log_2\left(\frac{n}{r}\right)\right\rceil\}$. Then $\frac{n}{r} \leqslant \frac{1}{m}2^m$. The desired result follows from Lemma 9. ◀

▶ Remark 2. Theorem 11 gives a better bound on the field size than Theorem 10 for $h > \frac{2r}{\log_2 r} - 1$, while Theorem 10 gives a better bound on the field size than Theorem 11 for $h < \frac{2r}{\log_2 r} - 1$.

3.2.2 The case where $2 \leqslant a \leqslant r-1$

▶ **Lemma 12.** *Let $a \leqslant r \leqslant q+1$ and $m \geqslant h+a$. If $q^m \geqslant \frac{mn}{r}$, then there exists an MR (n, r, h, a)-LRC code over a field of size $\ell = q^{\min\{mh, \frac{mn}{r}\}}$.*

Proof. When $a \leqslant r \leqslant q+1$ and $m \geqslant h+a$, we have an $[r, r-a, a+1]_q$-MDS code and an $[r, r-m, h+a+1]_q$-linear code. The result thus follows from Theorem 8. ◀

▶ **Theorem 13.** *There exists an MR (n, r, h, a)-LRC over a field of size*

$$\ell \leqslant \left(\max\left\{\tilde{O}(\frac{n}{r}), (2r)^{h+a}\right\}\right)^{\min\{h, \frac{n}{r}\}}.$$

Proof. Let q be the smallest prime power such that $q-1 \geqslant r$. We may take q to be a power of two, so that $q \leqslant 2r$. Consider the rational function field $F = \mathbb{F}_q(x)$ and let

$$m = \max\left\{h+a, \left\lceil \log_q\left(\frac{n}{r}\right) + 2\log_q\log_q\left(\frac{n}{r}\right)\right\rceil\right\}.$$

Then $\frac{n}{r} \leqslant \frac{1}{m}q^m$. The desired result follows from Theorem 8. ◀

▶ Remark 3. The field size $\ell \leqslant \tilde{O}\left(\max\left\{\frac{n}{r}, r^{h+a}\right\}^h\right)$ in Theorem 13 was already given in [5, Corollary 11]. Here we provide a better result for $h > \frac{n}{r}$ via a different approach.

▶ **Theorem 14.** *There exists an MR (n, r, h, a)-LRC over a field of size*

$$\ell \leqslant \left(\max\left\{\tilde{O}(\frac{n}{r}), (2r)^r\right\}\right)^{\min\{h, \frac{n}{r}\}}.$$

Proof. Put $q = 2^{\lceil \log_2 r \rceil}$. Then $2r \geqslant q \geqslant r$ and hence we have a q-ary $[r, a]$-MDS code for any $a \leqslant r$. Put $m = \max\{r, \lceil \log_q\left(\frac{n}{r}\right) + 2\log_q \log_q\left(\frac{n}{r}\right)\rceil\}$. Then $\frac{n}{r} \leqslant \frac{1}{m}q^m$. The desired result follows from Theorem 10. ◀

▶ Remark 4. Theorem 14 gives a better bound on the field size than Theorem 13 for $h+a > r$, while Theorem 13 gives a better bound on the field size than Theorem 14 for $h + a < r$.

4 Explicit construction via general function fields

The construction via rational function fields given in Section 3 can be easily generalized to arbitrary function fields. We only generalize the constructions of MR LRCs via parity-check matrices given in Section 3.2. The necessary background on algebraic function fields, and specifically Hermitian and Garcia-Stichtenoth tower of function fields, can be found in the full version of this paper. We refer the proofs in this section to the full version of this paper.

Let q be a prime power and let a, r, h, g be integers with $a \leqslant r \leqslant q + 1$. Let $F/\mathbb{F}_q$ be a function field of genus $\mathfrak{g}$. Let $P_1, P_2, \ldots, P_g$ be g positive divisors of degree r whose supports are pairwise disjoint. Let G be a divisor of degree $2\mathfrak{g} - 1$. By Riemann-Roch, $\dim \mathcal{L}(G) = \mathfrak{g}$. Assume that $\{f_1, f_2, \ldots, f_\mathfrak{g}\}$ is a basis of $\mathcal{L}(G)$. For each i, extend this basis to a basis $\{f_1, f_2, \ldots, f_\mathfrak{g}, f_{i1}, f_{i2}, \ldots, f_{ir}\}$ of $\mathcal{L}(G + P_i)$.

Let Q be a place of degree $2\mathfrak{g} + \min\{hr, n\}$ and define the matrix

$$D_i = \begin{pmatrix} f_{i1}(Q) & f_{i2}(Q) & \cdots & f_{ir}(Q) \\ f_{i1}^q(Q) & f_{i2}^q(Q) & \cdots & f_{ir}^q(Q) \\ \vdots & \vdots & \vdots & \vdots \\ f_{i1}^{q^{h-1}}(Q) & f_{i2}^{q^{h-1}}(Q) & \cdots & f_{ir}^{q^{h-1}}(Q) \end{pmatrix} \tag{9}$$

By mimicking the proof of Theorem 8, we have the following result.

▶ **Lemma 15.** *Let $A_i \in \mathbb{F}_q^{a\times r}$ be a generator matrix of an $[r, a]_q$-MDS code for $1 \leqslant i \leqslant g$. Let D_i be the matrix given in (9). Put $\ell = q^{2\mathfrak{g}+\min\{hr,n\}}$. Then the ℓ-ary code C with the matrix H defined in (7) is an MR $(n, r, h, a)_\ell$-LRC.*

Consequently, we have the following theorem.

▶ **Theorem 16.** *Let r, g, a, h be positive integers with $a \leqslant r \leqslant q + 1$. If there is a function field $F/\mathbb{F}_q$ of genus $\mathfrak{g}$ with g positive divisors of degree r whose supports are disjoint, then there exists an MR (n, r, h, a)-LRC with $n = rg$ over a field of size $\ell = q^{2\mathfrak{g}+\min\{hr,n\}}$.*

Finally, let us instantiate the above result with the Hermitian function fields and the Garcia-Stichtenoth tower, to deduce Parts (v) and (vi) promised in Theorem 1 respectively. Note that both the results below kick-in for block lengths which are asymptotically at least $r^{O(r)}$, which is why we have the condition $r \leqslant O(\frac{\log n}{\log\log n})$ in the statement of Theorem 1, Parts (v), (vi).

▶ **Theorem 17.** *Let $a \leqslant r$ be integers. Then there are infinitely many $n \geqslant r^{\Omega(r)}$ such that there is MR (n, r, h, a)-LRC over a field of size at most $n^{\frac{2h}{3}(1+\varepsilon)}$ for any desired $\varepsilon \in (0, 0.5)$ provided $hr \geqslant \Omega\left(\frac{n^{\frac{2}{3}}}{\varepsilon}\right)$.*

We finally state a similar result using the Garcia-Stichtenoth tower of function fields.

▶ **Theorem 18.** *Let $a \leqslant r$ be positive integers and let $\varepsilon \in (0, 0.5)$. Then there are infinitely many $n \geqslant r^{\Omega(r/\varepsilon)}$ such that there is MR (n, r, h, a)-LRC over a field of size at most $n^{\varepsilon h}$ provided $hr \geqslant \Omega\left(n^{1-\varepsilon}\right)$.*

References

1. Mario Blaum. Construction of PMDS and SD Codes extending RAID 5. *arXiv preprint arXiv:1305.0032*, 2013. `arXiv:1305.0032`.
2. Mario Blaum, James Lee Hafner, and Steven Hetzler. Partial-MDS codes and their application to RAID type of architectures. *IEEE Transactions on Information Theory*, 59(7):4510–4519, 2013.
3. Mario Blaum, James S Plank, Moshe Schwartz, and Eitan Yaakobi. Construction of partial MDS and sector-disk codes with two global parity symbols. *IEEE Transactions on Information Theory*, 62(5):2673–2681, 2016.
4. Gokhan Calis and O Ozan Koyluoglu. A general construction for PMDS codes. *IEEE Communications Letters*, 21(3):452–455, 2017.
5. Ryan Gabrys, Eitan Yaakobi, Mario Blaum, and Paul H Siegel. Constructions of partial MDS codes over small fields. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 1–5. IEEE, 2017.
6. Parikshit Gopalan, Guangda Hu, Swastik Kopparty, Shubhangi Saraf, Carol Wang, and Sergey Yekhanin. Maximally recoverable codes for grid-like topologies. In *28th Annual Symposium on Discrete Algorithms (SODA)*, pages 2092–2108. Society for Industrial and Applied Mathematics, 2017.
7. Parikshit Gopalan, Cheng Huang, Bob Jenkins, and Sergey Yekhanin. Explicit maximally recoverable codes with locality. *IEEE Transactions on Information Theory*, 60(9):5245–5256, 2014.
8. Parikshit Gopalan, Cheng Huang, Huseyin Simitci, and Sergey Yekhanin. On the locality of codeword symbols. *IEEE Transactions on Information Theory*, 58(11):6925–6934, 2012.
9. Sivakanth Gopi, Venkatesan Guruswami, and Sergey Yekhanin. On maximally recoverable local reconstruction codes. *arXiv preprint arXiv:1710.10322*, 2017.
10. Venkatesan Guruswami, Chaoping Xing, and Chen Yuan. How long can optimal locally repairable codes be? In *Proceedings of RANDOM 2018*, pages 41:1–41:11, 2018.
11. Guangda Hu and Sergey Yekhanin. New constructions of SD and MR codes over small finite fields. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 1591–1595. IEEE, 2016.
12. Cheng Huang, Huseyin Simitci, Yikang Xu, Aaron Ogus, Brad Calder, Parikshit Gopalan, Jin Li, and Sergey Yekhanin. Erasure coding in windows azure storage. In *USENIX Annual Technical Conference (ATC)*, pages 15–26, 2012.
13. Daniel Kane, Shachar Lovett, and Sankeerth Rao. The independence number of the birkhoff polytope graph, and applications to maximally recoverable codes. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 252–259. IEEE, 2017.
14. Rudolf Lidl and Harald Niederreiter. *Finite fields*, volume 20. Cambridge university press, 2003.
15. Alessandro Neri and Anna-Lena Horlemann-Trautmann. Random Construction of Partial MDS Codes. *arXiv preprint arXiv:1801.05848*, 2018.
16. Dimitris S Papailiopoulos and Alexandros G Dimakis. Locally repairable codes. *IEEE Transactions on Information Theory*, 60(10):5843–5855, 2014.

17 Itzhak Tamo and Alexander Barg. A family of optimal locally recoverable codes. *IEEE Transactions on Information Theory*, 60(8):4661–4676, 2014.

18 Itzhak Tamo, Dimitris S Papailiopoulos, and Alexandros G Dimakis. Optimal locally repairable codes and connections to matroid theory. *IEEE Transactions on Information Theory*, 62(12):6661–6671, 2016.

Quantum Chebyshev's Inequality and Applications

Yassine Hamoudi
Université de Paris, IRIF, CNRS, F-75013 Paris, France
hamoudi@irif.fr

Frédéric Magniez
Université de Paris, IRIF, CNRS, F-75013 Paris, France
magniez@irif.fr

Abstract

In this paper we provide new quantum algorithms with polynomial speed-up for a range of problems for which no such results were known, or we improve previous algorithms. First, we consider the approximation of the frequency moments F_k of order $k \geq 3$ in the multi-pass streaming model with updates (turnstile model). We design a P-pass quantum streaming algorithm with memory M satisfying a tradeoff of $P^2M = \widetilde{\mathcal{O}}\left(n^{1-2/k}\right)$, whereas the best classical algorithm requires $PM = \Theta(n^{1-2/k})$. Then, we study the problem of estimating the number m of edges and the number t of triangles given query access to an n-vertex graph. We describe optimal quantum algorithms that perform $\widetilde{\mathcal{O}}\left(\sqrt{n}/m^{1/4}\right)$ and $\widetilde{\mathcal{O}}\left(\sqrt{n}/t^{1/6} + m^{3/4}/\sqrt{t}\right)$ queries respectively. This is a quadratic speed-up compared to the classical complexity of these problems.

For this purpose we develop a new quantum paradigm that we call Quantum Chebyshev's inequality. Namely we demonstrate that, in a certain model of quantum sampling, one can approximate with *relative error* the mean of any random variable with a number of quantum samples that is linear in the ratio of the square root of the variance to the mean. Classically the dependence is quadratic. Our algorithm subsumes a previous result of Montanaro [47]. This new paradigm is based on a refinement of the Amplitude Estimation algorithm of Brassard et al. [11] and of previous quantum algorithms for the mean estimation problem. We show that this speed-up is optimal, and we identify another common model of quantum sampling where it cannot be obtained. Finally, we develop a new technique called "variable-time amplitude estimation" that reduces the dependence of our algorithm on the sample preparation time.

2012 ACM Subject Classification Theory of computation → Quantum computation theory

Keywords and phrases Quantum algorithms, approximation algorithms, sublinear-time algorithms, Monte Carlo method, streaming algorithms, subgraph counting

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.69

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1807.06456`.

Funding This research was supported by the French ANR project ANR-18-CE47-0010 (QUDATA) and the QuantERA ERA-NET Cofund project QuantAlgo.

Acknowledgements The authors want to thank the anonymous referees for their valuable comments and suggestions which helped to improve this paper.

1 Introduction

Motivations and Background. Randomization and probabilistic methods are among the most widely used techniques in modern science, with applications ranging from mathematical economics to medicine or particle physics. One of the most successful probabilistic approaches is the Monte Carlo Simulation method for algorithm design, that relies on repeated random sampling and statistical analysis to estimate parameters and functions of interest. From Buffon's needle experiment, in the eighteenth century, to the simulations of galaxy formation

or nuclear processes, this method and its variations have become increasingly popular to tackle problems that are otherwise intractable. The Markov chain Monte Carlo method [35] led for instance to significant advances for approximating parameters whose exact computation is #P-hard [39, 37, 20, 36].

The analysis of Monte Carlo Simulation methods is often based on concentration inequalities that characterize the deviation of a random variable from some parameter. In particular, the Chebyshev inequality is a key element in the design of randomized methods that estimate some target numerical value. Indeed, this inequality guarantees that the arithmetic mean of Δ^2/ϵ^2 independent samples, from a random variable with variance σ^2 and mean μ satisfying $\Delta \geq \sigma/\mu$, is an approximation of μ under relative error ϵ with high probability. This basic result is at the heart of many computational problems, such as counting via Markov chains [35, 54], estimating graph parameters [16, 25, 28, 22], testing properties of classical [29, 8, 15, 13] or quantum [12, 7] distributions, approximating the frequency moments in the data stream model [2, 46, 4].

Various quantum algorithms have been developed to speed-up or generalize classical Monte Carlo methods (e.g. sampling the stationary distributions of Markov-chains [55, 51, 19, 53, 17], estimating the expected values of observables or partition functions [41, 56, 51, 47]). The mean estimation problem (as addressed by Chebyshev's inequality) has also been studied in the *quantum sampling model*. In this model, a distribution is represented by a unitary transformation (called a *quantum sampler*) preparing a superposition over the elements of the distribution, with the amplitudes encoding the probability mass function. A *quantum sample* is defined as one execution of a quantum sampler or its inverse. The number of quantum samples needed to estimate the mean of a distribution on a bounded space $[0, B]$, with *additive* error ϵ, was proved to be $\mathcal{O}(B/\epsilon)$ [32, 10], or $\widetilde{\mathcal{O}}(\bar{\sigma}/\epsilon)$ [47] given an upper-bound $\bar{\sigma}^2$ on the variance. On the other hand, the mean estimation problem with *relative* error ϵ can be solved with $\mathcal{O}\left(\sqrt{B}/(\epsilon\sqrt{\mu})\right)$ quantum samples [11, 56]. Interestingly, this is a quadratic improvement over $\sigma^2/(\epsilon\mu)^2$ if the sample space is $\{0, B\}$ (this case maximizes the variance). Montanaro [47] posed the problem of whether this speed-up can be generalized to other distributions. He assumed that one knows an upper bound[1] Δ on $1 + \sigma/\mu$, and gave an algorithm using[2] $\widetilde{\mathcal{O}}\left(\Delta^2/\epsilon\right)$ quantum samples (thus improving the dependence on ϵ, compared to the classical setting). This result was reformulated in [43] to show that, having bounds $L \leq \mu \leq H$, it is possible to use $\widetilde{\mathcal{O}}(\Delta/\epsilon \cdot H/L)$ quantum samples. However, it is usually the case that the only upper-bound known on μ is $H = B$. In this situation, the latter algorithm is less efficient than previous works [11, 56].

Quantum Chebyshev's Inequality. Our main contribution (Theorem 10 and Theorem 11) is to show that the mean μ of any distribution with variance σ^2 can be approximated with relative error ϵ using $\widetilde{\mathcal{O}}(\Delta \cdot \log(H/L) + \Delta/\epsilon)$ quantum samples, given an upper bound Δ on $1 + \sigma/\mu$ and two bounds L, H such that $L < \mu < H$. This is an exponential improvement in H/L compared to previous works [43]. Moreover, if $\log(H/L)$ is negligible, this is a quadratic improvement over the number of classical samples needed when using the Chebyshev inequality. A corresponding lower bound is deduced from [50] (Theorem 12). We also show (Theorem 14) that no such speed-up is possible if we only had access to *copies* of the quantum state representing the distribution.

---

[1] More precisely, Δ is an upper bound on ϕ/μ where ϕ^2 is the second moment, which satisfies $\sigma/\mu \leq \phi/\mu \leq 1 + \sigma/\mu$.

[2] We use the notation $\widetilde{\mathcal{O}}(x)$ to indicate $\mathcal{O}(x \cdot \operatorname{polylog} x)$.

Our algorithm is based on *sequential analysis*. Given a threshold $b \geq 0$, we will consider the "truncated" mean $\mu_{<b}$ defined by replacing the outcomes larger than b with 0. Using standard techniques, this mean can be encoded in the amplitude of some quantum state $\sqrt{1-\mu_{<b}/b}|\psi\rangle + \sqrt{\mu_{<b}/b}|\psi^{\perp}\rangle$ (Corollary 4). We then run the *Amplitude Estimation* algorithm of Brassard et al. [11] on this state for Δ steps (i.e. with Δ quantum samples), only to see whether the estimate of $\mu_{<b}/b$ it returns is nonzero (this is our *stopping rule*). A property of this algorithm (Corollary 4 and Remark 7) guarantees that it is zero with high probability if and only if the number of quantum samples is below the inverse $\sqrt{b/\mu_{<b}}$ of the estimated amplitude. The crucial observation (Lemma 9) is that $\sqrt{b/\mu_{<b}}$ is smaller than Δ for large values of b, and it becomes larger than Δ when $b \approx \mu\Delta^2$. Thus, by repeatedly running the amplitude estimation algorithm with Δ quantum samples, and doing $\mathcal{O}(\log(H/L))$ steps of a logarithmic search on decreasing values of b, the first non-zero value is obtained when b/Δ^2 is approximately equal to μ. The precision of the result is later improved, by using more precise "truncated" means.

This algorithm is extended to cover the common situation where one knows a non-increasing function f such that $f(\mu) \geq 1 + \sigma/\mu$, instead of having explicitly $\Delta \geq 1 + \sigma/\mu$. For this purpose, we exhibit another property (Corollary 4 and Remark 6) of the amplitude estimation algorithm, namely that it always outputs a number smaller than the estimated value (up to a constant factor) with high probability. This shall be seen as a quantum equivalent of the Markov inequality. Combined with the previous algorithm, it allows us to find a value $f(\widetilde{\mu}) \geq 1 + \sigma/\mu$, with a second logarithmic search on $\widetilde{\mu}$. This result is detailed in the full version of the paper [31].

Next, we study the quantum analogue of the following standard fact: s classical samples, each taking *average* time T_{av} to be prepared, can be obtained in total average time $s \cdot T_{av}$. The notion of "average preparation time" is adapted to the quantum setting using the framework of variable-time algorithms introduced by Ambainis [3]. This captures the situation where the superposition prepared by the quantum sampler has different parts taking different times to be computed. We develop a variable-time amplitude estimation algorithm that approximates the target value efficiently in this case. We use it in place of the standard amplitude estimation technique to obtain an algorithm whose complexity depends on the average, instead of worst-case, sample preparation time. This result is detailed in the full version of the paper [31].

Applications. We describe two applications that illustrate the use of the above results. We first study the problem of approximating the frequency moments F_k of order $k \geq 3$ in the multi-pass streaming model with updates. Classically, the best P-pass algorithms with memory M satisfy $PM = \Theta\left(n^{1-2/k}\right)$ [46, 57]. We give a quantum algorithm for which $P^2M = \widetilde{\mathcal{O}}\left(n^{1-2/k}\right)$ (Theorem 18). This problem was studied before in [48], where the author obtained quantum speed-ups for F_0, F_2 and F_∞, but no significant improvement for $k \geq 3$. Similar tradeoff results are known for Disjointness ($P^2M = \widetilde{\Theta}(n)$ in the quantum streaming model [42] vs. $PM = \Theta(n)$ classically), and Dyck(2) ($P^3M = \Omega(\sqrt{n})$ [49] vs. $PM = \widetilde{\Theta}(\sqrt{n})$ [45, 14, 34]).

Our construction starts with a classical one-pass *linear sketch* streaming algorithm [46, 4] with memory polylog n, that samples (approximately) from a distribution with mean F_k and variance $\mathcal{O}\left(n^{1-2/k}F_k^2\right)$. We implement it with a quantum sampler, that needs two passes for one quantum sample. The crucial observation is that the reverse computation of a linear sketch algorithm can be done efficiently in one pass (whereas usually that would require processing the same stream but in the reverse direction).

As a second application, we study the approximation of graph parameters using neighbor, vertex-pair and degree queries. We show that the numbers m of edges and t of triangles, in an n-vertex graph, can be estimated with $\widetilde{\Theta}\left(n^{1/2}/m^{1/4}\right)$ (Theorem 19) and $\widetilde{\Theta}\left(\sqrt{n}/t^{1/6} + m^{3/4}/\sqrt{t}\right)$ (Theorem 21) quantum queries respectively. This is a quadratic speed-up over the best classical algorithms [28, 22]. The lower bounds (Theorems 20 and 22) are obtained with a property testing to communication complexity reduction method.

The number of edges is approximated by translating a classical estimator [52] into a quantum sampler. The triangle counting algorithm is more involved. We need a classical estimator [22] approximating the number t_v of adjacent triangles to any vertex v. The average sample preparation time of this estimator being small, we obtain a quadratic speed-up for estimating t_v using our mean estimation algorithm for variable-time samplers. We then diverge from the classical triangle counting algorithm of [22], that requires to set up a data structure for sampling edges uniformly in the graph. This technique seems to be an obstacle for a quadratic speed-up. We circumvent this problem by adapting instead a bucketing approach from [21] that partitions the graph's vertices according to the value of t_v. The size of each bucket is estimated using a second quantum sampler.

2 Preliminaries

2.1 Computational Model

In this paper we consider probability distributions d on some finite sample spaces $\Omega \subset \mathbb{R}^+$. We denote by $d(x)$ the probability to sample $x \in \Omega$ in the distribution d. We also make the assumption, which is satisfied for most of applications, that Ω is equipped with an efficient encoding of its elements $x \in \Omega$. In particular, we can perform quantum computations on the Hilbert space $\mathcal{H}_\Omega$ defined by the basis $\{|x\rangle\}_{x\in\Omega}$. Moreover, given any two values $0 \leq a < b$, we assume the existence of a unitary $R_{a,b}$ that can perform the *Bernoulli sampling* (see below) in time polylogarithmic in b. In the rest of the paper we will neglect this complexity, including the required precision for implementing any of those unitary operators.

▶ **Definition 1.** *Given a finite space $\Omega \subset \mathbb{R}^+$ and two reals $0 \leq a < b$, an* (a,b)-Bernoulli sampler over Ω *is a unitary $R_{a,b}$ acting on $\mathcal{H}_\Omega \otimes \mathbb{C}^2$ and satisfying for all $x \in \Omega$:*

$$R_{a,b}(|x\rangle|0\rangle) = \begin{cases} |x\rangle\left(\sqrt{1-\frac{x}{b}}|0\rangle + \sqrt{\frac{x}{b}}|1\rangle\right) & \text{when } a \leq x < b, \\ |x\rangle|0\rangle & \text{otherwise.} \end{cases}$$

We say that Ω is Bernoulli samplable *if any (a,b)-Bernoulli sampler can be implemented in polylogarithmic time in b, when a, b have polylog-size encodings in b.*

The $R_{a,b}$ operation can be implemented with a controlled rotation, and is reminiscent of related works on mean estimation (e.g. [56, 10, 47]). In what follows, we always use $a = 0$ or $a = b/2$. Using these notions, we can now define what a *quantum sample* is.

▶ **Definition 2.** *Given a finite Bernoulli samplable space $\Omega \subset \mathbb{R}^+$ and a distribution d on Ω, a* (quantum) sampler $\mathcal{S}$ *for d is a unitary operator acting on $\mathcal{H}_g \otimes \mathcal{H}_\Omega$, for some Hilbert space $\mathcal{H}_g$, such that*

$$\mathcal{S}(|0\rangle|0\rangle) = \sum_{x\in\Omega} \sqrt{d(x)}|\psi_x\rangle|x\rangle$$

where $|\psi_x\rangle$ are arbitrary unit vectors. A quantum sample *is one execution of $\mathcal{S}$ or $\mathcal{S}^{-1}$ (including their controlled versions). The* output *of $\mathcal{S}$ is the random variable $v(\mathcal{S})$ obtained by measuring the x-register of $\mathcal{S}(|0\rangle|0\rangle)$. Its* mean *is denoted by $\mu_\mathcal{S}$, its* variance *by $\sigma^2_\mathcal{S}$, and its* second moment *by $\phi^2_\mathcal{S} = \mathbb{E}\left[v(\mathcal{S})^2\right]$.*

Given a non-negative random variable X and two numbers $0 \leq a \leq b$, we define the random variable $X_{a,b} = \mathrm{id}_{a,b}(X)$ where $\mathrm{id}_{a,b}(x) = x$ when $a \leq x < b$ and $\mathrm{id}_{a,b}(x) = 0$ otherwise. If $a = 0$, we let $X_{<b} = X_{0,b}$. Similarly, $X_{\geq b} = \mathrm{id}_{\geq b}(X)$ where $\mathrm{id}_{\geq b}(x) = x$ when $x \geq b$ and $\mathrm{id}_{\geq b}(x) = 0$ otherwise.

We motivate the use of a Bernoulli sampler $R_{a,b}$ by the following observation: for any sampler $\mathcal{S}$ and values $0 \leq a < b$, the modified sampler $\hat{\mathcal{S}} = (I_{\mathcal{H}_g} \otimes R_{a,b})(\mathcal{S} \otimes I_{\mathbb{C}^2})$ acting on $\mathcal{H}_{\hat{g}} \otimes \mathcal{H}_{\hat{\Omega}}$, where $\mathcal{H}_{\hat{g}} = \mathcal{H}_g \otimes \mathcal{H}_\Omega$ and $\hat{\Omega} = \{0,1\}$, generates the Bernoulli distribution $d(0) = 1 - p$, $d(1) = p$ of mean $p = \mathbb{E}\left[v(\hat{\mathcal{S}})\right] = b^{-1}\mathbb{E}\left[v(\mathcal{S})_{a,b}\right]$ (see the proof of Corollary 4). This central result will be used all along this paper.

Other Quantum Sampling Models. Instead of having access to the unitary $\mathcal{S}$, one could only have copies of the state $\sum_{x\in\Omega}\sqrt{d(x)}|\psi_x\rangle|x\rangle$ (as in [5] for instance). However, as we show in Theorem 14, the speed-up presented in this paper is impossible to achieve in this model. On another note, Aharonov and Ta-Shma [1] studied the *Qsampling* problem, which is the ability to prepare $\sum_{x\in\Omega}\sqrt{d(x)}|x\rangle$ given the decription of a classical circuit with output distribution d. This problem becomes straightforward if a garbage register ψ_x can be added (using standard reversible-computation techniques). Bravyi, Harrow and Hassidim [12] considered an oracle-based model, that is provably weaker than Qsampling, where a distribution $d = (d(1), \ldots, d(N))$ on $\Omega = [N]$ is represented by an oracle $O_d : [S] \to [N]$ (for some S), such that $d(x)$ equals the proportion of inputs $s \in [S]$ with $O_d(s) = x$. It is extended to the quantum query framework with a unitary $\mathcal{O}_d$ such that $\mathcal{O}_d|s\rangle|0\rangle = |s\rangle|O_d(s)\rangle$. It is not difficult to see that applying $\mathcal{O}_d$ on a uniform superposition gives $\sum_{x\in[N]}\sqrt{d(x)}\left(\frac{1}{\sqrt{d(x)S}}\sum_{s\in[S]:O_d(s)=x}|s\rangle\right)|x\rangle$, as required by Definition 2 (where $|\psi_x\rangle = \frac{1}{\sqrt{d(x)S}}\sum_{s\in[S]:O_d(s)=x}|s\rangle$). Finally, Montanaro [47] presented a model that is similar to ours, where he replaced the x-register of $\mathcal{S}(|0\rangle|0\rangle)$ with a k-qubit register (for some k) combined with a mapping $\phi : \{0,1\}^k \to \Omega$ where $x = \phi(s)$ is the sample associated to each $s \in \{0,1\}^k$.

2.2 Amplitude Estimation

The essential building block of this paper is the amplitude estimation algorithm [11], combined with ideas from [56, 10, 47], to estimate the modified mean $b^{-1}\mathbb{E}\left[v(\mathcal{S})_{a,b}\right]$ of a quantum sampler $\mathcal{S}$ to which a Bernoulli sampler $R_{a,b}$ has been applied. We will need the following result about amplitude estimation.

▶ **Theorem 3.** *There is a quantum algorithm* AmplEst*, called* Amplitude Estimation*, that takes as input a unitary operator U, an orthogonal projector Π, and an integer $t > 2$. The algorithm outputs an estimate $\widetilde{p} = \mathsf{AmplEst}\,(U, \Pi, t)$ of $p = \langle\psi|\Pi|\psi\rangle$, where $|\psi\rangle = U|0\rangle$, such that*

$$\begin{cases} |\widetilde{p} - p| \leq 2\pi\frac{\sqrt{p}}{t} + \frac{\pi^2}{t^2}, & \text{with probability } 8/\pi^2; \\ \widetilde{p} = 0, & \text{with probability } \frac{\sin^2(t\theta)}{t^2\sin^2(\theta)}. \end{cases}$$

and $0 \leq \theta \leq \pi/2$ satisfies $\sin(\theta) = \sqrt{p}$. It uses $\mathcal{O}\left(\log^2(t)\right)$ 2-qubit quantum gates (independent of U and Π) and makes $2t+1$ calls to (the controlled versions of) U and U^{-1}, and t calls to the reflection $I - 2\Pi$.

We now present an adaptation of the algorithms from [56, 10, 47] to estimate $b^{-1}\mathbb{E}\left[v(\mathcal{S})_{a,b}\right]$.

Input: a sampler $\mathcal{S}$ acting on $\mathcal{H}_g \otimes \mathcal{H}_\Omega$, two values (a, b), an integer t, a failure parameter $0 < \delta < 1$.
Output: an estimate $\widetilde{p} = \mathsf{BasicEst}(\mathcal{S}, (a, b), t, \delta)$ of $p = b^{-1}\mathbb{E}[v(\mathcal{S})_{a,b}]$

1. Let $U = (I_{\mathcal{H}_g} \otimes R_{a,b})(\mathcal{S} \otimes I_{\mathbb{C}^2})$ and $\Pi = I_{\mathcal{H}_g} \otimes I_{\mathcal{H}_\Omega} \otimes |1\rangle\langle 1|$.
2. For $i = 1, \ldots, \Theta(\log(1/\delta))$: compute $\widetilde{p}_i = \mathsf{AmplEst}(U, \Pi, t)$.
3. Output $\widetilde{p} = \mathrm{median}\{\widetilde{p}_1, \ldots, \widetilde{p}_{\Theta(\log(1/\delta))}\}$.

Algorithm 1 The Basic Estimation algorithm BasicEst.

▶ **Corollary 4.** *Consider a quantum sampler $\mathcal{S}$ and two values $0 \leq a < b$. Denote $p = b^{-1}\mathbb{E}[v(\mathcal{S})_{a,b}]$. Given an integer $t > 2$ and a real $0 < \delta < 1$, $\mathsf{BasicEst}(\mathcal{S}, (a, b), t, \delta)$ (see Algorithm 1) uses $\mathcal{O}(t \log(1/\delta))$ quantum samples and outputs $\widetilde{p}$ satisfying all of the following inequalities with probability $1 - \delta$:*

(1) $|\widetilde{p} - p| \leq 2\pi\frac{\sqrt{p}}{t} + \frac{\pi^2}{t^2}$, *for any t;* (2) $\widetilde{p} \leq (1 + 2\pi)^2 \cdot p$, *for any t;*

(3) $\widetilde{p} = 0$, *when $t < \frac{1}{2\sqrt{p}}$;* (4) $|\widetilde{p} - p| \leq \epsilon \cdot p$, *when $t \geq \frac{8}{\epsilon\sqrt{p}}$ and $0 < \epsilon < 1$.*

Proof. We show that each $\widetilde{p}_i$ satisfies the inequalities stated in the corollary, with probability $8/\pi^2$. Since $\widetilde{p}$ is the median of $\Theta(\log 1/\delta)$ such values, the probability is increased to $1 - \delta$ using the Chernoff bound.

For each $x \in \Omega$, denote $\nu_x = \frac{x}{b}$ if $a \leq x < b$, and $\nu_x = 0$ otherwise. Since $p = \sum_{x\in\Omega} \nu_x d(x)$, observe that

$$U(|0\rangle|0\rangle|0\rangle) = \sum_{x\in\Omega} \sqrt{d(x)}|\psi_x\rangle|x\rangle \left(\sqrt{1-\nu_x}|0\rangle + \sqrt{\nu_x}|1\rangle\right) = \sqrt{1-p}|\psi_0'\rangle|0\rangle + \sqrt{p}|\psi_1'\rangle|1\rangle$$

where $|\psi_0'\rangle = \frac{1}{\sqrt{1-p}} \sum_{x\in\Omega} \sqrt{d(x)}\sqrt{1-\nu_x}|\psi_x\rangle|x\rangle$ and $|\psi_1'\rangle = \frac{1}{\sqrt{p}} \sum_{x\in\Omega} \sqrt{d(x)}\sqrt{\nu_x}|\psi_x\rangle|x\rangle$ are unit vectors. Thus, the output $\widetilde{p}_i$ of the AmplEst algorithm applied on U and Π is an estimate of p satisfying the output conditions of Theorem 3. Therefore $|\widetilde{p}_i - p| \leq 2\pi\frac{\sqrt{p}}{t} + \frac{\pi^2}{t^2}$ with probability $8/\pi^2$, for any t. By plugging $t \geq \frac{8}{\epsilon\sqrt{p}}$ into this inequality we have $|\widetilde{p}_i - p| \leq \epsilon \cdot p$. By plugging $t \geq \frac{1}{2\sqrt{p}}$ we also have $|\widetilde{p}_i - p| \leq (4\pi + 4\pi^2)p$, and thus $\widetilde{p}_i \leq (1 + 2\pi)^2 \cdot p$. Finally, if $t < \frac{1}{2\sqrt{p}}$, denote $0 \leq \theta \leq \pi/2$ such that $\sin(\theta) = \sqrt{p}$ and observe that $\theta \leq \frac{\pi}{2}\sqrt{p} \leq \frac{\pi}{4t}$ (since $\frac{2}{\pi}x \leq \sin(x) \leq x$, for $x \in [0, \pi/2]$). The probability to obtain $\widetilde{p}_i = 0$ is $\frac{\sin^2(t\theta)}{t^2\sin^2(\theta)} \geq \frac{\sin^2(t\pi/(4t))}{t^2\sin^2(\pi/(4t))} \geq \frac{\sin^2(\pi/4)}{t^2(\pi/(4t))^2} = 8/\pi^2$, since $x \mapsto \sin^2(tx)/(t^2\sin^2(x))$ is decreasing for $0 < x \leq \pi/t$. Moreover, when $t < \frac{1}{2\sqrt{p}}$, the first two inequalities are obviously satisfied if $\widetilde{p}_i = 0$. ◀

The four results on p in Corollary 4 lie at the heart of this paper. We make a few comments on them.

▶ Remark 5. Consider a sampler $\mathcal{S}$ over $\Omega = \{0, 1\}$ for the Bernoulli distribution of parameter p. Using the Chebyshev inequality, we get that $\mathcal{O}\left((1-p)/(\epsilon^2 p)\right)$ classical samples are enough for estimating p with relative error ϵ. The inequality (4) of Corollary 4 shows that $t = \mathcal{O}\left(1/(\epsilon\sqrt{p})\right)$ quantum samples are sufficient. Our main result (Section 3) generalizes this quadratic speed-up to the non-Bernoulli case.

▶ Remark 6. The inequality (2) shall be seen as an equivalent of the Markov inequality[3], namely that $\widetilde{p}$ does not exceed p by a large factor with large probability.

[3] The Markov inequality for a non-negative random variable X states that $\mathbb{P}(X \geq k\mathbb{E}[X]) \leq 1/k$ for any $k > 0$. Here, although we do not need this result, it is possible to prove that $\mathbb{P}(\widetilde{p} \geq kp) \leq C/\sqrt{k}$, for some absolute constant C.

Input: a sampler $\mathcal{S}$, an integer $\Delta_{\mathcal{S}}$, two values $0 < L < H$, two reals $0 < \epsilon, \delta < 1/2$.
Output: an estimate $\widetilde{\mu}_{\mathcal{S}}$ of $\mu_{\mathcal{S}}$.

1. Set $M = 8H$ and $\widetilde{p} = 0$
2. While $\widetilde{p} = 0$ and $M \geq 2L$:
   a. Set $M = M/2$.
   b. Compute $\widetilde{p} = \mathsf{BasicEst}\left(\mathcal{S}, (0, M\Delta_{\mathcal{S}}^2), 25\Delta_{\mathcal{S}}, \delta'\right)$ where $\delta' = \frac{\delta}{2(3+\log(H/L))}$.
3. If $M < 2L$ then output $\widetilde{\mu}_{\mathcal{S}} = 0$.
4. Else, compute $\widetilde{q} = \mathsf{BasicEst}\left(\mathcal{S}, (0, \epsilon^{-1}M\Delta_{\mathcal{S}}^2), 35^2\epsilon^{-3/2}\Delta_{\mathcal{S}}, \delta/2\right)$ and output $\widetilde{\mu}_{\mathcal{S}} = (\epsilon^{-1}M\Delta_{\mathcal{S}}^2) \cdot \widetilde{q}$.

Algorithm 2 $\epsilon-$approximation of the mean of a quantum sampler $\mathcal{S}$.

▶ Remark 7. If $p \neq 0$, inequalities (3) and (4) imply that, with large probability, $t < 8/\sqrt{p}$ when $\widetilde{p} = 0$, and $t \geq 1/(2\sqrt{p})$ when $\widetilde{p} \neq 0$. This phenomenon, at $t = \Theta(1/\sqrt{p})$, is crucially used in the next section.

3 Quantum Chebyshev's Inequality

We describe our main algorithm for estimating the mean $\mu_{\mathcal{S}}$ of any quantum sampler $\mathcal{S}$, given an upper bound $\Delta_{\mathcal{S}} \geq \phi_{\mathcal{S}}/\mu_{\mathcal{S}}$ (we recall that $\phi_{\mathcal{S}}^2 = \mathbb{E}\left[v(\mathcal{S})^2\right]$ and $\sigma_{\mathcal{S}}/\mu_{\mathcal{S}} \leq \phi_{\mathcal{S}}/\mu_{\mathcal{S}} \leq 1 + \sigma_{\mathcal{S}}/\mu_{\mathcal{S}}$). The two main tools used in this section are the BasicEst algorithm of Corollary 4, and the following lemma on "truncated" means. We recall that $X_{<b}$ (resp. $X_{\geq b}$) is defined from a non-negative random variable X by substituting the outcomes greater or equal to b (resp. less than b) with 0. Note that $X = X_{<b} + X_{\geq b}$ for all $b > 0$.

▶ **Fact 8.** *For any random variable X and real numbers $0 < a \leq b$, we have $\mathbb{E}\left[X_{a,b}\right] \leq \frac{\mathbb{E}\left[X_{a,b}^2\right]}{a}$ and $\mathbb{E}\left[X_{\geq b}\right] \leq \frac{\mathbb{E}\left[X_{\geq b}^2\right]}{b}$.*

▶ **Lemma 9.** *Let X be a non-negative random variable and $\Delta \geq \sqrt{\mathbb{E}\left[X^2\right]}/\mathbb{E}\left[X\right]$. Then, for all $c_1, c_2, M > 0$ such that $c_1 \cdot \mathbb{E}\left[X\right] \leq M \leq c_2 \cdot \mathbb{E}\left[X\right]$, we have*

$$\left(1 - \frac{1}{c_1}\right)\mathbb{E}\left[X\right] \leq \mathbb{E}\left[X_{<M\Delta^2}\right] \leq \mathbb{E}\left[X\right] \quad \text{and} \quad \sqrt{c_1}\cdot\Delta \leq \sqrt{\frac{M\Delta^2}{\mathbb{E}\left[X_{<M\Delta^2}\right]}} \leq \sqrt{c_2\left(1 - \frac{1}{c_1}\right)}\cdot\Delta$$

Proof. The first inequality is a consequence of $\mathbb{E}\left[X_{<M\Delta^2}\right] = \mathbb{E}\left[X\right] - \mathbb{E}\left[X_{\geq M\Delta^2}\right]$ and $0 \leq \mathbb{E}\left[X_{\geq M\Delta^2}\right] \leq \mathbb{E}\left[X_{\geq M\Delta^2}^2\right]/(M\Delta^2) \leq \mathbb{E}\left[X^2\right]/(M\Delta^2) \leq (1/c_1)\cdot\mathbb{E}\left[X\right]$ (using Fact 8). The second inequality is a direct consequence of the left one, and of the hypothesis $c_1 \cdot \mathbb{E}\left[X\right] \leq M \leq c_2 \cdot \mathbb{E}\left[X\right]$. ◀

Our mean estimation algorithm works in two stages. We first compute a rough estimate $M \in [2\mu_{\mathcal{S}}, 2500\mu_{\mathcal{S}}]$ with $\widetilde{\mathcal{O}}\left(\Delta_{\mathcal{S}} \cdot \log(H/L)\right)$ quantum samples (where $0 < L < \mu_{\mathcal{S}} < H$ are known bounds on $\mu_{\mathcal{S}}$). Then, we improve the accuracy of the estimate to any value ϵ, at extra cost $\widetilde{\mathcal{O}}\left(\Delta_{\mathcal{S}}/\epsilon^{3/2}\right)$.

▶ **Theorem 10.** *If $\Delta_{\mathcal{S}} \geq \phi_{\mathcal{S}}/\mu_{\mathcal{S}}$ and $L < \mu_{\mathcal{S}} < H$ then the output $\widetilde{\mu}_{\mathcal{S}}$ of Algorithm 2 satisfies $|\widetilde{\mu}_{\mathcal{S}} - \mu_{\mathcal{S}}| \leq \epsilon\mu_{\mathcal{S}}$ with probability $1 - \delta$. Moreover, for any $\Delta_{\mathcal{S}}, L, H$ it satisfies $\widetilde{\mu}_{\mathcal{S}} \leq (1 + 2\pi)^2\mu_{\mathcal{S}}$ with probability $1 - \delta$. The number of quantum samples used by the algorithm is $\mathcal{O}\left(\Delta_{\mathcal{S}} \cdot \left(\log\left(\frac{H}{L}\right)\log\left(\frac{\log(H/L)}{\delta}\right) + \epsilon^{-3/2}\log\left(\frac{1}{\delta}\right)\right)\right)$.*

Proof. Assume that $\Delta_{\mathcal{S}} \geq \phi_{\mathcal{S}}/\mu_{\mathcal{S}}$ and $L < \mu_{\mathcal{S}} < H$. We denote $p = (M\Delta_{\mathcal{S}}^2)^{-1} \cdot \mathbb{E}\left[v(\mathcal{S})_{<M\Delta_{\mathcal{S}}^2}\right]$. By Lemma 9, if $M \geq 2500\mu_{\mathcal{S}}$ then $25\Delta_{\mathcal{S}} \leq \frac{1}{2\sqrt{p}}$, and if $2\mu_{\mathcal{S}} \leq M \leq 4\mu_{\mathcal{S}}$ then $25\Delta_{\mathcal{S}} > \frac{8}{\sqrt{p}}$. Therefore, by Corollary 4, with probability $1-\delta'$, the value $\widetilde{p}$ computed at Step 2.(b) is equal to 0 when $M \geq 2500\mu_{\mathcal{S}}$, and is different from 0 when $2\mu_{\mathcal{S}} \leq M \leq 4\mu_{\mathcal{S}}$. Thus, the first time Step 2.(b) of Algorithm 2 computes $\widetilde{p} \neq 0$ happens for $M \in [2\mu_{\mathcal{S}}, 2500\mu_{\mathcal{S}}]$, with probability at least $(1-\delta')^{1+\log(4H/(2\mu_{\mathcal{S}}))} > 1-\delta/2$.

Consequently, we can assume that Step 4 is executed with $M \in [2\mu_{\mathcal{S}}, 2500\mu_{\mathcal{S}}]$, and we let $M' = M/\epsilon$. According to Lemma 9 we have $(1-\epsilon/2)\mu_{\mathcal{S}} \leq \mathbb{E}\left[v(\mathcal{S})_{<M'\Delta_{\mathcal{S}}^2}\right] \leq \mu_{\mathcal{S}}$ and $35^2\epsilon^{-3/2}\Delta_{\mathcal{S}} \geq \frac{8}{(\epsilon/2)\sqrt{q}}$, where $q = (M'\Delta_{\mathcal{S}}^2)^{-1} \cdot \mathbb{E}\left[v(\mathcal{S})_{<M'\Delta_{\mathcal{S}}^2}\right]$. Thus, according to Corollary 4, the value $\widetilde{q}$ satisfies $|\widetilde{q}-q| \leq (\epsilon/2)q$ with probability $1-\delta/2$. Using the triangle inequality, it implies $|(\epsilon^{-1}M\Delta_{\mathcal{S}}^2)\cdot\widetilde{q} - \mu_{\mathcal{S}}| \leq \epsilon\mu_{\mathcal{S}}$.

If $L \geq \mu_{\mathcal{S}}$ this may only increase the probability to stop at Step 3 and output $\widetilde{\mu}_{\mathcal{S}} = 0$. If Step 4 is executed we still have $\widetilde{\mu}_{\mathcal{S}} \leq (1+2\pi)^2\mu_{\mathcal{S}}$ with probability $1-\delta$, as a consequence of Corollary 4. ◀

In the full version [31], we improve Step 4 of Algorithm 2 to obtain the following result with (nearly) optimal dependence on ϵ.

▶ **Theorem 11.** *There is an algorithm that, given a sampler $\mathcal{S}$, an integer $\Delta_{\mathcal{S}}$, two values $0 < L < H$, and two reals $0 < \epsilon, \delta < 1$, outputs an estimate $\widetilde{\mu}_{\mathcal{S}}$ of $\mu_{\mathcal{S}}$. If $\Delta_{\mathcal{S}} \geq \phi_{\mathcal{S}}/\mu_{\mathcal{S}}$ and $L < \mu_{\mathcal{S}} < H$, it satisfies $|\widetilde{\mu}_{\mathcal{S}} - \mu_{\mathcal{S}}| \leq \epsilon\mu_{\mathcal{S}}$ with probability $1-\delta$. Moreover, for any $\Delta_{\mathcal{S}}, L, H$ it satisfies $\widetilde{\mu}_{\mathcal{S}} \leq (1+2\pi)^2\mu_{\mathcal{S}}$ with probability $1-\delta$. The number of quantum samples used by the algorithm is $\mathcal{O}\left(\Delta_{\mathcal{S}} \cdot \left(\log\left(\frac{H}{L}\right)\log\left(\frac{\log(H/L)}{\delta}\right) + \epsilon^{-1}\log^{3/2}(\Delta_{\mathcal{S}})\log\left(\frac{\log\Delta_{\mathcal{S}}}{\delta}\right)\right)\right)$.*

In Section 4, we describe an $\Omega((\Delta_{\mathcal{S}}-1)/\epsilon)$ lower bound for this mean estimation problem. Before, we present three kinds of generalizations of the above algorithms.

- **Higher moments.** Given an upper-bound $\Delta_{\mathcal{S}}^2 \geq \left(\mathbb{E}\left[v(\mathcal{S})^k\right]/\mathbb{E}\left[v(\mathcal{S})\right]^k\right)^{1/(k-1)}$ on the relative moment of order $k \geq 2$, one can easily generalize Facts 8, Lemma 9 and Theorem 11 to show that $\mu_{\mathcal{S}}$ can be estimated using $\widetilde{\mathcal{O}}\left(\Delta_{\mathcal{S}} \cdot \epsilon^{-1/(2(k-1))}\log(H/L)\log(1/\delta)\right)$ quantum samples.
- **Implicit upper bound on $\boldsymbol{\phi_{\mathcal{S}}/\mu_{\mathcal{S}}}$.** If instead of an explicit value $\Delta_{\mathcal{S}} \geq \phi_{\mathcal{S}}/\mu_{\mathcal{S}}$ we are given a non-increasing function f such that $f(\mu_{\mathcal{S}}) \geq \phi_{\mathcal{S}}/\mu_{\mathcal{S}}$, we can still estimate the mean $\mu_{\mathcal{S}}$ using $\widetilde{\mathcal{O}}\left(f(\mu_{\mathcal{S}}/c)\cdot\epsilon^{-1}\log(H/L)\log(1/\delta)\right)$ quantum samples, where $c > 1$ is an absolute constant. The proof is deferred to the full version [31] (it crucially uses the Markov-like inequality "$\widetilde{\mu}_{\mathcal{S}} \leq (1+2\pi)^2\mu_{\mathcal{S}}$" of Corollary 4 and Remark 6).
- **Time complexity and variable-time samplers.** The *time complexity* (number of quantum gates) of all above algorithms is essentially equal to the number of quantum samples multiplied by the time complexity $T_{max}(\mathcal{S})$ of the considered sampler. However, $T_{max}(\mathcal{S})$ is often much larger than the more desirable *ℓ_2-average running time* $T_{\ell_2}(\mathcal{S})$ defined by Ambainis [3] in the context of variable-time algorithms, where some branches of computation may stop earlier than the others. In the full version [31], we develop a new technique called *variable-time amplitude estimation* that improves the *time* complexity of our algorithm to $\widetilde{\mathcal{O}}\left(\Delta_{\mathcal{S}}\cdot\epsilon^{-2}T_{\ell_2}(\mathcal{S})\cdot\log^4(T_{max}(\mathcal{S}))\log(H/L)\log(1/\delta)\right)$.

The last two results are combined together in the algorithm of Theorem 21 to approximate the number of triangles in any graph.

4 Optimality and Separation Results

Using a result due to Nayak and Wu [50] on approximate counting, we can show a corresponding lower bound to Theorem 11 already in the simple case of Bernoulli variables. For this purpose, we define that an algorithm $\mathcal{A}$ solves the *Mean Estimation problem for parameters* ϵ, Δ if, for any sampler $\mathcal{S}$ satisfying $\phi_{\mathcal{S}}/\mu_{\mathcal{S}} \in [\Delta, 4\Delta]$ (the constant 4 is arbitrary), it outputs a value $\widetilde{\mu}_{\mathcal{S}}$ satisfying $|\widetilde{\mu}_{\mathcal{S}} - \mu_{\mathcal{S}}| \leq \epsilon\mu_{\mathcal{S}}$ with probability 2/3.

▶ **Theorem 12.** *Any algorithm solving the Mean Estimation problem for parameters* $0 < \epsilon < 1/5$ *and* $\Delta > 1$ *on the sample space* $\Omega = \{0,1\}$ *must use* $\Omega\left((\Delta-1)/\epsilon\right)$ *quantum samples.*

Proof. Consider an algorithm $\mathcal{A}$ solving the Mean Estimation problem for parameters $0 < \epsilon < 1/5$, $\Delta > 1$ using N quantum samples. Take two integers $0 < t < n$ large enough such that $\sqrt{2}\Delta \leq \sqrt{n/t} \leq 4\Delta$ and $\epsilon t > 1$. For any oracle $\mathcal{O} : \{1, \ldots, n\} \to \{0,1\}$, define the quantum sampler $\mathcal{S}_{\mathcal{O}}(|0\rangle|0\rangle) = \frac{1}{\sqrt{n}} \sum_{i\in[n]} |i\rangle|\mathcal{O}(i)\rangle$ and let $t_{\mathcal{O}} = |\{i \in [n] : \mathcal{O}(i) = 1\}|$. Observe that $\mu_{\mathcal{S}_{\mathcal{O}}} = \phi^2_{\mathcal{S}_{\mathcal{O}}} = t_{\mathcal{O}}/n$, and one quantum sample from $\mathcal{S}_{\mathcal{O}}$ can be implemented with one quantum query to $\mathcal{O}$.

According to [50, Corollary 1.2], any algorithm that can distinguish $t_{\mathcal{O}} = t$ from $t_{\mathcal{O}} = \lceil (1+4\epsilon)t \rceil$ makes $\Omega\left(\sqrt{n/(\epsilon t)} + \sqrt{t(n-t)/(\epsilon t)}\right) = \Omega\left((\sqrt{n/t} - 1)/\epsilon\right) = \Omega\left((\Delta-1)/\epsilon\right)$ quantum queries to $\mathcal{O}$. However, given the promise that $t_{\mathcal{O}} = t$ or $t_{\mathcal{O}} = \lceil (1+4\epsilon)t \rceil$ we can use $\mathcal{A}$ with input $\mathcal{S}_{\mathcal{O}}$, ϵ, Δ to distinguish between the two cases using N samples, that is N queries to $\mathcal{O}$. Indeed, $\phi_{\mathcal{S}_{\mathcal{O}}}/\mu_{\mathcal{S}_{\mathcal{O}}} = \sqrt{n/t_{\mathcal{O}}} \in [\Delta, 4\Delta]$ for such samplers (since $\lceil (1+4\epsilon)t \rceil \leq (1+5\epsilon)t \leq 2t$). Thus, $\mathcal{A}$ must use $N = \Omega\left((\Delta-1)/\epsilon\right)$ quantum samples. ◀

One may wonder whether the quantum speed-up presented in this paper holds if we only have access to copies of a quantum state $\sum_{x\in\Omega} \sqrt{d(x)}|\psi_x\rangle|x\rangle$ (instead of access to a unitary $\mathcal{S}$ preparing it). Below we answer this question negatively. For this purpose, we define that an algorithm $\mathcal{A}$ solves the *state-based Mean Estimation problem for parameters* ϵ, Δ if, using access to some copies of an unknown state $|d\rangle = \sum_{x\in\Omega} \sqrt{d(x)}|x\rangle$ satisfying $\phi_d/\mu_d \in [\Delta, 4\Delta]$ (where $\mu_d = \sum_x d(x)x$ and $\phi_d^2 = \sum_x d(x)x^2$), it outputs a value $\widetilde{\mu}_d$ satisfying $|\widetilde{\mu}_d - \mu_d| \leq \epsilon\mu_d$ with probability 2/3.

▶ **Lemma 13.** *Consider two distributions* d, d' *represented by the states* $|d\rangle = \sum_{x\in\Omega} \sqrt{d(x)}|x\rangle$ *and* $|d'\rangle = \sum_{x\in\Omega} \sqrt{d'(x)}|x\rangle$. *The smallest integer* T *needed to be able to discriminate* $|d\rangle^{\otimes T}$ *and* $|d'\rangle^{\otimes T}$ *with success probability* 2/3 *satisfies* $T \geq \frac{\ln(9/8)}{D(d||d')}$, *where* $D(d||d')$ *is the KL-divergence from* d *to* d'.

Proof. According to Helstrom's bound [33] the best success probability to discriminate two states $|\psi\rangle$ and $|\phi\rangle$ is $\frac{1}{2}(1 + \sqrt{1 - |\langle\psi|\phi\rangle|^2})$. Consequently, T must satisfy $\frac{1}{2}(1 + \sqrt{1 - \langle d|d'\rangle^{2T}}) \geq 2/3$, which implies

$$T \geq \frac{\ln(9/8)}{-\ln(\langle d|d'\rangle^2)} = \frac{\ln(9/8)}{-2\ln\left(\sum_x d(x)\sqrt{d'(x)/d(x)}\right)} \geq \frac{\ln(9/8)}{\sum_x d(x)\ln\left(d(x)/d'(x)\right)} = \frac{\ln(9/8)}{D(d||d')}$$

where we used the concavity of the $-\ln$ function. ◀

▶ **Theorem 14.** *Any algorithm solving the state-based Mean Estimation problem for parameters* $0 < \epsilon < 1/100$ *and* $\Delta > 1$ *on the sample space* $\Omega = \{0,1\}$ *must use* $\Omega\left((\Delta^2-1)/\epsilon^2\right)$ *copies of the input state.*

Proof. Consider an algorithm $\mathcal{A}$ solving the state-based Mean Estimation problem for parameters $0 < \epsilon < 1/100$, $\Delta > 1$ using N copies of the input state. Given any $|d\rangle = \sqrt{1-p}|0\rangle + \sqrt{p}|1\rangle$ with $\phi_d/\mu_d \in [\sqrt{6}\Delta, \sqrt{8}\Delta]$ (notice that $\mu_d = \phi_d^2 = p$ and $1-p \geq 5/6 \geq 12\epsilon$), we show how to construct a state $|d'\rangle = \sqrt{1-p'}|0\rangle + \sqrt{p'}|1\rangle$ such that

$$(1)\ (1+4\epsilon)\mu_d < \mu_{d'} < (1+24\epsilon)\mu_d\ ;\ \ (2)\ \phi_{d'}/\mu_{d'} \in [\Delta, 4\Delta]\ ;\ \ (3)\ D(d||d') \leq (12\epsilon)^2/(\Delta^2-1).$$

It is clear that $\mathcal{A}$ can be used to discriminate two such states. On the other hand, according to Lemma 13, any such algorithm muse use $N = \Omega\left(1/D(d||d')\right) = \Omega\left((\Delta^2-1)/\epsilon^2\right)$ copies of the input state.

The construction of d' is adapted from [18, Section 7]. We set $p' = pe^{\alpha(1-p)}/\psi$ where $\alpha = 12\epsilon/(1-p) < 1$ and $\psi = (1-p)e^{-\alpha p} + pe^{\alpha(1-p)}$ (so that $1-p' = (1-p)e^{-\alpha p}/\psi$). We let $\dot{\psi}$ (resp. $\ddot{\psi}$) denote the first (resp. second) derivative of ψ with respect to α. A simple calculation shows that $\mu_{d'} - \mu_d = \dot{\psi}/\psi$ and $D(d||d') = \ln\psi$. Moreover, $\sigma_{d'}^2 = \mathbb{E}_{x\sim d'}\left[(x-\mu_{d'})^2\right] = \mathbb{E}_{x\sim d'}\left[(x-\mu_d)^2\right] + 2(\mu_d - \mu_{d'})\mathbb{E}_{x\sim d'}\left[x-\mu_d\right] + (\mu_d-\mu_{d'})^2 = \mathbb{E}_{x\sim d}\left[(x-p)^2 e^{\alpha(x-p)-\ln\psi}\right] - (\mu_d-\mu_{d'})^2 = \ddot{\psi}/\psi - (\dot{\psi}/\psi)^2$.

Since $\psi = \mathbb{E}_{x\sim d}\left[e^{\alpha(x-p)}\right]$, it can be deduced from the standard inequality $1+u+u^2/3 \leq e^u \leq 1+u+u^2$ (when $|u| \leq 1$) that $1 \leq 1 + \frac{p(1-p)}{3}\cdot\alpha^2 \leq \psi \leq 1 + p(1-p)\cdot\alpha^2 \leq 2$. Consequently, $\frac{2p(1-p)}{3}\cdot\alpha \leq \dot{\psi} \leq 2p(1-p)\cdot\alpha$ and $\frac{2p(1-p)}{3} \leq \ddot{\psi} \leq 2p(1-p)$. It implies that $4\epsilon p \leq \mu_{d'} - \mu_d \leq 24\epsilon p$ and $p(1-p)/3 - (24\epsilon p)^2 \leq \sigma_{d'}^2 \leq 2p(1-p)$. Thus, $(1+4\epsilon)\mu_d \leq \mu_{d'} \leq (1+24\epsilon)\mu_d \leq \sqrt{2}\mu_d$ and $\frac{1}{6}\sigma_d^2/\mu_d^2 - (24\epsilon/\sqrt{2})^2 \leq \sigma_{d'}^2/\mu_{d'}^2 \leq 2\sigma_d^2/\mu_d^2$. Since $\sigma_{d'}^2/\mu_{d'}^2 = \phi_{d'}^2/\mu_{d'}^2 - 1$ and $\phi_d/\mu_d \in [\sqrt{6}\Delta, \sqrt{8}\Delta]$, we obtain that $\Delta \leq \frac{1}{\sqrt{6}}\phi_d/\mu_d \leq \phi_{d'}/\mu_{d'} \leq \sqrt{2}\phi_d/\mu_d \leq 4\Delta$. Finally, $D(d||d') = \ln\psi \leq p(1-p)\cdot\alpha^2 = (12\epsilon)^2 p/(1-p) \leq (12\epsilon)^2/(\Delta^2-1)$. ◀

▶ Remark 15. An intermediate version of Theorem 12 can be deduced from Theorem 14, when $\mathcal{S}$ is accessed via the *reflection oracle* $\mathcal{O}_{\mathcal{S}} = I - 2\mathcal{S}(|0\rangle|0\rangle)(\langle 0|\langle 0|)\mathcal{S}^{-1}$ only (observe that this is the case for our algorithms). Indeed, according to [38, Theorem 4], for any algorithm performing q queries to a reflection oracle $\mathcal{O} = I - 2|\phi\rangle\langle\phi|$, it is possible to remove the queries to $\mathcal{O}$ by using $\sim q^2$ copies of $|\phi\rangle$ instead.

5 Applications

We describe two applications of the Quantum Chebyshev Inequality. The first one (Section 5.1) concerns the computation of the frequency moments F_k of order $k \geq 3$ in the streaming model. We design a P-pass algorithm with quantum memory M satisfying a tradeoff of $P^2M = \widetilde{\mathcal{O}}\left(n^{1-2/k}\right)$, whereas the best algorithm with classical memory requires $PM = \Theta(n^{1-2/k})$. We then study (Section 5.2) the edge and triangle counting problems in the general graph model with quantum query access. We describe nearly optimal algorithms that approximate these parameters quadratically faster than in the classical query model.

5.1 Frequency Moments in the Multi-Pass Streaming Model

In the streaming model with update (*turnstile model*), the input is a vector $x \in \mathbb{R}^n$ obtained through a stream $\vec{u} = u_1, u_2, \ldots$ of updates. Initially, $x(0) = (0, \ldots, 0)$, and each $u_j = (i, \lambda) \in [n] \times \mathbb{R}$ modifies the i-th coordinate of $x(j)$ by adding λ to it. The goal of a *streaming algorithm* $\mathcal{T}$ is to output, at the end of the stream, some function of the final vector x while minimizing the number $M \ll n$ of memory cells. In the *multi-pass* model, the same stream is repeated for a certain number P of passes, before the algorithm outputs its result.

Input: a stream $\vec{u}$, an integer $k \geq 3$, a real $\widetilde{F}_2$, an approximation parameter $0 < \epsilon < 1$.
Output: an estimate $\widetilde{F}_k$ of the frequency moment of order k of $\vec{u}$.

1. Compute $i \in [n]$ using the streaming algorithm of Theorem 16 with input $\vec{u}$, $\epsilon/4$, $\widetilde{F}_2$.
2. Compute x_i using a second pass over $\vec{u}$.
3. Output $\widetilde{F}_2 \cdot |x_i|^{k-2}$.

Estimator 3 Frequency moment F_k of a stream.

The *frequency moment of order k* is defined, for the final vector $x = (x_1, \ldots, x_n)$, as $F_k(x) = \sum_{i \in [n]} |x_i|^k$. The problem of approximating F_k when $k \geq 3$ has been addressed first with the AMS algorithm [2], that uses $\mathcal{O}\left(n^{1-1/k}\right)$ classical memory cells in the insertion-only model (where $u_j \in [n] \times \mathbb{R}^+$). A series of works in the turnstile model culminated in optimal one-pass algorithms with memory $\Theta\left(n^{1-2/k}\right)$ [44, 26], and nearly optimal P-pass algorithms with memory $\widetilde{\Theta}\left(n^{1-2/k}/P\right)$ [46, 4, 57]. In the quantum setting, Montanaro [48] obtained a small improvement in terms of the approximation parameter ϵ only.

Our algorithm relies on a classical procedure for ℓ_2 *sampling*. Given $x \in \mathbb{R}^n$, we let $D_{q,x}$ denotes the ℓ_q *distribution* that returns $i \in [n]$ with probability $\frac{|x_i|^q}{F_q(x)}$. One can observe that the (suboptimal) AMS algorithm [2] essentially samples $i \sim D_{1,x}$ and computes $F_1 \cdot |x_i|^{k-1}$. This is an unbiased estimator for $F_k(x)$ with variance $\mathcal{O}\left(n^{1-1/k} F_k(x)^2\right)$ (thus requiring to compute $\mathcal{O}\left(n^{1-1/k}\right)$ samples in one pass). Instead, we base our algorithm on the estimator $F_2(x) \cdot |x_i|^{k-2}$ where $i \sim D_{2,x}$. It reduces the variance to $\mathcal{O}\left(n^{1-2/k} F_k(x)^2\right)$ [46], but it requires a procedure for ℓ_2 sampling. To this end, we use the following algorithm from [4] to sample from an (ϵ, δ)-*approximator* to $D_{2,x}$ (meaning that each $i \in [n]$ is sampled with a probability p_i satisfying $(1-\epsilon)\frac{|x_i|^2}{F_2(x)} - \delta \leq p_i \leq (1+\epsilon)\frac{|x_i|^2}{F_2(x)} + \delta$).

▶ **Theorem 16** ([4]). *There is a randomized streaming algorithm that, given a stream $\vec{u}$ with final vector x, a real $0 < \epsilon < 1/3$ and a value $\widetilde{F}_2$ such that $|\widetilde{F}_2 - F_2(x)| \leq (1/2) \cdot F_2(x)$, outputs a value $i \in [n]$ that is distributed according to an (ϵ, n^{-2})-approximator to $D_{2,x}$. The algorithm uses $M = \mathcal{O}\left(\epsilon^{-2} \log^3 n\right)$ classical memory cells. Moreover, each element of the stream is processed in time $T_{upd} = \mathcal{O}\left(\epsilon^{-1} \log n\right)$, and the output is computed in time $T_{rec} = \mathcal{O}\left(\epsilon^{-1} n \log n\right)$ after the last element is received.*

▶ **Proposition 17** ([46, 4]). *If we let X denote the output random variable of Estimator 3, then $\mathbb{E}[X] = (1 \pm \epsilon/2) F_k$ and $\mathrm{Var}[X] \leq \mathcal{O}\left(n^{1-2/k} F_k^2\right)$, when $|\widetilde{F}_2 - F_2| \leq (\epsilon/4) \cdot F_2$.*

It is known that any deterministic computation can be made reversible, and therefore implemented by a unitary map with a limited overhead on the time and space complexities [9]. Nonetheless, implementing naively the reverse computation of a streaming algorithm would require processing the same stream but in the *reverse* direction, which may not be always possible. This motivates our specific notion of *reversible streaming algorithms*. We say that a streaming algorithm $\mathcal{T}$ with memory size M is *reversible* if there exists a streaming algorithm $\mathcal{T}^{-1}$ with memory size M such that each computational steps of $\mathcal{T}$ and $\mathcal{T}^{-1}$ are reversible, and in addition each pass of $\mathcal{T}$ can be undone by one pass of $\mathcal{T}^{-1}$ in the *same* direction. In the full version [31] we show how to make the algorithm of Theorem 16 reversible (our result is in fact more general and holds for any *linear sketch* streaming algorithm). We combine the quantum sampler that is obtained from this result with the Quantum Chebyshev Inequality (Theorem 11) to obtain the following tradeoff.

▶ **Theorem 18.** *There is a quantum streaming algorithm that, given a stream $\vec{u}$, two integers $P \geq 1$, $k \geq 3$ and an approximation parameter $0 < \epsilon < 1$, outputs an estimate $\widetilde{F}_k$ such that $|\widetilde{F}_k - F_k| \leq \epsilon F_k$ with probability $2/3$. The algorithm uses $\widetilde{\mathcal{O}}\left(n^{1-2/k}/(\epsilon P)^2\right)$ quantum memory cells, and it makes $\widetilde{\mathcal{O}}\left(P \cdot (k \log n + \epsilon^{-1})\right)$ passes over the stream $\vec{u}$.*

Proof. We first compute, in one pass, a value $\widetilde{F}_2$ such that $|\widetilde{F}_2 - F_2| \leq (\epsilon/2)F_2$ with high probability, using [2, 48] for instance. The complexity is absorbed by the final result. Then, using the *reversible* streaming algorithm associated to Estimator 3, we can design a quantum sampler $\mathcal{S}$ using memory $M = \widetilde{\mathcal{O}}\left(\epsilon^{-2} \log^3 n\right)$ such that $\mathcal{S}(|0\rangle|0\rangle) = \sum_{r \in \{0,1\}^M} |r\rangle|\psi_r\rangle|f_r\rangle$ where each $|r\rangle$ corresponds to a different random seed for the linear sketch algorithm of Theorem 16, $|f_r\rangle$ is the output of Estimator 3, and $|\psi_r\rangle$ is some garbage state obtained when making Estimator 3 reversible. According to Proposition 17, we have $\mu_{\mathcal{S}} = (1 \pm \epsilon/2)F_k$ and $\sigma_{\mathcal{S}} \leq \mathcal{O}\left(\sqrt{n^{1-2/k}}F_k\right)$. Moreover one quantum sample can be implemented with two passes over the stream.

We "concatenate" $Q = n^{1-2/k}/P^2$ such samplers and compute the mean $\bar{f} = Q^{-1} \cdot (f_{r_1} + \cdots + f_{r_Q})$ of their results to obtain

$$\bar{\mathcal{S}}(|0\rangle|0\rangle) = \sum\nolimits_{r_1,\ldots,r_Q \in \{0,1\}^M} |r_1, \ldots, r_Q\rangle|\psi_1, \ldots, \psi_Q\rangle|f_{r_1}, \ldots, f_{r_Q}\rangle|\bar{f}\rangle.$$

This sampler satisfies $\mu_{\bar{\mathcal{S}}} = \mu_{\mathcal{S}}$ and $\sigma_{\bar{\mathcal{S}}} = \sigma_{\mathcal{S}}/\sqrt{Q} \leq \mathcal{O}\left(PF_k\right)$, and it requires two passes and memory $\bar{M} = \widetilde{\mathcal{O}}\left(Q \cdot \epsilon^{-2} \log^3 n\right)$ to be implemented. Finally, we approximate F_k by applying Theorem 11 on $\bar{\mathcal{S}}$, which uses $\widetilde{\mathcal{O}}\left(P \cdot (k \log n + \epsilon^{-1})\right)$ quantum samples. ◀

5.2 Approximating Graph Parameters in the Query Model

In this section, we consider the *general graph model* [40, 27] that provides query access to a graph $G = (V, E)$ through the following operations: (1) *degree query* (given $v \in V$, returns the degree d_v of v), (2) *neighbor query* (given $v \in V$ and i, returns the i-th neighbor of v if $i \leq d_v$, and $\perp$ otherwise), and (3) *vertex-pair query* (given $u, v \in V$, indicates if $(u, v) \in E$). This is a combination of the dense graph model (pair queries) and the bounded-degree model (neighbor and degree queries). We refer the reader to [27, Chapter 10] for a more detailed discussion about it. It can be extended to the standard quantum query framework. A quantum degree query is represented as a unitary $\mathcal{O}_{deg}$ such that $\mathcal{O}_{deg}|v\rangle|b\rangle = |v\rangle|y \oplus d_v\rangle$ where $v \in V$ and $y \in \{0,1\}^{\lceil \log n \rceil}$. The quantum neighbor $\mathcal{O}_{neigh}$ and vertex-pair $\mathcal{O}_{pair}$ queries are defined similarly. The *query complexity* of an algorithm in the *quantum general graph model* is the number of times it uses $\mathcal{O}_{deg}$, $\mathcal{O}_{nei}$ or $\mathcal{O}_{pair}$.

In the following, we let n denote the number of vertices, m the number of edges and t the number of triangles in G. We consider the problems of estimating m and t, for which we provide nearly optimal quantum algorithms. The description and analysis of these algorithms is deferred to the full version [31].

Edge counting. In the classical setting, Feige [25] showed that $\Theta(n/(\epsilon\sqrt{m}))$ degree queries are sufficient to compute a factor $(2+\epsilon)$ approximation of m, but no factor $(2-\epsilon)$ approximation can be obtained in sublinear time. Using both degree and neighbor queries, it is possible to compute a factor $(1+\epsilon)$ approximation with $\Theta\left(n/(\sqrt{\epsilon m})\right)$ classical queries [28, 52, 23]. These results were generalized to k-star counting in [30, 23]. In the quantum setting, we prove the following results.

▶ **Theorem 19.** *There is an algorithm that, given query access to any n-vertex graph G with m edges, and an approximation parameter $\epsilon < 1$, outputs an estimate $\widetilde{m}$ of m such that $|\widetilde{m} - m| \leq \epsilon m$ with probability $2/3$. This algorithm performs $\widetilde{\mathcal{O}}\left(\frac{n^{1/2}}{\epsilon m^{1/4}}\right)$ quantum degree and neighbor queries in expectation. Moreover, it does not use vertex-pair queries.*

▶ **Theorem 20.** *Any algorithm that computes an ϵ-approximation of the number m of edges in any n-vertex graph, given query access to it, must use $\Omega\left(\frac{n^{1/2}}{(\epsilon m)^{1/4}} \cdot \log^{-1}(n)\right)$ quantum queries in expectation.*

Triangle counting. In the classical general graph model, the triangle counting problem requires $\widetilde{\Theta}(n/t^{1/3} + \min(m, m^{3/2}/t))$ queries in expectation [21, 22]. This result was generalized to k-clique counting in [24]. In the quantum setting, we prove the following results.

▶ **Theorem 21.** *There is an algorithm that, given query access to any n-vertex graph G with m edges and t triangles, and an approximation parameter $\epsilon < 1$, outputs an estimate $\widetilde{t}$ of t such that $|\widetilde{t} - t| \leq \epsilon t$ with probability $2/3$. This algorithm performs $\widetilde{\mathcal{O}}\left(\left(\frac{\sqrt{n}}{t^{1/6}} + \frac{m^{3/4}}{\sqrt{t}}\right) \cdot poly(1/\epsilon)\right)$ quantum queries in expectation.*

▶ **Theorem 22.** *Any algorithm that computes an ϵ-approximation to the number t of triangles in any n-vertex graph with m vertices, given query access to it, must use $\Omega\left(\left(\frac{\sqrt{n}}{t^{1/6}} + \frac{m^{3/4}}{\sqrt{t}}\right) \cdot \log^{-1}(n)\right)$ quantum queries in expectation.*

6 Open Questions

Is it possible to improve the complexity of our main result (Theorem 11) to $\mathcal{O}(\Delta_{\mathcal{S}}/\epsilon)$ exactly? Can we generalize it to sample spaces with negative values? What are other possible applications? Two promising problems are minimum spanning tree weight [16] and arbitrary subgraph counting [24, 6].

References

1. D. Aharonov and A. Ta-Shma. Adiabatic Quantum State Generation. *SIAM Journal on Computing*, 37(1):47–82, 2007.
2. N. Alon, Y. Matias, and M. Szegedy. The Space Complexity of Approximating the Frequency Moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.
3. A. Ambainis. Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations. Technical report, arXiv.org, 2010. `arXiv:1010.4458`.
4. A. Andoni, R. Krauthgamer, and K. Onak. Streaming Algorithms via Precision Sampling. In *Proceedings of the 52nd Symposium on Foundations of Computer Science*, FOCS '11, pages 363–372, 2011.
5. S. Arunachalam and R. de Wolf. Optimal Quantum Sample Complexity of Learning Algorithms. In *Proceedings of the 32nd Computational Complexity Conference*, CCC '17, pages 25:1–25:31, 2017.
6. S. Assadi, M. Kapralov, and S. Khanna. A Simple Sublinear-Time Algorithm for Counting Arbitrary Subgraphs via Edge Sampling. In *Proceedings of the 10th Conference on Innovations in Theoretical Computer Science*, ITCS '19, pages 6:1–6:20, 2019.
7. C. Badescu, R. O'Donnell, and J. Wright. Quantum state certification. Technical report, arXiv.org, 2017. `arXiv:1708.06002`.
8. T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White. Testing Closeness of Discrete Distributions. *Journal of the ACM*, 60(1):4:1–4:25, 2013.

9 C. Bennett. Time/Space Trade-Offs for Reversible Computation. *SIAM Journal on Computing*, 18(4):766–776, 1989.

10 G. Brassard, F. Dupuis, S. Gambs, and A. Tapp. An optimal quantum algorithm to approximate the mean and its application for approximating the median of a set of points over an arbitrary distance. Technical report, arXiv.org, 2011. `arXiv:1106.4267`.

11 G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. *Quantum Computation and Quantum Information: A Millennium Volume*, 1:53–74, 2002.

12 S. Bravyi, A. W. Harrow, and A. Hassidim. Quantum Algorithms for Testing Properties of Distributions. *IEEE Transactions on Information Theory*, 57(6):3971–3981, 2011.

13 C. L. Canonne, I. Diakonikolas, D. M. Kane, and A. Stewart. Testing conditional independence of discrete distributions. In *Proceedings of the 50th Symposium on Theory of Computing*, STOC '18, pages 735–748, 2018.

14 A. Chakrabarti, G. Cormode, R. Kondapally, and A. McGregor. Information Cost Tradeoffs for Augmented Index and Streaming Language Recognition. *SIAM Journal on Computing*, 42(1):61–83, 2013.

15 S. Chan, I. Diakonikolas, P. Valiant, and G. Valiant. Optimal Algorithms for Testing Closeness of Discrete Distributions. In *Proceedings of the 25th Symposium on Discrete Algorithms*, SODA '14, pages 1193–1203, 2014.

16 B. Chazelle, R. Rubinfeld, and L. Trevisan. Approximating the Minimum Spanning Tree Weight in Sublinear Time. *SIAM Journal on Computing*, 34(6):1370–1379, 2005.

17 A. N. Chowdhury and R. D. Somma. Quantum Algorithms for Gibbs Sampling and Hitting-time Estimation. *Quantum Information and Computation*, 17(1-2):41–64, 2017.

18 P. Dagum, R. Karp, M. Luby, and S. Ross. An Optimal Algorithm for Monte Carlo Estimation. *SIAM Journal on Computing*, 29(5):1484–1496, 2000.

19 N. Destainville, B. Georgeot, and O. Giraud. Quantum Algorithm for Exact Monte Carlo Sampling. *Physical Review Letters*, 104:250502, 2010.

20 M. Dyer, A. Frieze, and R. Kannan. A Random Polynomial-time Algorithm for Approximating the Volume of Convex Bodies. *Journal of the ACM*, 38(1):1–17, 1991.

21 T. Eden, A. Levi, and D. Ron. Approximately Counting Triangles in Sublinear Time. Technical Report TR15-046, ECCC, 2015.

22 T. Eden, A. Levi, D. Ron, and C. Seshadhri. Approximately Counting Triangles in Sublinear Time. *SIAM Journal on Computing*, 46(5):1603–1646, 2017.

23 T. Eden, D. Ron, and C. Seshadhri. Sublinear Time Estimation of Degree Distribution Moments: The Degeneracy Connection. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming*, ICALP '17, pages 7:1–7:13, 2017.

24 T. Eden, D. Ron, and C. Seshadhri. On Approximating the Number of K-cliques in Sublinear Time. In *Proceedings of the 50th Symposium on Theory of Computing*, STOC '18, pages 722–734, 2018.

25 U. Feige. On Sums of Independent Random Variables with Unbounded Variance and Estimating the Average Degree in a Graph. *SIAM Journal on Computing*, 35(4):964–984, 2006.

26 S. Ganguly. Taylor Polynomial Estimator for Estimating Frequency Moments. In *Proceedings of the 42nd International Colloquium on Automata, Languages and Programming*, ICALP '15, pages 542–553, 2015.

27 O. Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.

28 O. Goldreich and D. Ron. Approximating Average Parameters of Graphs. *Random Structures & Algorithms*, 32(4):473–493, 2008.

29 O. Goldreich and D. Ron. On Testing Expansion in Bounded-Degree Graphs. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 68–75. Springer-Verlag, 2011.

30 M. Gonen, D. Ron, and Y. Shavitt. Counting Stars and Other Small Subgraphs in Sublinear-Time. *SIAM Journal on Discrete Mathematics*, 25(3):1365–1411, 2011.

31 Y. Hamoudi and F. Magniez. Quantum Chebyshev's Inequality and Applications. Technical report, arXiv.org, 2019. `arXiv:1807.06456`.

32 S. Heinrich. Quantum Summation with an Application to Integration. *Journal of Complexity*, 18(1):1–50, 2002.

33 C. W. Helstrom. Quantum detection and estimation theory. *Journal of Statistical Physics*, 1(2):231–252, June 1969.

34 R. Jain and A. Nayak. The Space Complexity of Recognizing Well-Parenthesized Expressions in the Streaming Model: the Index Function Revisited. *IEEE Transactions on Information Theory*, 60(10):6646–6668, 2014.

35 M. Jerrum and A. Sinclair. The Markov Chain Monte Carlo Method: An Approach to Approximate Counting and Integration. In *Approximation Algorithms for NP-hard Problems*, chapter 12, pages 482–520. PWS Publishing, 1996.

36 M. Jerrum, A. Sinclair, and E. Vigoda. A Polynomial-time Approximation Algorithm for the Permanent of a Matrix with Nonnegative Entries. *Journal of the ACM*, 51(4):671–697, 2004.

37 M. R. Jerrum, L. G. Valiant, and V. V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

38 Z. Ji, Y.-K. Liu, and F. Song. Pseudorandom Quantum States. In *Advances in Cryptology*, CRYPTO '18, pages 126–152, 2018.

39 R. M. Karp and M. Luby. Monte-Carlo algorithms for enumeration and reliability problems. In *Proceedings of the 24th Symposium on Foundations of Computer Science*, FOCS '83, pages 56–64, 1983.

40 T. Kaufman, M. Krivelevich, and D. Ron. Tight Bounds for Testing Bipartiteness in General Graphs. *SIAM Journal on Computing*, 33(6):1441–1483, 2004.

41 E. Knill, G. Ortiz, and R. D. Somma. Optimal quantum measurements of expectation values of observables. *Physical Review A*, 75:012328, 2007.

42 F. Le Gall. Exponential Separation of Quantum and Classical Online Space Complexity. *Theory of Computing Systems*, 45(2):188–202, 2009.

43 T. Li and X. Wu. Quantum query complexity of entropy estimation. Technical report, arXiv.org, 2017. `arXiv:1710.06025`.

44 Y. Li and D. P. Woodruff. A Tight Lower Bound for High Frequency Moment Estimation with Small Error. In *Proceedings of the Workshop on Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, APPROX/RANDOM '13, pages 623–638, 2013.

45 F. Magniez, C. Mathieu, and A. Nayak. Recognizing Well-Parenthesized Expressions in the Streaming Model. *SIAM Journal on Computing*, 43(6):1880–1905, 2014.

46 M. Monemizadeh and D. P. Woodruff. 1-pass Relative-error Lp-sampling with Applications. In *Proceedings of the 21st Symposium on Discrete Algorithms*, SODA '10, pages 1143–1160, 2010.

47 A. Montanaro. Quantum speedup of Monte Carlo methods. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 471(2181), 2015.

48 A. Montanaro. The quantum complexity of approximating the frequency moments. *Quantum Information and Computation*, 16:1169–1190, 2016.

49 A. Nayak and D. Touchette. Augmented Index and Quantum Streaming Algorithms for DYCK(2). In *Proceedings of the 32nd Conference on Computational Complexity*, CCC '17, pages 23:1–23:21, 2017.

50 A. Nayak and F. Wu. The Quantum Query Complexity of Approximating the Median and Related Statistics. In *Proceedings of the 31st Symposium on Theory of Computing*, STOC '99, pages 384–393, 1999.

51 D. Poulin and P. Wocjan. Sampling from the Thermal Quantum Gibbs State and Evaluating Partition Functions with a Quantum Computer. *Physical Review Letters*, 103:220502, 2009.

52 C. Seshadhri. A simpler sublinear algorithm for approximating the triangle count. Technical report, arXiv.org, 2015. `arXiv:1505.01927`.

53 K. Temme, T. J. Osborne, K. Vollbrecht, D. Poulin, and F. Verstraete. Quantum Metropolis Sampling. *Nature*, 471:87, 2011.

54 D. Štefankovič, S. Vempala, and E. Vigoda. Adaptive Simulated Annealing: A Near-optimal Connection Between Sampling and Counting. *Journal of the ACM*, 56(3):18:1–18:36, 2009.

55 P. Wocjan and A. Abeyesinghe. Speedup via quantum sampling. *Physical Review A*, 78:042336, 2008.

56 P. Wocjan, C.-F. Chiang, D. Nagaj, and A. Abeyesinghe. Quantum algorithm for approximating partition functions. *Physical Review A*, 80:022340, 2009.

57 D. P. Woodruff and Q. Zhang. Tight Bounds for Distributed Functional Monitoring. In *Proceedings of the 44th Symposium on Theory of Computing*, STOC '12, pages 941–960, 2012.

Retracting Graphs to Cycles

Samuel Haney
Duke University, Durham, NC, USA
shaney@cs.duke.edu

Mehraneh Liaee
Northeastern University, Boston, MA, USA
mehraneh@ccs.neu.edu

Bruce M. Maggs
Duke University, Durham, NC, USA
Akamai Technologies, Cambridge, MA, USA
bmm@cs.duke.edu

Debmalya Panigrahi
Duke University, Durham, NC, USA
debmalya@cs.duke.edu

Rajmohan Rajaraman
Northeastern University, Boston, MA, USA
rraj@ccs.neu.edu

Ravi Sundaram
Northeastern University, Boston, MA, USA
koods@ccs.neu.edu

Abstract

We initiate the algorithmic study of retracting a graph into a cycle in the graph, which seeks a mapping of the graph vertices to the cycle vertices so as to minimize the maximum stretch of any edge, subject to the constraint that the restriction of the mapping to the cycle is the identity map. This problem has its roots in the rich theory of retraction of topological spaces, and has strong ties to well-studied metric embedding problems such as minimum bandwidth and 0-extension. Our first result is an $O(\min\{k, \sqrt{n}\})$-approximation for retracting any graph on n nodes to a cycle with k nodes. We also show a surprising connection to Sperner's Lemma that rules out the possibility of improving this result using certain natural convex relaxations of the problem. Nevertheless, if the problem is restricted to planar graphs, we show that we can overcome these integrality gaps by giving an optimal combinatorial algorithm, which is the technical centerpiece of the paper. Building on our planar graph algorithm, we also obtain a constant-factor approximation algorithm for retraction of points in the Euclidean plane to a uniform cycle.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Graph algorithms, Graph embedding, Planar graphs, Approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.70

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1904.11946`.

Funding All the authors were partially supported by NSF grant CCF 1535972. Additional funding information appears below.
Debmalya Panigrahi: NSF grant CCF 1527084, an NSF CAREER Award CCF 1750140, and the Joint Indo-US Networked Center for Algorithms under Uncertainty.
Ravi Sundaram: NSF grant CNS 1718286.

Acknowledgements We would like to thank Seffi Naor for helpful discussions on the problems considered in this paper.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 70; pp. 70:1–70:15

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

1 Introduction

Originally introduced in 1930 by K. Borsuk in his PhD thesis [5], *retraction* is a fundamental concept in topology describing continuous mappings of a topological space into a subspace that leaves the position of all points in the subspace fixed. Over the years, this has developed into a rich theory with deep connections to fundamental results in topology such as Brouwer's Fixed Point Theorem [22]. Inspired by this success, graph theorists have extensively studied a discrete version of the problem in graphs, where a *retraction* is a mapping from the vertices of a graph to a given subgraph that produces the identity map when restricted to the subgraph (i.e., it leaves the subgraph fixed). For a rich history of retraction in graph theory, we refer the reader to [21]. Define the *stretch* of a retraction to be the maximum distance between the images of the endpoints of any edge, as measured in the subgraph. We use stretch-k retraction to mean a retraction whose stretch is k; in particular, a stretch-1 retraction is a mapping where every edge of the graph is mapped to either an edge of the subgraph, or both its ends are mapped to the same vertex of the subgraph[1].

In this paper, we study the algorithmic problem of finding a *minimum stretch retraction* in a graph. This problem belongs to the rich area of metric embeddings, but somewhat surprisingly, has not received much attention in spite of the deep but non-constructive results in the graph theory literature. The graph retraction problem has a close resemblance to the well-studied 0-extension problem [6, 24, 25] (and its generalizations such as metric labeling [27, 8]), which is also an embedding of a graph G to a metric over a subset of terminals H with the constraint that each vertex in H maps to itself. The two problems differ in their objective: whereas 0-extension seeks to minimize the *average* stretch of edges, graph retraction minimizes the *maximum* stretch. The different objectives lead to significant technical differences. For instance, a well-studied linear program called the earthmover LP has a nearly logarithmic integrality gap for 0-extension. In contrast, we show that a corresponding earthmover LP for graph retraction has integrality gap $\Omega(\sqrt{n})$. A well-studied problem in the metric embedding literature that considers the maximum stretch objective is the *minimum bandwidth* problem, where one seeks an isomorphic embedding of a graph into a line (or cycle) that minimizes maximum stretch. In contrast, in graph retraction, we allow homomorphic maps[2] but additionally require a subset of vertices (called the *anchors*) to be mapped to themselves.

From an applications standpoint, our original motivation for studying minimum-stretch graph retraction comes from a distributed systems scenario where the aim is to map processes comprising a distributed computation to a network of servers where some processes are constrained to be mapped onto specific servers. The objective is to minimize the maximum communication latency between two communicating processes in the embedding. Such anchored embedding problems can be shown to be equivalent to graph retraction for general subgraphs, and arise in several other domains including VLSI layout, multi-processor placement, graph drawing, and visualization [20, 19, 31].

---

[1] In the literature, a stretch-1 retraction is often simply referred to as a retraction or a retract [21]. Also, in many studies, a (stretch-1) retraction requires that the two end-points of an edge in the graph are mapped to two end-points of an edge in the subgraph. These studies differentiate between the case where the subgraph being retracted to is reflexive (has self-loops) or irreflexive (no self-loops). In this sense, our notion of graph retraction corresponds to their notion of retraction to a reflexive subgraph.

[2] A *homomorphic* map is one where an image can have multiple pre-images, while an *isomorphic* map requires that every image has at most one pre-image.

1.1 Problem definition, techniques, and results

We begin with a formal definition of the minimum stretch retraction problem.

▶ **Definition 1.** *Given an unweighted* guest *graph $G = (V, E)$ and a* host *subgraph $H = (A, E')$ of G, a mapping $f : V \to A$ is a retraction of G to H if $f(v) = v$ for all $v \in A$. For a given retraction f of G to H, define the* stretch *of an edge $e = (u, v) \in E(G)$ to be $d_H(f(u), f(v))$, where d_H is the distance metric induced by H, and define the* stretch *of f to be the maximum stretch over all edges of graph G. The goal of the* minimum-stretch graph retraction *problem is to find a retraction of G to H with minimum stretch. We refer to the vertices of A as* anchors*.*

The graph retraction problem is easy if the subgraph H is acyclic (see, e.g., [29]); therefore, the first non-trivial problem is to retract a graph into a cycle. Indeed, this problem is NP-hard even when H is just a 4-cycle [13]. Given this intractability result, a natural goal is to obtain an algorithm for retracting graphs to cycles that *approximately* minimizes the stretch of the retraction. This problem is the focus of our work. While there has been considerable interest in identifying conditions under which retracting to a cycle with stretch 1 is tractable [17, 21, 37], there has been no work (to the best of our knowledge) on deriving approximations to the minimum stretch.[3]

We consider the following lower bound for the problem: if anchors u and v are distance ℓ in H, and there exists a path of p vertices in G between u and v, then every retraction has stretch at least ℓ/p. This lower bound turns out to be tight when H is acyclic, which is the reason retraction to acyclic graphs is an easy problem. However, this lower bound is no longer tight when H is a cycle. For example, consider a grid graph where H is the border of the grid. The lower bound given above says that any retraction has stretch at least $\Omega(1)$. However, using the well-known Sperner's lemma, we show that the optimal retraction has stretch at least $\Omega(\sqrt{n})$.

Using just the simple distance based lower bound, we show that the gap on the grid is in fact the worst possible by giving a $O(\min\{k, \sqrt{n}\})$-approximation for the problem, where k is the number of vertices of H. Our algorithm works by first mapping vertices of the graph into a grid, then projecting vertices outward to the border from the largest *hole* in the grid, which is the largest region containing no vertices.

▶ **Theorem 2.** *There is a deterministic, polynomial-time algorithm that computes a retraction of a graph to a cycle with stretch at most $\min\{k/2, O(\sqrt{n})\}$ times the optimal stretch, where n and k are respectively the number of vertices in the graph and the cycle.*

Our results for retracting a general graph to a cycle appear in Section 2. We also give evidence that the gap induced by Sperner's lemma on a grid graph is fundamental, showing an $\Omega(\min\{k, \sqrt{n}\})$ integrality gap for natural linear and semi-definite programming relaxations of the problem. To overcome this gap, we focus on the special case of planar graphs, of which the grid is an example. Retraction in planar graphs has been considered in the past, most notably in a beautiful paper of Quilliot [30] that uses homotopy techniques to characterize stretch-1 retractions of a planar graph to a cycle. Quillot's proof, however, does not yield an efficient algorithm. In Section 3, we provide an exact algorithm for retraction in planar graphs by developing the gap induced by Sperner's lemma on a grid into a general lower bound on the optimal stretch for planar graphs.

[3] One direct implication of the NP-hardness proof is that approximating the maximum stretch to a multiplicative factor better than 2 is also NP-hard.

▶ **Theorem 3.** *There is a deterministic, polynomial-time algorithm that computes a retraction of a planar graph to a cycle with optimal stretch.*

Unfortunately, our techniques rely heavily on the planarity of the graph, and do not appear to generalize to arbitrary graphs. While we leave the question of obtaining a better approximation for general graphs open, we provide a more sophisticated linear programming formulation that captures the Sperner lower bound on general graphs as a possible route to attack the problem.

We also study natural special cases and generalizations of the problem, all of which are presented in the full version of our paper [18]. First, we consider a geometric setting, where a set of points in the Euclidean plane has to be retracted to a uniform cycle of anchors. By a uniform cycle of anchors we mean a set of anchors which are distributed uniformly on a circle in the plane. We obtain a constant approximation algorithm for this problem, by building on our planar graph algorithm. We next consider retraction of a graph of bounded treewidth to an *arbitrary subgraph*, and obtain a polynomial-time exact algorithm. Finally, we apply the lower bound argument of [24] for 0-extension to show that a general variant of the problem that seeks a retraction of an arbitrary weighted graph G to a metric over a subset of the vertices of G is hard to approximate to within a factor of $\Omega(\log^{1/4-\epsilon} n)$ for any $\epsilon > 0$.

1.2 Related work

List homomorphisms and constraint satisfaction. The graph retraction problem is a special case of the *list homomorphism* problem introduced by Feder and Hell [13], who established conditions under which the problem is NP-complete. Given graphs G, H, and $L(v) \subset V(H)$ for each $v \in V(G)$, a list homomorphism of G to H with respect to L is a homomorphism $f : G \to H$ with $f(v) \in L(v)$ for each $v \in V(G)$.

Several special cases of graph retraction and variants of list homomorphism have been subsequently studied (e.g., [12, 21, 36, 37]). These studies have established and exploited the rich connections between list homomorphism and Constraint Satisfaction Problems (CSPs). Though approximation algorithms for CSPs and related problems such as Label Cover have been extensively studied, the objective pursued there is that of maximizing the number of constraints that are satisfied. For our graph retraction problem, this would correspond to maximizing the number of edges that have stretch below a certain threshold. Our notion of approximation in graph retraction, however, is the least factor by which the stretch constraints need to be relaxed so that all edges are satisfied.

0-extension, minimum bandwidth, and low-distortion embeddings. From an approximation algorithms standpoint, the graph retraction problem is closely related to the 0-extension and minimum bandwidth problems [14, 4, 15, 35, 9, 32]. In the 0-extension problem, one seeks to minimize the average stretch, which can be solved to an $O(\log k / \log \log k)$ approximation using a natural LP relaxation [6, 11]. In contrast, we give polynomial integrality gaps for the graph retraction problem. In the minimum bandwidth problem, the objective is to find an embedding to a line that minimizes maximum stretch, but the constraint is that the map must be isomorphic rather than that the anchor vertices must be fixed. In a seminal result [14], Feige designed the first polylogarithmic-approximation using a novel concept of volume-respecting embeddings. A slightly improved approximation was achieved in [10] by combining Feige's approach with another bandwidth algorithm based on semidefinite-programming [4]. Interestingly, the minimum bandwidth problem is NP-hard even for (guest) trees, while graph retraction to (host) trees is solvable in polynomial time.

Conversely, the bandwidth problem is solvable in time $O(n^b)$ for bandwidth b graphs [16], while graph retraction to a cycle is NP-complete even when the host cycle has only four vertices. Nevertheless, it is conceivable that volume-respecting embeddings, in combination with random projection, could lead to effective approximation algorithms for graph retraction to a cycle in a manner similar to what was achieved for VLSI layout on the plane [35].

Also related are the well-studied variants of linear and circular arrangements, but their objective functions are average stretch, as opposed to maximum stretch. Finally, another related area is that of *low-distortion embeddings* (e.g., [23]), where recent work has considered embedding one specific n-point metric to another n-point metric [26, 28, 2] similar to the graph retraction problem. But low-distortion embeddings typically require *non-contracting* isomorphic maps, which distinguishes them significantly from the graph retraction problem.

A related recent work studies low-distortion *contractions* of graphs [3]. Specifically, the goal is to determine a maximum number of edge contractions of a given graph G such that for every pair of vertices, the distance between corresponding vertices in the contracted graph is at least a given affine function of the distance in G. Several upper bounds and hardness of approximations are presented in [3] for many special cases and problem variants. While graph retraction and contraction problems share the notion of mapping to a subgraph, the problems are considerably different; for instance, in the graph retraction problem the subgraph H is part of the input, and the objective is to minimize the maximum stretch.

2 Retracting an arbitrary graph to a cycle

In this section, we study the problem of retracting an arbitrary graph to a cycle over a subset of vertices of the graph. Let G denote the guest graph over a set V of n vertices, with shortest path distance function d_G. Let H denote the host cycle with shortest path distance function d_H over a subset $A \subseteq V$ of k anchors.

Arguably, the simplest lower bound on the optimal stretch is the *distance-based* bound $\ell(G, H) = \max_{u,v \in A} d_H(u, v)/d_G(u, v)$, since every retraction places a path of length $d_G(u, v)$ in G on a path of length at least $d_H(u, v)$ in H.

We now present our algorithm (Algorithm 1), which achieves a stretch of $\min\{k/2, \ell(G, H)\sqrt{n}\}$. Here, we give a high level overview of the algorithm. The first step of algorithm is to embed the input graph G into a grid of size $k/4 \times k/4$ subject to some constraints. The second step is to find the largest empty sub-grid D such that no point is mapped inside of D and center of D is within a desirable distance from center of grid M. And final step is to project the points in grid M to its boundary with respect to center of sub-grid D.

We now show how to implement the first step of Algorithm 1. Our goal is to embed each vertex $u \in G$ to some point $g(u)$ in a $k/4 \times k/4$ grid such that for every u, v, we have the following inequality, where $d_\infty(a, b)$ denotes the L_∞ distance between a and b. (That is, for two points (x_1, y_1) and (x_2, y_2), $d_\infty((x_1, y_1), (x_2, y_2)) = \max\{|x_1 - x_2|, |y_1 - y_2|\}$.)

$$d_\infty(g(u), g(v)) \leq \ell(G, H) d_G(u, v) \tag{1}$$

Additionally, we require that H is embedded to the boundary of the grid, such that adjacent anchors lie on adjacent grid points.

▶ **Lemma 4.** *For every G, we can find an embedding g satisfying inequality 1.*

Proof. We incrementally construct the embedding g. Initially, we place the anchors on the boundary of the grid so that the boundary is isometric to d_H. (This can be done since H is a cycle.) Since $d_\infty(g(u), g(v)) \leq d_H(u, v)$ and $d_H(u, v) \leq \ell(G, H) d_G(u, v)$, inequality 1 holds for all anchors u and v in H.

Algorithm 1 Algorithm for retracting an arbitrary graph to a cycle.

Input: Graph G, host cycle H
Output: Embedding function f

Embedding in a grid: Determine embedding g from G into a $k/4 \times k/4$ grid M such that H is embedded one-to-one to the boundary of M and for every $u, v \in V$, $d_\infty(g(u), g(v)) \leq \ell(G, H) d_G(u, v)$.

Find largest hole: Find the largest square sub-grid D of M such that (a) its center c is at L_∞ distance at most $k/16$ from the center of M and (b) there is no vertex u in G for which $g(u)$ is in the interior of D.

Projection embedding: For all v in G:

1. $R(v) \leftarrow$ ray originating from the center of D and passing through $g(v)$.
2. $f(v) \leftarrow$ the anchor on the boundary of grid M nearest in the clockwise direction to the intersection of $R(v)$ with the boundary of M.

return f

We next inductively embed the remaining vertices of G. Suppose we need to embed vertex v_i, and vertices $U = v_1, \ldots, v_{i-1}$ have already been embedded. Assume inductively that the embedding of the vertices of U satisfies inequality 1 for the vertices in U.

Let $B_\infty(g(u), r)$ denote the L_∞ ball around $g(u)$ with radius r (note that these balls are axis-aligned squares). Let x be any point in $\bigcap_{u \in U} B_\infty(g(u), \ell(G, H) d_G(u, v_i))$. If we set $g(v_i) = x$, then inequality 1 holds for all points in $U \cup \{v_i\}$. We now show that this intersection is nonempty (it is straightforward to find an element in the intersection). The set of axis aligned squares has Helly number[4] 2; therefore it is enough to show that for every $u, u' \in U$, $B_\infty(g(u), \ell(G, H) d_G(u, v_i))$ and $B_\infty(g(u'), \ell(G, H) d_G(u', v_i))$ intersect. Otherwise,

$$d_\infty(g(u), g(u')) > \ell(G, H)(d_G(u, v_i) + d_G(u', v_i)) \geq \ell(G, H) d_G(u, u').$$

This contradicts our induction hypothesis that the set of vertices in U satisfies inequality 1, and completes the proof of the lemma. ◀

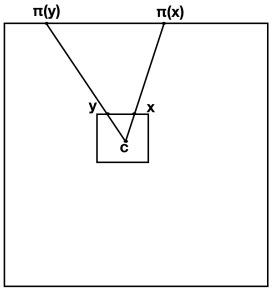
In the following lemma, we analyze the projection embedding step of the algorithm.

▶ **Lemma 5.** *Suppose r is the side length of the largest empty square D inside M. Then for any vertices u and v in G, $d_H(f(u), f(v))$ is at most $1 + (10\sqrt{2}k/r) d_\infty(g(u), g(v))$.*

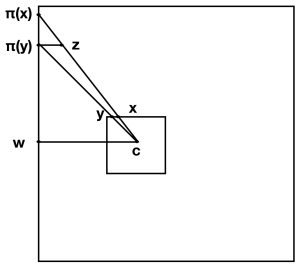
Proof. For any point x, let $\pi(x)$ denote the intersection of the boundary of M and the ray from the center c of D passing through x. Note that for any vertex v in G, $f(v)$ is the anchor in H nearest in clockwise direction to $\pi(g(v))$. We show that for any $x, y \in M$, the distance between $\pi(x)$ and $\pi(y)$ along the boundary of M is at most $(10\sqrt{2}k/r) d_\infty(x, y)$.

We first argue that it is sufficient to establish the preceding claim for points on the boundary of D, at the loss of a factor of $\sqrt{2}$. Let x and y be two arbitrary points in M but not in the interior of D. Let x' (resp., y') denote the intersection of $R(x)$ (resp., $R(y)$) and the boundary of D. From elementary geometry, it follows that $d(x', y') \leq d(x, y)$, where d is the Euclidean distance; since $d_\infty(x, y) \geq d(x, y)/\sqrt{2}$ and $d_\infty(x', y') \leq d(x', y')$, we obtain $d_\infty(x', y') \leq \sqrt{2} d_\infty(x, y)$. Since $\pi(x) = \pi(x')$ and $\pi(y) = \pi(y')$, establishing the above statement for x' and y' implies the same for x and y, up to a factor of $\sqrt{2}$.

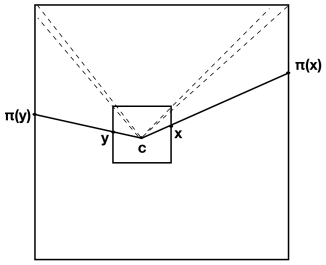
[4] A family of sets has Helly number h if any minimal subfamily with an empty intersection has h or fewer sets in it.



(a) Points x and y are on the same side of square D, and points $\pi(x)$ and $\pi(y)$ are on one side of boundary of M parallel to segment $\overline{xy}$.



(b) Points x and y are on the same side of square D, and points $\pi(x)$ and $\pi(y)$ are on one side of boundary of M orthogonal to segment $\overline{xy}$.



(c) General case where Points x and y (resp. points $\pi(x)$ and $\pi(y)$) are anywhere on the boundary of D (resp. on the boundary of M).

Figure 1 Embedding of points inside the grid M to its boundary using an empty square D. Referred to in the proof of Lemma 5.

Consider points x and y on the boundary of D. We consider three cases. In the first two cases, x and y are on the same side of D. In the first case (Figure 1a), $\pi(x)$ and $\pi(y)$ are on the same side of the boundary of M and segment $\overline{\pi(x)\pi(y)}$ is parallel to segment $\overline{xy}$; then, by similarity of triangle formed by c, x, and y and the one formed by c, $\pi(x)$ and $\pi(y)$, we obtain that the distance between $\pi(x)$ and $\pi(y)$ is at most $3kd_\infty(x,y)/(16r)$. In the second case (Figure 1b), $\pi(x)$ and $\pi(y)$ are on same side of the boundary of M, and segment $\overline{\pi(x)\pi(y)}$ is orthogonal to segment $\overline{xy}$. In this case, w.l.o.g. assume that $\pi(y)$ is closer to center c than $\pi(x)$ with respect to d_∞ distance. Let point z be a point on segment $\overline{c\pi(x)}$ such that segments $\overline{xy}$ and $\overline{\pi(y)z}$ are parallel. From center c extend a line parallel to segment $\overline{xy}$ until it hits the side of M on which $\pi(x)$ and $\pi(y)$ are. Let w be the intersection. Using elementary geometry and similarity argument, we have the following:

$$\frac{|\overline{\pi(x)\pi(y)}|}{|\overline{z\pi(y)}|} = \frac{|\overline{\pi(x)w}|}{|\overline{cw}|} \leq \frac{k/4}{k/16} = 4 \quad \text{and} \quad \frac{\overline{z\pi(y)}}{\overline{xy}} = \frac{\overline{\pi(y)w}}{r} \leq \frac{k}{4r}$$

We thus obtain $\frac{|\overline{\pi(x)\pi(y)}|}{|\overline{xy}|} \leq k/r$. For the third case (Figure 1c), we observe that $d_\infty(x,y)$ is at least half the shortest path between x and y that lies within the boundary of D. This latter shortest path consists of at most five segments, each residing completely on one side of the boundary of D. We apply the argument of the first and second case to each of these segments to obtain that the distance between $\pi(x)$ and $\pi(y)$ is at most $10kd_\infty(x,y)/r$.

To complete the proof, we note that distance between anchor nearest (clockwise) to $\pi(x)$ and anchor nearest (clockwise) to $\pi(y)$ is at most one plus the distance between $\pi(x)$ and $\pi(y)$. Therefore, the $d_H(f(u), f(v))$ is at most $1 + 10\sqrt{2}kd_\infty(g(u), g(v))/r$. ◀

▶ **Theorem 6.** *Algorithm 1 computes a retraction of G to the cycle H with stretch at most the minimum of $k/2$ and $O(\sqrt{n})$ times the optimal stretch.*

Proof. By Lemma 4, the embedding g satisfies inequality 1 for every u and v in G. By a straightforward averaging argument, there exists a square of side length $k/(8\sqrt{n})$ whose center is at L_∞ distance at most $k/16$ from the center of M and which does not contain $g(u)$ for any u in V. By Lemma 5, the projection embedding ensures that for any u and v in V, $d_H(f(u), f(v))$ is at most $1 + O(\sqrt{n})\ell(G,H)d_G(u,v)$. Since the distance in H cannot exceed $k/2$, the claim of the theorem follows. ◀

The Sperner bottleneck. Unfortunately, we cannot improve on the approximation ratio in Theorem 6 using only the distance-based lower bound. Consider the following instance: the guest graph G is the $\sqrt{n} \times \sqrt{n}$ grid, and the host H is the cycle of G formed by the $4\sqrt{n}$ vertices on the outer boundary of G. It is easy to see that the distance-based lower bound has a value of 2 on this instance. On the other hand, using Sperner's Lemma from topology, we show that a stretch of $o(\sqrt{n})$ is ruled out:

▶ **Lemma 7.** *The optimal stretch achievable for an n-vertex grid is at least $2\sqrt{n}/3$.*

Proof. Suppose we triangulate the grid by adding northwest-to-southeast diagonals in each cell of the grid. Consider the following coloring of the boundary H with 3 colors. Divide H into three segments, each consisting of a contiguous sequence of at least $\lfloor 4\sqrt{n}/3 \rfloor$ vertices; all vertices in the first, second, and third segment are colored red, green, and blue, respectively. Let f be any retraction from G to H. Let c_f denote the following coloring for $G \setminus H$: the color of u is the color of $f(u)$. By Sperner's Lemma [34], there exists a tri-chromatic triangle. This implies that there are two vertices within distance at most two in G that are at least $4\sqrt{n}/3$ apart in the retraction f, resulting in a stretch of at least $2\sqrt{n}/3$. ◀

Note that $k = \Theta(\sqrt{n})$ in this instance, so the above lemma also rules out an $o(k)$ approximation using the distance-based lower bound. A natural approach to improving the approximation factor is to use an LP or SDP relaxation for the problem. Indeed, the so-called *earthmover LP* used for the closely related 0-extension problem [24, 7] can be easily adapted to our minimum stretch retraction problem. Similarly, SDP relaxations previously used for minimum bandwidth and related problems [4, 33] can also be adapted to our problem. However, these convex relaxations also have an integrality gap of $\Omega(\sqrt{n})$ for precisely the same reason: they capture the distance-based lower bound but not the one from Sperner's lemma on the grid (see the full version of the paper [18] for a detailed discussion of these LP/SDP relaxations and integrality gaps).

In spite of these gaps, we show that the grid is not a particularly challenging instance of the problem. In fact, in the next section, we give an exact algorithm for retraction in planar graphs, of which the grid is an example. Retraction of planar graphs to cycles has been considered in the past, and non-constructive characterizations of stretch-1 embeddings were known [30]. Our constructive result, while using planarity extensively, suggests that there might be a general technique for addressing the Sperner bottleneck described above. Indeed, we give a candidate LP relaxation (in the full version of the paper [18]) that captures the Sperner bound on the grid. Rounding this LP to obtain a better approximation ratio, or showing an integrality gap for it, is an interesting open question.

3 Retracting a planar graph to a cycle

The main result of this section is the following theorem.

▶ **Theorem 8.** *Let G be a planar graph and H a cycle of G. Then there is a polynomial time algorithm that finds a retraction from G to H with optimal stretch.*

We begin by presenting some useful definitions and elementary claims in Section 3.1. We then present an overview of our algorithm in Section 3.2. Finally, we present the algorithm and its analysis in Section 3.3, leading to the proof of Theorem 8.

3.1 Preliminaries

We begin with a simple lemma that reduces the problem of finding a minimum-stretch retraction to the problem of finding a stretch-1 retraction, in polynomial time. Formally, suppose we have an algorithm $\mathcal{A}$ that, given graphs G and H either finds a stretch-1 retraction from G to H, or proves that no such retraction exists. Then, we can use this algorithm to find the minimum stretch embedding of G into H, using Lemma 9 below, whose straightforward proof is deferred to the full paper [18]. Let G_k be the graph where we replace each edge $e \in G, e \notin H$ with a path of k edges. Clearly, G_k can be computed in polynomial time.

▶ **Lemma 9.** *G can be retracted to H with stretch k if and only if G_k can be retracted in H with stretch-1.*

The following lemma, proved in [18], implies that degree-1 vertices can be eliminated.

▶ **Lemma 10.** *Without loss of generality, we can assume G is 2-vertex connected.*

Lemmas 9 and 10 apply to general graphs. In the rest of this subsection, we focus our attention on planar graphs. We note that all the transformations in Lemmas 9 and 10 preserve planarity of the graph. In 2-connected planar graph, every face of a plane embedding is bordered by a simple cycle. Finally, we can assume that there is a planar embedding of G with H bordering the outer face. If this is not the case, $G \setminus H$ contains at least two connected components, which can each be retracted independently.

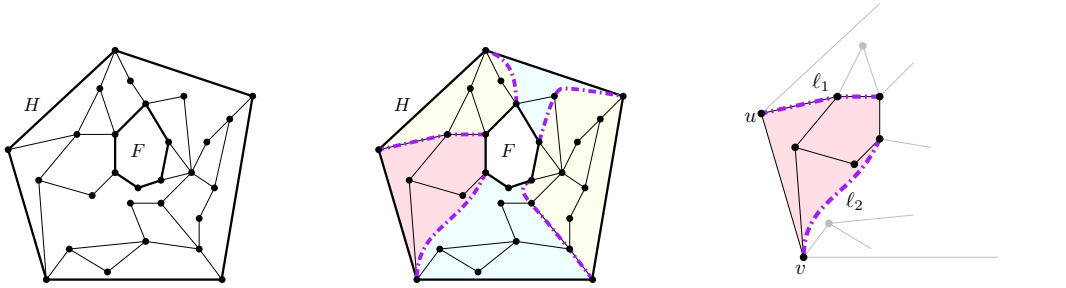
Next, we give some definitions related to planar graphs. We call G *triangulated* if it is maximally planar, i.e., adding any edge results in a graph that is not planar. Equivalently, G is triangulated if every face of a plane embedding (including the outer face) of G has 3 edges. We will make use of the Jordan curve theorem, which says that any closed loop partitions the plane into an inner and outer region (see e.g. [1]). In particular, this implies that any curve crossing from the inner to the outer region intersects the loop. For some cycle C in G and a plane embedding of G, we denote the subset of $\mathbb{R}^2$ surrounded by C as R_C (including the intersection with C itself). We say that $R \subset \mathbb{R}^2$ is *inside* cycle C of G for a plane embedding if $R \subseteq R_C$. If R is inside C, we also say that C *surrounds* R. In a slight abuse of notation, we say C surrounds subgraph G' of G for some fixed plane embedding, if C surrounds the subset of $\mathbb{R}^2$ on which G' is drawn in the plane embedding.

3.2 Overview of our algorithm

Consider some plane embedding of graph G such that H is the subgraph of G bordering G's outer face. We give a polynomial-time algorithm that finds a stretch-1 retraction from G to H or proves that none exists. Using Lemma 9, this immediately yields an algorithm that finds a minimum stretch retraction from G to H.

Fix a planar embedding of G, let H be defined as above, and let F be a bounded face of G. A key component of our algorithm is to find a suitable set of curves connecting F to H. Our aim is to find a set of $k = |V(H)|$ curves in $\mathbb{R}^2$ such that the following hold.

- Each curve begins at a distinct vertex of F and ends at a distinct vertex of H.
- The curves do not intersect each other.
- A curve that intersects an edge of G either contains the edge, or intersects the edge only at its vertices.
- Each curve lies totally in $R_H \setminus F$.



(a) A graph G. The outer cycle H and the face F are shown in bold.

(b) Non-intersecting curves partition the region contained in H but not in F.

(c) Vertices on ℓ_1 are mapped to u, and vertices on ℓ_2 are mapped to v. All other vertices in the region are mapped arbitrarily to u or v.

Figure 2 Using non-intersecting curves to find an embedding from face F to H.

We call curves with these properties *valid* with respect to F. We argue that the curves partition $R_H \setminus F$ (up to their boundaries being duplicated) into a set of regions. Each of these regions is defined by the subset of $\mathbb{R}^2$ surrounded by the closed loop made up of two of the aforementioned curves, a single edge of H, and a path on the boundary of F.

Given a face F and a set of curves valid with respect to F, we can give a stretch-1 retraction from G to H. In essence, the curves partition the graph into regions such that all vertices in a particular region map to one of two end-points of a particular edge of H. See Figure 2 for an illustration.

Of course, it is not obvious that a valid set of curves exists for a given face, and, if it does, how to compute it. We show that if the graph has a stretch-1 retraction, then there is some face F with k valid curves, and that we can efficiently compute them. Our algorithm (Algorithm 2) iterates over all faces in the graph, in each case finding the maximum number of valid curves it can with respect to that face. The number of valid curves we can find is the length of the shortest cycle surrounding F. If the shortest cycle C surrounding F has length ℓ, then it is impossible to find more than ℓ valid curves with respect to F: By the Jordan curve theorem, each curve must intersect C, and by the definition, valid curves do not intersect each other and can intersect C only at its vertices. Our construction of the valid curves shows that this is tight (i.e. we can always find ℓ curves). We show that if a stretch-1 retraction exists, then there is some face for which $\ell = k$. Algorithm 2 gives an outline of the algorithm.

Algorithm 2 Outline for finding a stretch-1 retraction, or proving that none exists.

```
for inner face F in G do
    Compute maximum number of valid curves between F and H p_1, ..., p_ℓ
    if ℓ = k then
        Compute stretch-1 retraction from G to H using p_1, ..., p_k
    end if
end for
If no retraction was computed, report no stretch-1 retraction exists
```

3.3 Algorithm and analysis

This section gives the details of various components of Algorithm 2, and provides a proof of correctness. The following is an outline of the rest of the section:

1. Lemma 12 shows how to compute a stretch-1 retraction using the k valid curves in line 4 of Algorithm 2.
2. Next, Lemma 13 shows that if a stretch-1 retraction exists, there must be some face F in the graph such that the smallest cycle surrounding F has length k.
3. Finally, Lemma 15 gives a construction of largest set of valid curves for a given face F from line 2, and shows that the number of curves computed equals the length of the smallest cycle surrounding F.

We begin by showing in Lemma 11 a somewhat obvious fact: A set of valid curves partition $R_H \setminus F$, and each region of the partition contains a single edge of H. We then show in Lemma 12 that this partition can be used to produce a stretch-1 embedding. See Figure 2 for pictorial presentation of these two lemmas.

▶ **Lemma 11.** *Let $\{p_1, \cdots, p_k\}$ be a set of curves that are valid with respect to F. Let Z denote the set of faces of $H \cup F \cup \bigcup_i p_i$ excluding the outer face and F. Then, each face $f \in Z$ is bordered by exactly 1 edge of H, and every vertex of $G \setminus \bigcup_i p_i$ is in a unique face of Z.*

Proof. Consider the faces of $H \cup F \cup \bigcup_i p_i$. H and F still define faces since the paths p_i fall in $R_H \setminus F$. Let (u, v) be an edge of H, and consider $X = p_i \cup (u, v) \cup p_j \cup p_F(i, j)$ where p_i is the path containing u, p_j is the path containing v, and $p_F(i, j)$ is the path on the boundary of F between the vertices where i and j meet F such that F is not contained in X. If p_i and p_j are both degenerate (i.e., each is empty), then $(u, v) = p_F(i, j)$. Otherwise X is a simple cycle. We claim that X defines a face. In particular, we show that the path $p_F(i, j)$ contains no other vertex of path p_z for all $z \neq i, j$. Suppose it does and let w be that vertex. Let w' be the vertex adjacent to w on p_z. Then $w' \in R_H \setminus F$, and so $w' \in X$. The other end of path p_z, call it vertex y, is in H, but $y \neq u, v$. By the Jordan curve theorem, $p_z \setminus w$ must cross X. Since the graph is planar, $p_z \setminus w$ must contain a vertex of F, H, p_i, or p_j. Any of these outcomes leads to a contradiction. ◀

▶ **Lemma 12.** *Given a non-outer face F and a set $\{p_1, p_2, \ldots, p_k\}$ of curves that are valid with respect to F, we can construct a stretch-1 retraction from G to H in polynomial time.*

Proof. Let Z be as defined in Lemma 11. For each vertex w on p_i, map w to the unique vertex $v \in H \cap p_i$. Otherwise, map w to u or v, where (u, v) is the unique edge of H contained in the same face of Z as w. Fix a face f of Z. Let (u, v) be the unique edge of H contained in f. Any edge (x, y) contained in f also has $x, y \in f$, and so x and y are each mapped to either u or v. Thus, this retraction to H has stretch 1. ◀

As mentioned earlier, we will show that our construction produces ℓ valid curves for face F, where ℓ is the minimum length cycle surrounding F. So we must show that if a stretch-1 retraction exists, there is some F such that every cycle surrounding F has length at least k.

▶ **Lemma 13.** *Fix a plane embedding of G where H defines the outer face of the embedding and suppose there is a stretch-1 retraction G to H. Then there exists a non-outer face F such that every cycle surrounding F has length at least k.*

Proof. We prove a related claim that implies the statement in the lemma. Fix some stretch-1 retraction of G to H. Then there exists a non-outer face F such that for every cycle C in the set of cycles surrounding F, and for each vertex $v \in H$, there is some vertex of C mapped to v. This implies that each of these cycles has length at least k, since the statement says that vertices of C are mapped to k vertices of H.

The claim is very similar to Sperner's lemma, and the proof is similar as well. Let $\phi : V(G) \to V(H)$ denote the retraction. We associate a score with each cycle C of the graph: Order the vertices of the cycle in clockwise order, denoted $v_1, v_2, \ldots, v_j, v_{j+1} = v_1$. Consider the sequence $\phi(v_1), \ldots, \phi(v_j), \phi(v_{j+1})$. Let the score of C be 0 to start. For each pair $\phi(v_i), \phi(v_{i+1})$, we have: either $\phi(v_i) = \phi(v_{i+1})$, or $\phi(v_i)$ and $\phi(v_{i+1})$ are adjacent in H. If $\phi(v_{i+1})$ is clockwise of $\phi(v_i)$ (i.e. if they are in the same order as on C), add 1 to the score of C. If they are in counterclockwise order, subtract 1. If they are the same vertex, the score remains the same. If $\phi(v_1), \ldots, \phi(v_j)$ does not contain every vertex on the outer cycle, the score of C must be 0, since each edge along the path $\phi(v_1), \ldots, \phi(v_{j+1})$ is traversed exactly the same number of times in each direction. On the other hand, a cycle with a non-zero score must have a score that is divisible by k.

Next, we claim that the score of cycle C is the same as the sum of the scores of the cycles defining the faces contained in C. To see this, consider the total contribution to the scores of these cycles from any fixed edge. If the edge is not in cycle C, it is a member of exactly 2 faces contained in C, and contributes either 0 to both faces, or -1 to one and 1 to the other. Edges in C are each a member of just one face surrounded by C. Therefore, the score of cycle C is the same as the sum of scores of its surrounded faces. Since the score of cycle H is k, there must be some face f that has non-zero score.

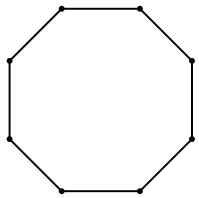
Finally, we show that there is some face with nonzero score such that every cycle surrounding the face also has nonzero score. Suppose this is not the case. Then, every face with a non-zero score is surrounded by a cycle with score 0, which implies that the sum of all scores of faces with non-zero scores is 0. This is a contradiction, since it implies that the sum of scores of all internal faces in the graph is 0. ◀

We complete the section by giving a construction of the largest set of valid curves with respect to some face F, and show that the number of curves equals the length of the shortest cycle surrounding F. Our curves will be disjoint paths in a supergraph $G_\Delta(F)$ of G. It is necessary to relate the maximum number of disjoint paths to the length of the shortest cycle surrounding F. The following lemma, proved in full paper [18], establishes this connection. We believe this lemma should be known, but cannot find it in the relevant literature.

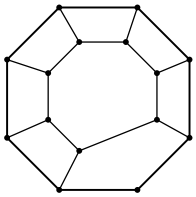
▶ **Lemma 14.** *Let G be a triangulated graph. The graph induced by any minimum s-t vertex cut is the shortest simple cycle separating s and t.*

If G was already triangulated, we could compute a set of vertex disjoint paths from F to H (note that a set of vertex disjoint paths yields a set of valid curves). By Menger's theorem and Lemma 14, we would find ℓ paths, where ℓ is the shortest cycle surrounding F. G may not be triangulated, so instead we could first triangulate G and then compute the paths. However, the number of paths we find in this case is the length of the shortest cycle surrounding F in the triangulation of G, which may be smaller than ℓ. We prevent this from happening by producing a triangulation that adds vertices as well as edges.

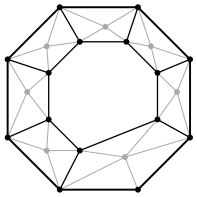
▶ **Lemma 15.** *Fix a planar embedding of G with H as the outer face, and let F be other face. Then we can compute ℓ valid curves in polynomial time, where ℓ is the length of the shortest cycle surrounding F.*



(a) Some face F' with y>3 edges.



(b) Add a new cycle C with $y-1$ edges inside F' along with connecting edges.



(c) Add stars in the newly created faces, except the one formed by C. Distances between vertices of the original face are preserved.

Figure 3 Iteratively triangulate faces.

Proof. We build a triangulated graph $G_\Delta(F)$ from the planar embedding of G. First, add vertices and edges to every face of G, excluding the outer face and F. We do this such that (1) every face except F and the outer face is a triangle, and (2) the distance between any $u, v \in G$ is preserved. From each face with more than 3 edges, we create one new face that has one fewer edge. One step of this iterative construction is shown in Figure 3.

Note that distances are preserved inductively, and we make progress by reducing the size of some face. The graph we produce has 3 edges bordering each face, except for the outer face and F. In all, the number of vertices and edges added to each face of G is polynomial in the number of edges bordering the face.

Finally, we add vertices s and t, and edges from s to each vertex of F and from t to each vertex of C. The resulting graph is triangulated, and we call this graph $G_\Delta(F)$.

At this point, we can find the maximum set of vertex disjoint paths between s and t in $G_\Delta(F)$, by setting vertex capacities to 1 and computing a max flow between s and t. Because we have preserved distances between vertices of G in our construction of $G_\Delta(F)$, the length of the minimum cycle surrounding F must be ℓ. Therefore, the number of disjoint paths we find must also be ℓ. Finally, we claim that this set of disjoint paths from F to H in $G_\Delta(F)$ is a set of valid curves for G. This is because G is a subgraph of $G_\Delta(F)$, and therefore the criteria for valid curves are still met after removing the vertices and edges of $G_\Delta(F) \setminus G$. ◀

We conclude by tying together the pieces of the section to show we proved Theorem 8.

Proof of Theorem 8. Fix a face F. By Lemma 14, we determine the set of ℓ disjoint paths from F to H where the surrounding minimum cycle is of length ℓ. By Lemma 13, there is a stretch-1 retraction only if there exists a face F whose surrounding min-cycle is of length k. So if there is no stretch-1 retraction, we find $< k$ disjoint paths for all faces, and our algorithm returns "no". Otherwise, there exists a face F for which the surrounding min-cycle is of length k, and this gives a set of k valid paths. Then, by Lemma 12, the retraction that we construct has stretch 1. ◀

4 Open problems

Our work leaves several interesting directions for further research. First, we would like to determine improved upper and/or lower bounds on the best approximation factor achievable for retracting a general graph to a cycle. Second, we would like to explore extending our approach for planar graphs (Section 3) and Euclidean metrics (details in the full paper [18]) to more general graphs and high-dimensional metrics. Another open problem is that

of finding approximation algorithms for retracting a general guest graph to an arbitrary host graph over a subset of anchor vertices, for which we present a hardness result in the full paper [18].

References

1 Mark Anthony Armstrong. *Basic Topology*. Springer Science & Business Media, 2013.

2 Mihai Badoiu, Kedar Dhamdhere, Anupam Gupta, Yuri Rabinovich, Harald Räcke, R. Ravi, and Anastasios Sidiropoulos. Approximation algorithms for low-distortion embeddings into low-dimensional spaces. In *SODA*, pages 119–128, 2005.

3 Aaron Bernstein, Karl Däubel, Yann Disser, Torsten Mütze Max Klimm, and Frieder Smolny. Distance-Preserving Graph Contractions. In *ITCS*, 2018.

4 A. Blum, G. Konjevod, R. Ravi, and S. Vempala. Semi-definite relaxations for minimum bandwidth and other vertex-ordering problems. *Theoretical Computer Science*, 235(1):25–42, 2000.

5 Karol Borsuk. *On retractions and related sets*. PhD thesis, University of Warsaw, 1930.

6 G. Calinescu, H. Karloff, and Y. Rabani. Approximation Algorithms for the 0-Extension Problem. *SIAM Journal on Computing*, 34(2):358–372, 2004.

7 Chandra Chekuri, Sanjeev Khanna, Joseph Naor, and Leonid Zosin. A Linear Programming Formulation and Approximation Algorithms for the Metric Labeling Problem. *SIAM Journal on Discrete Math.*, 18:608–625, 2004.

8 J. Chuzhoy and J. S. Naor. The Hardness of Metric Labeling. *SIAM Journal on Computing*, 36(2):498–515, 2006.

9 Chandan K. Dubey, Uriel Feige, and Walter Unger. Hardness results for approximating the bandwidth. *Journal of Computer and System Sciences*, 77(1):62–90, 2011.

10 John Dunagan and Santosh Vempala. On Euclidean Embeddings and Bandwidth Minimization. In *APPROX-RANDOM*, pages 229–240, 2001.

11 Jittat Fakcharoenphol, Chris Harrelson, Satish Rao, and Kunal Talwar. An Improved Approximation Algorithm for the 0-extension Problem. In *SODA*, pages 257–265, 2003.

12 T. Feder, P. Hell, P. Jonsson, A. Krokhin, and G. Nordh. Retractions to Pseudoforests. *SIAM Journal on Discrete Mathematics*, 24(1):101–112, 2010.

13 Tomas Feder and Pavol Hell. List Homomorphisms to Reflexive Graphs. *Journal of Combinatorial Theory*, 72(2):236–250, 1998.

14 Uriel Feige. Approximating the Bandwidth via Volume Respecting Embeddings. *Journal of Computer and System Sciences*, 60(3):510–539, 2000.

15 Anupam Gupta. Improved Bandwidth Approximation for Trees. In *SODA*, pages 788–793, 2000.

16 Eitan M. Gurari and Ivan Hal Sudborough. Improved Dynamic Programming Algorithms for Bandwidth Minimization and the MinCut Linear Arrangement Problem. *Journal of Algorithms*, 5(4):531–546, 1984.

17 Geňa Hahn and Claude Tardif. *Graph homomorphisms: structure and symmetry*, pages 107–166. Springer Netherlands, 1997.

18 Samuel Haney, Mehraneh Liaee, Bruce M. Maggs, Rajmohan Rajaraman, Debmalya Panigrahi, and Ravi Sundaram. Retracting Graphs to cycles. *CoRR*, 2019. `arXiv:1904.11946`.

19 Mark D. Hansen. Approximation algorithms for geometric embeddings in the plane with applications to parallel processing problems. In *FOCS*, pages 604–609, 1989.

20 Stephan Held, Bernhard Korte, Dieter Rautenbach, and Jens Vygen. Combinatorial Optimization in VLSI Design. In *Combinatorial Optimization - Methods and Applications*, pages 33–96. IOS Press, 2011.

21 P. Hell and J. Nesetril. *Graphs and Homomorphisms*. Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, 2004.

22 J.G. Hocking and G.S. Young. *Topology*. Addison-Wesley series in mathematics. Dover Publications, 1961.

23 Matoušek J. Embedding Finite Metric Spaces into Normed Spaces. In *Lectures on Discrete Geometry, Graduate Texts in Mathematics*, volume 212. Springer, 2002.

24 H. Karloff, S. Khot, A. Mehta, and Y. Rabani. On Eathmover Distance, Metric Labeling, and 0-Extension. *SIAM Journal on Computing*, 39(2):371–387, 2009.

25 A. V. Karzanov. Minimum 0-extension of graph metrics. *European Journal of Combinatorics*, 19:71–101, 1998.

26 Claire Kenyon, Yuval Rabani, and Alistair Sinclair. Low Distortion Maps Between Point Sets. *SIAM Journal on Computing*, 39(4):1617–1636, 2009.

27 J. Kleinberg and É Tardos. Approximation algorithms for classification problems with pairwise relationships. *Journal of the ACM*, 49:616–639, 2002.

28 Jiří Matoušek and Anastasios Sidiropoulos. Inapproximability for Metric Embeddings into R^d. In *FOCS*, pages 405–413, 2008.

29 R. Nowakowski and I. Rival. A fixed edge theorem for graphs with loops. *Journal of Graph Theory*, 3:339–350, 1979.

30 Alain Quilliot. A retraction problem in graph theory. *Discrete Mathematics*, 54(1):61–71, 1985.

31 Daniel Rotter and Jens Vygen. d-dimensional arrangement revisited. *Information Processing Letters*, 113(13):498–505, 2013.

32 James B. Saxe. Dynamic-Programming Algorithms for Recognizing Small-Bandwidth Graphs in Polynomial Time. *SIAM on Journal Matrix Analysis Applications*, 1:363–369, 1980.

33 Anthony Man-Cho So and Yinyu Ye. Theory of semidefinite programming for Sensor Network Localization. *Mathematical Programming*, 109(2-3):367–384, 2007.

34 E. Sperner. Neuer beweis für die invarianz der dimensionszahl und des gebietes. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 6(1):265–272, 1928.

35 Santosh Vempala. Random Projection: A New Approach to VLSI Layout. In *FOCS*, pages 389–395, 1998.

36 Narayan Vikas. Compaction, Retraction, and Constraint Satisfaction. *SIAM Journal on Computing*, 33(4):761–782, 2004.

37 Narayan Vikas. A complete and equal computational complexity classification of compaction and retraction to all graphs with at most four vertices and some general results. *Journal of Computer and System Sciences*, 71(4):406–439, 2005.

On Adaptive Algorithms for Maximum Matching

Falko Hegerfeld
Humboldt-Universität zu Berlin, Germany
hegerfeld@informatik.hu-berlin.de

Stefan Kratsch
Humboldt-Universität zu Berlin, Germany
kratsch@informatik.hu-berlin.de

Abstract

In the fundamental Maximum Matching problem the task is to find a maximum cardinality set of pairwise disjoint edges in a given undirected graph. The fastest algorithm for this problem, due to Micali and Vazirani, runs in time $\mathcal{O}(\sqrt{n}m)$ and stands unbeaten since 1980. It is complemented by faster, often linear-time, algorithms for various special graph classes. Moreover, there are fast parameterized algorithms, e.g., time $\mathcal{O}(km \log n)$ relative to tree-width k, which outperform $\mathcal{O}(\sqrt{n}m)$ when the parameter is sufficiently small.

We show that the Micali-Vazirani algorithm, and in fact any algorithm following the phase framework of Hopcroft and Karp, is adaptive to beneficial input structure. We exhibit several graph classes for which such algorithms run in linear time $\mathcal{O}(n + m)$. More strongly, we show that they run in time $\mathcal{O}(\sqrt{k}m)$ for graphs that are k vertex deletions away from any of several such classes, without explicitly computing an optimal or approximate deletion set; before, most such bounds were at least $\Omega(km)$. Thus, any phase-based matching algorithm with linear-time phases obliviously interpolates between linear time for $k = \mathcal{O}(1)$ and the worst case of $\mathcal{O}(\sqrt{n}m)$ when $k = \Theta(n)$. We complement our findings by proving that the phase framework by itself still allows $\Omega(\sqrt{n})$ phases, and hence time $\Omega(\sqrt{n}m)$, even on paths, cographs, and bipartite chain graphs.

2012 ACM Subject Classification Mathematics of computing → Matchings and factors; Theory of computation → Graph algorithms analysis; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Matchings, Adaptive Analysis, Parameterized Complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.71

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1904.11244`.

1 Introduction

The objective in the fundamental Maximum Matching problem is to find a set of disjoint edges of maximum cardinality in a given undirected graph $G = (V, E)$. Maximum Matching has been heavily studied and was the first problem for which a polynomial-time algorithm has explicitly been established [17]. Several algorithms [8, 23, 27, 39] achieve the best known running time of $\mathcal{O}(\sqrt{n}m)$ for graphs with n vertices and m edges, starting with the algorithm of Micali and Vazirani [39] in 1980. Since then, the time of $\mathcal{O}(\sqrt{n}m)$ remains unbeaten.

This state-of-the-art has motivated extensive research into faster algorithms for Maximum Matching on special inputs: Extensive effort went into beating the worst case running time of $\mathcal{O}(\sqrt{n}m)$ for Maximum Matching on special graph classes, resulting in a large number of publications [14, 21, 22, 24, 26, 29, 34, 35], often even obtaining linear-time algorithms [10, 12, 38, 43, 45]. Similarly, there is a great variety of algorithms whose running time depends on n and m but also on some structural parameter of the input graph, like its tree-width, its genus, or its vertex-deletion distance to a certain graph class (summarized in Table 2). As an example, one can solve Maximum Matching in time $\mathcal{O}(k(n + m))$ when

Table 1 The second column shows the running times of dedicated algorithms for Maximum Matching on restricted inputs, which follow as special cases of previous work (see Table 2). Columns three and four show our results for algorithms employing the phase framework with linear-time phases. Only vertex cover number and matching number had known time $\mathcal{O}(\sqrt{k}m)$, others had $\Omega(km)$ prior to our work. The parameters mw and md refer to modular-width and modular-depth.

	Dedicated algorithm	Phase framework algorithms	
Parameter / Graph class	Parameter s	Parameter s	Dist. k to parameter s
Vertex cover number	$\mathcal{O}(\sqrt{s}m)$	$\mathcal{O}(\sqrt{s}m)$	n.a.
Star forest	$\mathcal{O}(m)$	$\mathcal{O}(m)$	$\mathcal{O}(\sqrt{k}m)$
Bounded tree-depth	$\mathcal{O}(m)$	$\mathcal{O}(m)$	$\mathcal{O}(\sqrt{k}m)$
Cluster graph	$\mathcal{O}(m)$	$\mathcal{O}(m)$	$\mathcal{O}(\sqrt{k}m)$
Minimum degree $n-s$	$\mathcal{O}(sn^2 \log n)$	$\mathcal{O}(sm)$	$\mathcal{O}(\sqrt{k}sm)$
Independence number	none	$\mathcal{O}(sm)$	$\mathcal{O}(\sqrt{k}s^2m)$
Neighborhood diversity	$\mathcal{O}((s^2 \log s)n + m)$	$\mathcal{O}(sm)$	$\mathcal{O}(\sqrt{k}sm)$
Parameter mw and md	$\mathcal{O}((\mathrm{mw}^2 \log \mathrm{mw})n + m)$	$\mathcal{O}((c\,\mathrm{mw})^{\mathrm{md}}m)$	$\mathcal{O}(\sqrt{k}(c\,\mathrm{mw})^{\mathrm{md}}m)$

the input graph G is given together with a set S of k vertices such that $G - S$ belongs to a class $\mathcal{C}$ in which Maximum Matching can be solved in linear time (cf. [38]): It suffices to solve the problem on $G - S$ in linear time and to then apply at most k augmentation steps to account for vertices in S; each such step can be implemented in linear time.

A caveat of this great number of different algorithms for special cases is that we may have to first find the relevant structure, e.g., a set S so that $G - S$ belongs to a certain graph class $\mathcal{C}$, and to then decide which algorithm to apply. In some cases, finding the relevant optimal structure is NP-hard and using approximate structure may lead to increase in running time. Moreover, except for time $\mathcal{O}(\sqrt{k}m)$ relative to vertex cover number k or maximum matching size k, which can be seen to follow from the general analysis of Hopcroft and Karp [29], the previously known time bounds improve on $\mathcal{O}(\sqrt{n}m)$ only if $k = \mathcal{O}(\sqrt{n})$, at best.

Our results. We approach the Maximum Matching problem from the perspective of *adaptive analysis (of algorithms)*. Rather than developing further specialized algorithms for special classes of inputs, we prove that a single algorithm actually achieves the best time bounds relative to several graph classes and parameters; in particular, several new or improved time bounds are obtained. Moreover, that algorithm is known since 1980, namely it is the Micali-Vazirani-algorithm [39], and it is oblivious to the actual structure and parameter values. In fact, our analysis does not depend on overly specific aspects of that algorithm and, rather, applies to any algorithm for Maximum Matching that follows the "phase framework" established by Hopcroft and Karp [29] for Bipartite Matching and that implements each phase in linear time, e.g., the algorithms by Blum [8] and Goldberg and Karzanov [27]. In this framework, each phase is dedicated to finding a disjoint and maximal packing of shortest augmenting paths, and it can be shown that $\mathcal{O}(\sqrt{n})$ phases always suffice (cf. [29]).

We show that algorithms following the phase framework adapt to beneficial structure in the form of inputs from special graph classes or inputs that are few vertex deletions away from such a class, without running a recognition algorithm or computing the deletion distance (i.e., they are obliviously adaptive). Concretely, we show that any such algorithm solves Maximum Matching in linear time on several graph classes such as cluster graphs or graphs of bounded neighborhood diversity. Moreover, for many such classes we also show that any such algorithm takes time $\mathcal{O}(\sqrt{k}m)$ on graphs that are k vertex deletions away

from the class, without explicitly computing such a set of deletions (or even knowing the class in question). Furthermore, this running time interpolates between the worst-case time $\mathcal{O}(\sqrt{n}m)$ for $k \in \Theta(n)$ and linear time for $k \in \Theta(1)$, hence remaining competitive even in the absence of beneficial input structure. Except for the matching number and the vertex cover number, time bounds of the form $\mathcal{O}(\sqrt{k}m)$ are new, even for dedicated algorithms. Besides that, we improve upon the algorithm by Yuster [46] for the special case of minimum degree $n - s$ and we improve upon the algorithm by Kratsch and Nelles [33] for the special case of bounded neighborhood diversity. Our positive results are summarized in Table 1.

We complement our findings by exhibiting several graph classes on which the phase framework still allows the worst-case of $\Theta(\sqrt{n})$ phases, and hence $\Theta(\sqrt{n}m)$ time. We prove this for paths, trivially perfect graphs, which are a subclass of cographs, and bipartite chain graphs (and hence their superclasses). This, of course, does not contradict the existence of dedicated linear-time algorithms nor the possibility of tweaking a phase-based algorithm to avoid the obstructions. Nevertheless, these results do rule out the possibility of proving adaptiveness of arbitrary phase-based algorithms, and they showcase obstructions that need to be handled to obtain more general adaptive algorithms for Maximum Matching.

Related work. Our work fits into the recent program of "FPT in P"[1] or efficient parameterized algorithms, initiated independently by Abboud et al. [1] and Giannopoulou et al. [25]. This program seeks to apply the framework of parameterized complexity to tractable problems to obtain provable running times relative to certain parameters that outperform the fastest known algorithms or (conditional) lower bounds obtained in the fine-grained analysis program (see, e.g., [2, 9, 41]). In particular, Mertzios et al. [38] have suggested Maximum Matching as the "drosophila" of FPT in P, i.e., as a central subject of study, similar to the role that Vertex Cover plays in parameterized complexity. Already, there is a large number of publications on parameterized algorithms for Maximum Matching [11, 15, 16, 20, 33, 36], apart from large interest in FPT in P in general [1, 6, 19, 30, 31, 32]. There has also been interest in linear-time preprocessing, which, relative to some parameter k, reduces the problem to solving an instance of size $f(k)$ and leads to time bounds of the form $\mathcal{O}(n+m+g(k))$ [37].

Adaptive analysis of algorithms has been most successful in the context of sorting and searching [3, 4, 13, 18]. We are not aware of prior (oblivious) adaptive analysis of established algorithms for the Maximum Matching problem but two works have designed dedicated adaptive algorithms relative to tree-depth [31] and modular-width [33].

Bast et al. [5] analyzed the Micali-Vazirani algorithm for random graphs, obtaining a running time of $\mathcal{O}(m \log n)$ with high probability.

Organization. Some preliminaries on graphs and matchings are recalled in Section 2. Section 3 is dedicated to recalling the analysis of Hopcroft and Karp [29] and defining phase-based algorithms. In Section 4 we present the positive results, except for results related to neighborhood diversity and modular decomposition which can be found in the full version of the paper. The lower bounds for paths and trivially perfect graphs are given in Section 5; the lower bound for bipartite chain graphs is in the full version of the paper. We conclude in Section 6.

[1] An FPT-algorithm solves a given (usually NP-hard) problem in time $f(k)n^c$ where k is some problem-specific parameter and n is the input size.

Table 2 Known parameterized algorithms for Maximum Matching, k denotes the corresponding parameter value and ω is the matrix multiplication constant.

Parameter	Running Time	Reference
Matching Number	$\mathcal{O}(\sqrt{k}m)$	[8, 27, 29, 39]
Vertex Cover Number	$\mathcal{O}(\sqrt{k}m)$ / $\mathcal{O}(k(n+m)$ / $\mathcal{O}(n+m+k^3)$	[29, 39] / [36] / [36]
Feedback Vertex Number	$\mathcal{O}(k(n+m))$ / $\mathcal{O}(kn+2^{\mathcal{O}(k)})$	[36] / [37]
Feedback Edge Number	$\mathcal{O}(k(n+m))$ / $\mathcal{O}(n+m+k^{1.5})$	[36] / [37]
$\Delta(G)-\delta(G)$	$\mathcal{O}(kn^2\log n)$	[46]
Tree-width	$\mathcal{O}(k^4 n\log n)$ / $\mathcal{O}(km\log n)$	[20] / [31]
Tree-depth	$\mathcal{O}(km)$	[31]
Modular-width	$\mathcal{O}(k^4n+m)$ / $\mathcal{O}((k^2\log k)n+m)$	[11] / [33]
Split-width	$\mathcal{O}((k\log^2 k)(n+m)\log n)$	[15]
P_4-sparseness	$\mathcal{O}(k^4(n+m))$	[11]
Genus	$\mathcal{O}(f(k)n^{\omega/2})$	[47]
H-minor-free	$\mathcal{O}(f(H)n^{3\omega/(\omega+3)})$	[47]
Dist. to Cocomparability	$\mathcal{O}(k(n+m))$	[38]
Distance to Chain Graph	$\mathcal{O}(k(n+m))$ / $\mathcal{O}(n+m+k^3)$	[36] / [37]

2 Preliminaries

We mostly consider simple graphs, unless stated otherwise, and denote an edge between v and w as the concatenation of its endpoints, vw. Let $G=(V,E)$ denote a graph. A path P in G is denoted by listing its vertices in order, i.e., $P=v_1v_2\dots v_\ell$. We use the following notation to refer to subpaths of a path P:

$$P_{[v_i,v_j]}=\begin{cases} v_iv_{i+1}\dots v_j & \text{if } i\le j,\\ v_iv_{i-1}\dots v_j & \text{if } i>j.\end{cases}$$

By *disjoint* paths we will always mean *vertex-disjoint* paths. For a set of vertices $S\subseteq V$, we define $\delta(S)=\{vw\in E : v\in S, w\notin S\}$. For two sets X,Y their *symmetric difference* is denoted by $X\triangle Y=(X\setminus Y)\cup(Y\setminus X)$.

A set $C\subseteq V$ is a *vertex cover* of G if every edge of G has at least one endpoint in C; the *vertex cover number* $\tau(G)$ of G is the minimum cardinality of any vertex cover of G. An *independent set* of G is a set $S\subseteq V$ of pairwise nonadjacent vertices; the *independence number* $\alpha(G)$ of a graph G is the maximum cardinality of any independent set of G. The *maximum degree* and *minimum degree* of G are denoted by $\Delta(G)$ and $\delta(G)$ respectively. For a class $\mathcal{C}$ of graphs we define $\mathrm{d}_{\mathcal{C}}(G)=\min_{S\subseteq V:G-S\in\mathcal{C}}|S|$ to be the *vertex deletion distance* of G to $\mathcal{C}$; a set S such that $G-S\in\mathcal{C}$ is called a *modulator*. E.g., the vertex cover number $\tau(G)$ is the vertex deletion distance of G to edgeless graphs, i.e., independent sets.

A *matching* in a graph is a set of pairwise disjoint edges. Let $G=(V,E)$ be a graph and let $M\subseteq E$ be a matching in G. The matching M is *maximal* if there is no matching M' in G such that $M\subsetneq M'$ and M is *maximum* if there is no matching M' in G such that $|M|<|M'|$; the *matching number* $\nu(G)$ of G is the cardinality of a maximum matching in G.

A vertex $v\in V$ is called *M-matched* if there is an edge in M that contains v, and v is called *M-exposed* otherwise. We do not mention M if the matching is clear from the context. We say that an edge $e\in E$ is *blue* if $e\notin M$ and e is *red* if $e\in M$. An *M-alternating path* is a path in G that alternatingly uses red and blue edges. An *M-augmenting path* is an M-alternating path that starts and ends with an M-exposed vertex; a *shortest* M-augmenting path is an M-augmenting path that uses as few edges as possible.

It is well known that matchings can be enlarged along augmenting paths and that an augmenting path always exists if the matching is not maximum. We say that the matching $M \triangle E(P)$ is obtained by *augmenting* M along P.

▶ **Lemma 2.1.** *If M is a matching in G and P is an M-augmenting path, then $M \triangle E(P)$ is also a matching in G and has size $|M \triangle E(P)| = |M| + 1$.*

▶ **Theorem 2.2** ([29])**.** *Let M and N be matchings in $G = (V, E)$ with $|N| > |M|$. The subgraph $G' = (V, M \triangle N)$ of G contains at least $|N| - |M|$ vertex-disjoint M-augmenting paths.*

▶ **Corollary 2.3** ([7])**.** *A matching M is maximum if and only if there is no M-augmenting path.*

3 Hopcroft-Karp analysis

Many of the fastest algorithms for Maximum Matching make use of a framework introduced by Hopcroft and Karp [29] for the special case of bipartite matching. We give an overview of the framework in this section, mostly following Hopcroft and Karp [29]. The main idea is to search for shortest augmenting paths instead of arbitrary augmenting paths. Exhaustively searching for shortest augmenting paths and augmenting along them leads to Algorithm 1.

Algorithm 1: Generic Matching Algorithm.

Input: Graph G
Output: Maximum matching M

```
M ← ∅;
while M is not maximum do
    Find a shortest M-augmenting path P;
    M ← M△E(P);
return M;
```

Let $P_1, P_2, \ldots, P_\ell$ be the sequence of augmenting paths in the order found during an execution of Algorithm 1. Hopcroft and Karp [29] observed the following properties of the computed shortest augmenting paths.

▶ **Lemma 3.1** ([29])**.** *Let M be a matching, let P be a shortest M-augmenting path, and let P' be a $M \triangle E(P)$-augmenting path, then $|P'| \geq |P| + 2|P \cap P'|$.*

▶ **Corollary 3.2** ([29])**.** *The sequence $|P_1|, \ldots, |P_\ell|$ is non-decreasing.*

▶ **Corollary 3.3** ([29])**.** *If $|P_i| = |P_j|$ for some $i \neq j$, then P_i and P_j are vertex-disjoint.*

Following these observations, we can partition the sequence $P_1, \ldots, P_\ell$ into maximal contiguous subsequences $P_i, P_{i+1}, \ldots, P_j$ such that $|P_i| = |P_{i+1}| = \cdots = |P_j| < |P_{j+1}|$, due to Corollary 3.3 the paths $P_i, P_{i+1}, \ldots, P_j$ must be pairwise vertex-disjoint. Every such subsequence is called a *phase* and corresponds to a maximal set of vertex-disjoint shortest augmenting paths due to Corollary 3.3. With the terminology of phases introduced, it is useful to restate Algorithm 1 as follows.

Algorithm 2: Phase Framework.

Input: Graph G
Output: Maximum matching M

```
M ← ∅;
while M is not maximum do
    Find a maximal set S of vertex-disjoint shortest M-augmenting paths;
    Augment M along all paths in S;
return M;
```

In Algorithm 2 each iteration of the **while**-loop corresponds to a single phase. If an algorithm implements Algorithm 2 we say that it *employs the phase framework*. In the following, we will abstract from the implementation details of algorithms employing the phase framework and only bound the number of phases that are required in the worst case. Hopcroft and Karp [29] presented an upper bound in terms of the matching number $\nu(G)$.

▶ **Theorem 3.4** ([29]). *Every algorithm employing the phase framework requires at most* $2\left\lceil\sqrt{\nu(G)}\right\rceil + 2$ *phases.*

The next bound is a simple corollary of Theorem 3.4 by noticing that $\nu(G) \leq \frac{n}{2}$, but we opt to give an independent proof to serve as an instructive example for the proofs to come.

▶ **Theorem 3.5** (folklore). *Every algorithm employing the phase framework requires at most* $\mathcal{O}(\sqrt{n})$ *phases.*

Proof. Let M denote the matching obtained after performing $\lceil\sqrt{n}\rceil$ phases of Algorithm 2. Every further M-augmenting path has length at least $\lceil\sqrt{n}\rceil$ by Corollary 3.2. This implies that we can pack at most $\lceil\sqrt{n}\rceil$ such augmenting paths into G and hence by Theorem 2.2, at most $\lceil\sqrt{n}\rceil$ augmentations remain. Since we perform at least one augmentation per phase, Algorithm 2 must have terminated after an additional $\lceil\sqrt{n}\rceil$ phases. Thus, Algorithm 2 terminates after at most $2\lceil\sqrt{n}\rceil$ phases. ◀

Several of the fastest Maximum Matching algorithms employ the phase framework [8, 27, 39]. Any one of these algorithms yields the following time bound for a single phase.

▶ **Theorem 3.6.** *There is an algorithm that given a matching M computes a maximal set of vertex-disjoint shortest M-augmenting paths in time $\mathcal{O}(m)$. In particular, each phase of the phase framework can be implemented to run in time $\mathcal{O}(m)$.*

With Theorem 3.5 and Theorem 3.4 we obtain the following time bounds.

▶ **Theorem 3.7.** *There is an algorithm employing the phase framework that solves* Maximum Matching *in time $\mathcal{O}(\sqrt{n}m)$ and $\mathcal{O}(\sqrt{\nu(G)}m)$.*

4 Adaptive parameterized analysis

In this section we will perform an adaptive analysis for algorithms employing the phase framework by analyzing the required number of phases in terms of various graph parameters. Theorem 3.6 yields an improved running time if the considered parameter is small enough.

Many of the considered parameters are NP-hard to compute, e.g., the vertex cover number. This is not an issue, however, as we only require the parameter value for the running time analysis and not for the execution of the algorithm. In this sense, algorithms employing the phase framework are proved to obliviously adapt to the studied parameters.

4.1 Short alternating paths

Our main lemma relies on bounding the length of shortest augmenting paths. In the interest of simplifying later arguments, we will not only bound the length of shortest augmenting paths, but also of alternating paths that are not necessarily augmenting. The strategy for obtaining such upper bounds is to take a long alternating path P and deduce that additional edges must exist in $G[V(P)]$ that enable us to find a shorter alternating path P' in $G[V(P)]$ between the endpoints of P. Hence, the replacement path P' will only visit vertices that are also visited by the original path P. The following definition formalizes this idea.

▶ **Definition 4.1.** Given a matching M in a graph G and an M-alternating path $P = v_1 \dots v_\ell$. We say that an M-alternating path $P' = w_1 \dots w_k$ *replaces* P if the following is true:

- $V(P') \subseteq V(P)$,
- $w_1 = v_1$ and $w_k = v_\ell$,
- P' has the same parity as P with respect to M, i.e.,
  $v_1v_2 \in M \iff w_1w_2 \in M$ and $v_{\ell-1}v_\ell \in M \iff w_{k-1}w_k \in M$.

In particular, if P' replaces P, then P' is at most as long as P.

A technicality that arises from considering general alternating paths, as opposed to augmenting paths, is that an alternating path that starts and ends with a blue edge, i.e. an edge not in the matching, might have endpoints that are not exposed. If we want to shortcut by taking a different edge incident to such an endpoint, then this edge might be red which causes our constructions to fail. To avoid this issue, it suffices to consider the subpath resulting from the removal of the first and last edge.

▶ **Definition 4.2.** A graph G is *ℓ-replaceable* if for every matching M each M-alternating path can be replaced by an M-alternating path of length at most ℓ. The class of ℓ-replaceable graphs is denoted $\mathcal{R}[\ell]$.

We will now show that algorithms employing the phase framework require only few phases for graphs that are close, in the sense of vertex deletion distance, to ℓ-replaceable graphs.

▶ **Lemma 4.3.** *Every algorithm employing the phase framework requires at most $\mathcal{O}(\sqrt{k\ell})$ phases on graphs G with $\mathrm{d}_{\mathcal{R}[\ell]}(G) \leq k$. In particular,* Maximum Matching *can be solved in time $\mathcal{O}(\sqrt{k\ell}m)$ for such graphs.*

Proof. Let $S \subseteq V$ with $|S| \leq k$, such that $G - S \in \mathcal{R}[\ell]$ and let M be the matching obtained after performing $\lceil\sqrt{k\ell}\rceil$ phases of Algorithm 2. We claim that every shortest M-augmenting path uses at least $\lfloor\sqrt{k}\rfloor$ vertices of S; every such path has a length of at least $\lceil\sqrt{k\ell}\rceil$ due to Corollary 3.2. Consider such a path P, since $G - S$ is ℓ-replaceable, we can assume that every time P enters $G - S$ it uses at most $\ell + 1$ vertices of $G - S$ before going back to S. Hence, P must use at least $\lfloor\sqrt{k}\rfloor - 1$ vertices of S to have a length of $\lceil\sqrt{k\ell}\rceil$ or more.

Since S is of size at most k and due to the properties of replacing paths, this implies that we can pack at most $2\lceil\sqrt{k}\rceil$ M-augmenting paths into G as

$$2\left\lceil\sqrt{k}\right\rceil\left(\left\lfloor\sqrt{k}\right\rfloor - 1\right) \geq 2\sqrt{k}\left(\sqrt{k} - 2\right) = 2k - 4\sqrt{k} \geq k \quad \text{for } k \geq 16.$$

By Theorem 2.2 at most $2\lceil\sqrt{k}\rceil$ augmentations remain, which require at most $2\lceil\sqrt{k}\rceil$ phases. In total, we need $\lceil\sqrt{k\ell}\rceil + 2\lceil\sqrt{k}\rceil \in \mathcal{O}(\sqrt{k\ell})$ phases. Theorem 3.6 implies the time bound. ◀

The following running time bound relative to the vertex cover number $\tau(G)$ follows directly from Theorem 3.7 by the use of the well-known inequality $\nu(G) \leq \tau(G)$.

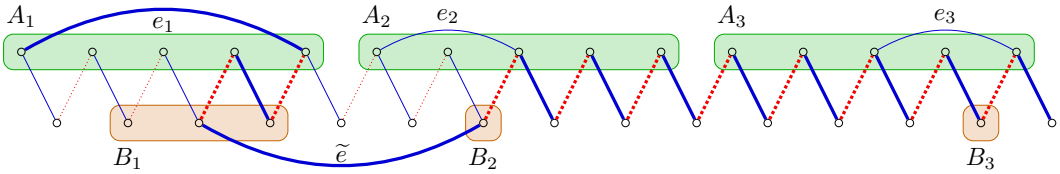


Figure 1 The construction in Theorem 4.6 for $\alpha(G) = 2$, the red dotted edges are matched and the blue edges are unmatched. The thickened edges represent the shorter alternating path P'.

▶ **Theorem 4.4.** *Every algorithm employing the phase framework requires at most* $\mathcal{O}(\sqrt{\tau(G)})$ *phases. In particular,* Maximum Matching *can be solved in time* $\mathcal{O}(\sqrt{\tau(G)}m)$.

Alternatively, observe that if $\mathcal{C}$ is the class of independent sets, then $\mathrm{d}_{\mathcal{C}}(G) = \tau(G)$ and hence Theorem 4.4 is implied by Lemma 4.3 as independent sets are trivially 1-replaceable.

More generally, every graph without paths of length $\ell + 1$ is ℓ-replaceable. It is known that a graph class has bounded path length if and only if it has bounded tree-depth (see, e.g., [40, Chapter 6]). As a further special case consider the class of *star forests*, i.e., graphs where every connected component is a star and therefore must be 2-replaceable. Hence, we obtain the following corollary of Lemma 4.3.

▶ **Corollary 4.5.** *Let* $\mathcal{C}$ *be the class of star forests. Every algorithm employing the phase framework requires at most* $\mathcal{O}(\sqrt{\mathrm{d}_{\mathcal{C}}(G)})$ *phases. In particular,* Maximum Matching *can be solved in time* $\mathcal{O}(\sqrt{\mathrm{d}_{\mathcal{C}}(G)}m)$.

4.2 Independence number

A graph with independence number k contains many edges in the sense that any set of $k+1$ vertices must induce an edge. We will use this property to shorten long alternating paths.

▶ **Theorem 4.6.** *Suppose that* G *is a graph such that* $\alpha(G) \leq k$*, then* G *is* $\mathcal{O}(k^2)$*-replaceable.*

Proof. First, fix a matching M. We show how to replace alternating paths that begin and end with a blue edge, i.e., an edge not in M. By replacing appropriate subpaths of long alternating paths with other parities, the general result will follow.

Suppose that $P = a_1 b_1 a_2 b_2 \dots a_\ell b_\ell$ is an alternating path that is longer than $4(k+1)^2 + 1$ and that starts and ends with a blue edge. We can assume that ℓ is odd. Distinguishing the vertices of P by their parity, we define $A = \{a_i : i \in [\ell]\}$ and $B = \{b_i : i \in [\ell]\}$. Furthermore, we define the sets A_i and A'_i for $i = 1, \dots, k+1$ by

$$A_i = \{a_{(i-1)(2k+2)+j} : j = 1, 2, 3, \dots, 2k+1\} \text{ and } A'_i = \{a_j \in A_i : j \text{ odd}\}.$$

Note that $|A'_i| = k + 1 > k$ for all i, hence the A'_i cannot be independent sets. Thus, there is at least one edge e_i in $G[A'_i]$. We denote the endpoints of e_i by

$$e_i = a_{p_i} a_{q_i}, \text{ where } p_i \leq q_i - 2, \text{ for all } i = 1, \dots, k+1. \tag{1}$$

We now consider the vertices of B that lie between the endpoints of e_i on P; we omit b_{p_i} to ensure that the constructed path is shorter than P. Concretely, let $B_i = \{b_{p_i+1}, b_{p_i+2}, \dots, b_{q_i-1}\}$ for all $i = 1, \dots, k+1$. Equation (1) implies that $B_i \neq \emptyset$. Now, we arbitrarily choose a vertex b'_i from each B_i and define $B' = \{b'_i : i = 1, \dots, k+1\}$. Observe that B' cannot be an independent set as B' contains $k+1$ vertices; thus, there must exist an edge $\tilde{e} = b'_i b'_j$ with $i < j$. We construct the path P' that replaces P by (see Figure 1)

$$P' = P_{[a_1, a_{p_i}]} P_{[a_{q_i}, b'_i]} P_{[b'_j, b_\ell]},$$

using edges $a_{p_i}a_{q_i}$ and $b'_ib'_j$; note that $P_{[a_{q_i},b'_i]}$ is a subpath of P in reverse order. Note that both edges are blue because a_{q_i} and b'_i are already incident with red edges on P. It can be easily checked that P' is an alternating path and, in particular, that it is a valid replacement for P. We will now show that P' is strictly shorter than P. It suffices to compare the length of $P_1 = P_{[a_{p_i},b'_j]}$ and $P_2 = P'_{[a_{p_i},b'_j]} = a_{p_i}P_{[a_{q_i},b'_i]}b'_j$ as P and P' agree on the remaining parts. Let s and t be the indices such that $b'_i = b_s$ and $b'_j = b_t$. The length of P_1 is $2(t-p_i)+1$. For the length of P_2 we obtain

$$\begin{aligned}|P_2| &= 1 + (2(s-q_i)+1) + 1 = 2(s-q_i)+3 < 2(s-(p_i+1))+3 = 2(s-p_i)+1\\ &< 2(t-p_i)+1 = |P_1|,\end{aligned}$$

where the first inequality follows from Equation (1). ◀

By combining this result with Lemma 4.3 we obtain the following corollary.

▶ **Corollary 4.7.** *Every algorithm employing the phase framework requires at most $\mathcal{O}(\alpha(G)^2)$ phases. In particular,* Maximum Matching *can be solved in time $\mathcal{O}(\alpha(G)^2 m)$.*

Let $\mathcal{C}_k$ denote the class of graphs with independence number at most k in each connected component. Every algorithm employing the phase framework requires at most $\mathcal{O}(\sqrt{\mathrm{d}_{\mathcal{C}_k}(G)}k^2)$ phases. In particular, Maximum Matching *can be solved in time $\mathcal{O}(\sqrt{\mathrm{d}_{\mathcal{C}_k}(G)}k^2 m)$.*

A better analysis in terms of the independence number can be achieved by not using replaceability, but in exchange we lose the square root dependence on the size of the modulator.

▶ **Lemma 4.8.** *Every maximal matching M covers at least $n - \alpha(G)$ vertices.*

Proof. If $\alpha(G)+1$ vertices were exposed, they would not be an independent set and hence have an edge between them. Thus, M would not be maximal. ◀

▶ **Corollary 4.9.** *Every algorithm employing the phase framework requires at most $\mathcal{O}(\alpha(G))$ phases. In particular,* Maximum Matching *can be solved in time $\mathcal{O}(\alpha(G)m)$.*

Proof. After the first phase, we know that at most $\alpha(G)$ vertices are exposed by Lemma 4.8. Hence, $\alpha(G)/2$ further augmentations suffice. ◀

4.3 s-plexes

An *s-plex*, $s \geq 1$, is an n-vertex graph G with minimum degree $\delta(G) \geq n - s$. They were introduced by Seidman and Foster [42] as a generalization of cliques, which are 1-plexes. Problems related to s-plexes have been studied in parameterized complexity before [28, 44]. Let $\mathcal{P}[s]$ denote the class of graphs that are disjoint unions of s-plexes. Given a vertex v and a set $W \subseteq V \setminus \{v\}$ of size at least s in an s-plex, we know that there is an edge vw for some $w \in W$. Thus, s-plexes allow for better control than a small independence number as we can guarantee the existence of an edge incident to some specific vertex instead of just getting some edge in a large set of vertices.

▶ **Theorem 4.10.** *If G is a k-plex, i.e., if $\delta(G) \geq n - k$, then G is $\mathcal{O}(k)$-replaceable.*

Proof. Let M be a matching in G and suppose that $P = v_1 \dots v_\ell$ is an M-alternating path in G of length $\ell - 1 \geq 2k + 5$. Let i be the smallest integer such that v_i is M-exposed or $v_{i-1}v_i$ is red, i.e., $v_{i-1}v_i \in M$. Let j be the largest integer such that v_j is M-exposed or v_jv_{j+1} is red. We have $i \in \{1, 2, 3\}$ and $j \in \{\ell - 2, \ell - 1, \ell\}$.

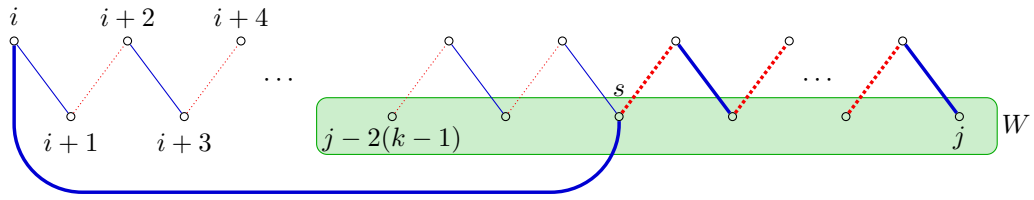


Figure 2 Replacing alternating paths in a k-plex.

Consider the vertices $W = \{v_j, v_{j-2}, v_{j-4}, \dots, v_{j-2(k-1)}\}$ and observe that $v_i \notin W$ as $3 < \ell - 2k$. The set W contains k vertices and since G is a k-plex there must be some $v_s \in W$ such that $v_i v_s \in E$. By choice of i and s the edges $v_i v_{i+1}$ and $v_{s-1} v_s$ must be blue. Similarly, by choice of i, the edge $v_i v_s$ is blue. Hence, as seen in Figure 2, we can replace P by the shorter M-alternating path $P' = P_{[v_1,v_i]} P_{[v_s,v_\ell]}$ with length $|P'| \leq 2 + 2(k-1) + 2 = 2k + 2$. ◀

By Lemma 4.3, we obtain the following corollary.

▶ **Corollary 4.11.** *Every algorithm employing the phase framework requires at most $\mathcal{O}(n - \delta(G))$ phases. In particular,* Maximum Matching *can be solved in time $\mathcal{O}((n - \delta(G))m)$.*

Every algorithm employing the phase framework requires at most $\mathcal{O}(\sqrt{\mathrm{d}_{\mathcal{P}[s]}(G)}s)$ phases. Hence, Maximum Matching *can be solved in time $\mathcal{O}(\sqrt{\mathrm{d}_{\mathcal{P}[s]}(G)}sm)$. In particular,* Maximum Matching *can be solved in time $\mathcal{O}(\sqrt{k}m)$ on graphs that have distance at most k to cluster graphs.*

5 Lower bounds on the number of phases

In this section, we show that several restrictive graph classes do not admit results such as those obtained in the previous section. To this end, we show that the assumptions made about algorithms that follow the phase framework still allow a worst case of $\Omega(\sqrt{n})$ phases. In other words, further assumptions about the behavior of such an algorithm are necessary to avoid these lower bounds and to again get adaptive running times.

5.1 Paths and forests

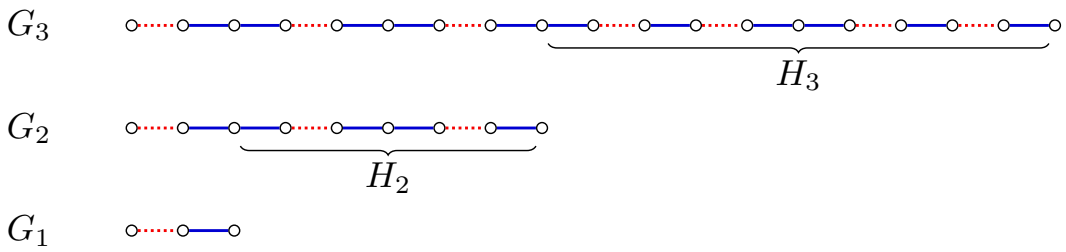


Figure 3 Lower bound construction for paths.

▶ **Lemma 5.1.** *An algorithm employing the phase framework may choose a sequence of augmentations resulting in at least $\Omega(\sqrt{n})$ phases on paths.*

Proof. We will concatenate increasingly long paths where we take every second edge into the matching in such a way that the matching on the newly added path is maximal but not maximum, so that every one of the concatenated paths requires a separate phase. More formally, on $P_{2i} = v_1 v_2 \dots v_{2i}$ we use the matching $\{v_{2k}v_{2k+1} : k \in [i-1]\}$. This matching is one edge away from the maximum and the only augmenting path has length $2i-1$.

Let H_i be obtained by concatenating two copies of P_{2i} through identification of two endpoints (obtaining a path on $4i-1$ vertices). The matching on H_i can be augmented once by an augmenting path of length $2i-1$. There are two augmenting paths on H_i, we always choose to augment along the copy of P_{2i} that is attached to the previously constructed path, i.e., the left copy in Figure 3. Using two copies of P_{2i} in H_i ensures that every H_i requires at least one augmentation. This is not the case if we simply concatenate $P_2, P_4, \dots, P_{2k}$.

We can now define our desired paths. Let $G_1 = H_1 = P_3$ and in this exceptional case we assume that the left edge of P_3 is matched. Furthermore, let G_{i+1} be obtained from G_i by concatenating H_{i+1} at the right. The number of vertices of $G_{\lceil\sqrt{n}\rceil}$ can be bounded by

$$\sum_{i=1}^{\lceil\sqrt{n}\rceil} 2|V(P_{2i})| \leq 4(\lceil\sqrt{n}\rceil)^2 \in \mathcal{O}(n).$$

Now, we argue that our choice of augmentations leads to $\Omega(\sqrt{n})$ phases for $G_{\lceil\sqrt{n}\rceil}$. In the first phase we choose to find the maximal matching that we associated with each H_i. Now, observe that $G_{\lceil\sqrt{n}\rceil}$ contains augmenting paths of lengths $3, 5, \dots, 2\lceil\sqrt{n}\rceil - 1$. In phase i, $i \geq 2$ we augment along the left copy of P_{2i} in H_i. As this does not affect the augmenting paths in the other H_i, we need $\lceil\sqrt{n}\rceil$ phases in total. ◀

Using the parameter values on paths, we obtain the following parameterized lower bounds.

▶ **Corollary 5.2.** *An algorithm employing the phase framework may choose a sequence of augmentations resulting in $\Omega(\sqrt{\alpha(G)})$, $\Omega(\sqrt{\tau(G)})$, $\Omega(\sqrt{\mathrm{d}_{\mathcal{P}[1]}(G)})$ or $\Omega(\sqrt{\mathrm{nd}(G)})$ phases, where $\mathrm{nd}(G)$ is the neighborhood diversity of G.*

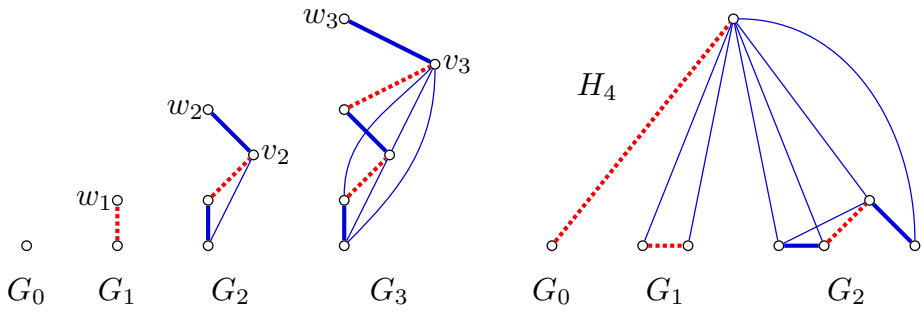
5.2 Cographs

Mertzios et al. [36] devised a Maximum Matching algorithm parameterized by vertex deletion distance to cocomparability graphs and there are several Maximum Matching algorithms parameterized by modular-width [11, 33]. In the interest of showing that these results cannot be replicated or improved by our approach, we give a lower bound result for cographs, i.e., the graphs of modular-width 2 and a subclass of cocomparability graphs.

▶ **Definition 5.3.** A graph is a *cograph* if it can be constructed from the following operations:

- K_1 is a cograph,
- the disjoint union $G \cup H$ of two cographs G and H is a cograph,
- the join $G \times H$ of two cographs G and H is a cograph, where $V(G \times H) = V(G) \cup V(H)$ and $E(G \times H) = E(G) \cup E(H) \cup \{vw : v \in V(G), w \in V(H)\}$.

▶ **Theorem 5.4.** *There is a family of cographs such that an algorithm employing the phase framework may choose a sequence of augmentations resulting in $\Omega(\sqrt{n})$ phases on this family.*



(a) Increasingly long augmenting paths. **(b)** Construction of H_4.

Figure 4 Cographs with increasingly long augmenting paths, the red dotted edges are matched and the blue edges are unmatched. The thickened edges denote the chosen augmenting paths.

Proof. We will construct appropriate cographs using the cograph operations. Let $G_0 = K_1$ and $G_1 = K_1 \times K_1 = P_2$ and for $i \geq 1$ define $G_{i+1} = (G_i \cup K_1) \times K_1$ as auxiliary graphs. Given n, we construct

$$H_n = \left(\bigcup_{i=0}^{\lceil \sqrt{n} \rceil} G_i \right) \times G_0.$$

Observe that G_i has at most $2i+1$ vertices, therefore H_n has $\mathcal{O}(n)$ vertices. We will now describe a sequence of augmentations in H_n that requires $\Omega(\sqrt{n})$ phases.

In each G_i, $i \geq 2$, there is exactly one vertex of degree one, which we call w_i. For G_1, we fix an arbitrary vertex that is called w_1. Furthermore, the vertex that is joined last in the construction of G_i, $i \geq 2$, is referred to as v_i, see Figure 4. First, we describe which maximal matchings to associate with the graphs G_i. In G_0 there is no edge to choose, in G_1 we choose the only possible edge and in G_2 we take the edge v_2w_1. Inductively, for G_{i+1}, $i \geq 2$, we use the maximal matching of G_i and add the edge $v_{i+1}w_i$. With H_n we associate the union of these matchings and add the edge between the two copies of G_0 to the matching. In this way we obtain the matching that is supposed to be found in the first phase.

We now argue that G_i, $i \geq 2$, has exactly one shortest augmenting path of length $2i-1$. This is true for $i = 2$; the other cases follow by induction. Let P_i be the shortest augmenting path in G_i starting at w_i, then $P_{i+1} = w_{i+1}v_{i+1}P_i$ is an augmenting path of length $2i+1$ in G_{i+1}. There are no further augmenting paths as there are only two exposed vertices in G_{i+1} and one of them is w_{i+1} with degree one; starting at w_{i+1}, we must first take the edge $w_{i+1}v_{i+1}$ and then the matched edge $v_{i+1}w_i$, hence we must take the path P_{i+1} by induction.

Let v denote the vertex in G_0 that is joined last in the construction of H_n. The construction of H_n does not create any additional augmenting paths as every maximal alternating path that passes through v must have one matched endpoint, namely the vertex in the other copy of G_0. In phase i we augment the shortest augmenting path of length $2i-1$ in the copy of G_i. By repeating the previous argument, these augmentations cannot introduce any new augmenting paths in the later phases. Hence, we require $\lceil \sqrt{n} \rceil$ phases for this sequence of augmentations, thereby proving the claimed lower bound. ◀

The graphs in the previous proof are also C_4-free, therefore these graphs are not only cographs but also *trivially perfect graphs*.

6 Conclusion

We have conducted an adaptive analysis that applies to all algorithms for Maximum Matching that follow the phase framework of Hopcroft and Karp [29], such as the algorithms due to Micali and Vazirani [39], Blum [8], and Goldberg and Karzanov [27]. The main take-away message of our paper is that these algorithms not only obtain the best known time $\mathcal{O}(\sqrt{n}m)$ for solving Maximum Matching but that they are also (obliviously) adaptive to beneficial structure. That is, they run in linear time on several graph classes and they run in time $\mathcal{O}(\sqrt{k}m)$ for graphs that are k vertex deletions away from any of several classes; before, most bounds were $\Omega(km)$. Arguably, such adaptive algorithms are the best possible result for dealing with unknown beneficial structure because they are never worse than the general bound, in this case taking $\Omega(\sqrt{n}m)$ when $k = \Theta(n)$, and smoothly interpolate to linear time on well-structured instances. Moreover, in the present case, they unify several special cases and remove the need to find exact or approximate beneficial structure.

We complemented our findings by proving that the phase framework alone still allows taking $\Omega(\sqrt{n})$ phases, and, hence, total time $\Omega(\sqrt{n}m)$, even on restrictive classes like paths, trivially perfect graphs, and bipartite chain graphs (and their superclasses), despite the existence of (dedicated) linear-time algorithms. Of course, all of these cases are easy to handle but it raises the question whether there are simple further properties to demand of a phase-based algorithm so that it is provably adaptive to larger classes such as cocomparability or bounded treewidth graphs? In the same vein, it would be interesting whether time $\mathcal{O}(\sqrt{k}m)$ is possible relative to feedback vertex number k, i.e., relative to deletion distance to a forest.

More generally, with the large interest in "FPT in P" (or efficient parameterized algorithms), it seems interesting what other fundamental problems admit adaptive algorithms that interpolate between, say, linear time and the best general bound. Are there other cases where a proven algorithmic paradigm, like the path packing phases of Hopcroft and Karp [29], also obliviously yields the best known running times relative to beneficial input structure?

References

1 Amir Abboud, Virginia Vassilevska Williams, and Joshua Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete Algorithms*, pages 377–391. SIAM, 2016.

2 Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching Triangles and Basing Hardness on an Extremely Popular Conjecture. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 41–50. ACM, 2015. doi:10.1145/2746539.2746594.

3 Jérémy Barbay, Johannes Fischer, and Gonzalo Navarro. LRM-Trees: Compressed indices, adaptive sorting, and compressed permutations. *Theor. Comput. Sci.*, 459:26–41, 2012. doi:10.1016/j.tcs.2012.08.010.

4 Jérémy Barbay and Gonzalo Navarro. On compressing permutations and adaptive sorting. *Theor. Comput. Sci.*, 513:109–123, 2013. doi:10.1016/j.tcs.2013.10.019.

5 Holger Bast, Kurt Mehlhorn, Guido Schafer, and Hisao Tamaki. Matching algorithms are fast in sparse random graphs. *Theory of Computing Systems*, 39(1):3–14, 2006.

6 Matthias Bentert, Till Fluschnik, André Nichterlein, and Rolf Niedermeier. Parameterized aspects of triangle enumeration. In *International Symposium on Fundamentals of Computation Theory*, pages 96–110. Springer, 2017.

7 Claude Berge. Two theorems in graph theory. *Proceedings of the National Academy of Sciences*, 43(9):842–844, 1957.

8 Norbert Blum. A new approach to maximum matching in general graphs. In *International Colloquium on Automata, Languages, and Programming*, pages 586–597. Springer, 1990.

9 Karl Bringmann. Why Walking the Dog Takes Time: Frechet Distance Has No Strongly Subquadratic Algorithms Unless SETH Fails. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 661–670. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.76.

10 Maw-Shang Chang. Algorithms for maximum matching and minimum fill-in on chordal bipartite graphs. In *International Symposium on Algorithms and Computation*, pages 146–155. Springer, 1996.

11 David Coudert, Guillaume Ducoffe, and Alexandru Popa. Fully polynomial FPT algorithms for some classes of bounded clique-width graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2765–2784. Society for Industrial and Applied Mathematics, 2018.

12 Elias Dahlhaus and Marek Karpinski. Matching and multidimensional matching in chordal and strongly chordal graphs. *Discrete Applied Mathematics*, 84(1-3):79–91, 1998.

13 Yann Disser and Stefan Kratsch. Robust and Adaptive Search. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, pages 26:1–26:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.STACS.2017.26.

14 Feodor F. Dragan. On greedy matching ordering and greedy matchable graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 184–198. Springer, 1997.

15 Guillaume Ducoffe and Alexandru Popa. A quasi linear-time b-Matching algorithm on distance-hereditary graphs and bounded split-width graphs. *arXiv preprint*, 2018. arXiv:1804.09393.

16 Guillaume Ducoffe and Alexandru Popa. The use of a pruned modular decomposition for Maximum Matching algorithms on some graph classes. In *29th International Symposium on Algorithms and Computation (ISAAC 2018)*, 29th International Symposium on Algorithms and Computation (ISAAC 2018), Jiaoxi, Yilan County, Taiwan, December 2018. doi:10.4230/LIPIcs.ISAAC.2018.144.

17 Jack Edmonds. Paths, Trees, and Flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965. doi:10.4153/CJM-1965-045-4.

18 Vladimir Estivill-Castro and Derick Wood. A Survey of Adaptive Sorting Algorithms. *ACM Comput. Surv.*, 24(4):441–476, 1992. doi:10.1145/146370.146381.

19 Till Fluschnik, Christian Komusiewicz, George B Mertzios, André Nichterlein, Rolf Niedermeier, and Nimrod Talmon. When can graph hyperbolicity be computed in linear time? In *Workshop on Algorithms and Data Structures*, pages 397–408. Springer, 2017.

20 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, Michał Pilipczuk, and Marcin Wrochna. Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. *ACM Transactions on Algorithms (TALG)*, 14(3):34, 2018.

21 Jean-Luc Fouquet, Vassilis Giakoumakis, and Jean-Marie Vanherpe. Bipartite graphs totally decomposable by canonical decomposition. *International Journal of Foundations of Computer Science*, 10(04):513–533, 1999.

22 Jean-Luc Fouquet, Igor Parfenoff, and Henri Thuillier. An $\mathcal{O}$(n) time algorithm for maximum matching in P_4-tidy graphs. *Information Processing Letters*, 62(6):281–287, 1997.

23 Harold N. Gabow and Robert E. Tarjan. Faster scaling algorithms for general graph matching problems. *Journal of the ACM (JACM)*, 38(4):815–853, 1991.

24 Frédéric Gardi. Efficient algorithms for disjoint matchings among intervals and related problems. In *Discrete Mathematics and Theoretical Computer Science*, pages 168–180. Springer, 2003.

25 Archontia C. Giannopoulou, George B. Mertzios, and Rolf Niedermeier. Polynomial fixed-parameter algorithms: A case study for longest path on interval graphs. *Theoretical Computer Science*, 689:67–95, 2017.

26 Fred Glover. Maximum matching in a convex bipartite graph. *Naval Research Logistics Quarterly*, 14(3):313–316, 1967.

27 Andrew V. Goldberg and Alexander V. Karzanov. Maximum skew-symmetric flows and matchings. *Mathematical Programming*, 100(3):537–568, 2004.

28 Jiong Guo, Christian Komusiewicz, Rolf Niedermeier, and Johannes Uhlmann. A more relaxed model for graph-based data clustering: s-plex editing. In *International Conference on Algorithmic Applications in Management*, pages 226–239. Springer, 2009.

29 John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

30 Thore Husfeldt. Computing Graph Distances Parameterized by Treewidth and Diameter. In Jiong Guo and Danny Hermelin, editors, *11th International Symposium on Parameterized and Exact Computation (IPEC 2016)*, volume 63 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:11, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.IPEC.2016.16.

31 Yoichi Iwata, Tomoaki Ogasawara, and Naoto Ohsaka. On the Power of Tree-Depth for Fully Polynomial FPT Algorithms. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 96. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

32 Leon Kellerhals. Parameterized Algorithms for Network Flows. Master's thesis, TU Berlin, 2018. URL: https://fpt.akt.tu-berlin.de/publications/thesis/MA-leon-kellerhals.pdf.

33 Stefan Kratsch and Florian Nelles. Efficient and Adaptive Parameterized Algorithms on Modular Decompositions. In Yossi Azar, Hannah Bast, and Grzegorz Herman, editors, *26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 55:1–55:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ESA.2018.55.

34 Y. Daniel Liang and Chongkye Rhee. Finding a maximum matching in a circular-arc graph. *Information Processing Letters*, 45(4):185–190, 1993.

35 Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 253–262. IEEE, 2013.

36 George B. Mertzios, André Nichterlein, and Rolf Niedermeier. Fine-grained algorithm design for matching. Technical report, Technical Report, 2016.

37 George B. Mertzios, André Nichterlein, and Rolf Niedermeier. The Power of Linear-Time Data Reduction for Maximum Matching. In Kim G. Larsen, Hans L. Bodlaender, and Jean-Francois Raskin, editors, *42nd International Symposium on Mathematical Foundations of Computer Science (MFCS 2017)*, volume 83 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 46:1–46:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.MFCS.2017.46.

38 George B. Mertzios, André Nichterlein, and Rolf Niedermeier. A Linear-Time Algorithm for Maximum-Cardinality Matching on Cocomparability Graphs. *SIAM Journal on Discrete Mathematics*, 32(4):2820–2835, 2018.

39 Silvio Micali and Vijay V. Vazirani. An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science, 1980*, pages 17–27. IEEE, 1980.

40 J. Nešetřil and P. Ossona de Mendez. *Sparsity (Graphs, Structures, and Algorithms), volume 28 of Algorithms and Combinatorics*. Springer, 2012.

41 Mihai Patrascu and Ryan Williams. On the Possibility of Faster SAT Algorithms. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1065–1075. SIAM, 2010. doi:10.1137/1.9781611973075.86.

42 Stephen B. Seidman and Brian L. Foster. A graph-theoretic generalization of the clique concept. *Journal of Mathematical sociology*, 6(1):139–154, 1978.

43 George Steiner and Julian S Yeomans. A linear time algorithm for maximum matchings in convex, bipartite graphs. *Computers & Mathematics with Applications*, 31(12):91–96, 1996.

44 René Van Bevern, Hannes Moser, and Rolf Niedermeier. Approximation and tidying—a problem kernel for s-plex cluster vertex deletion. *Algorithmica*, 62(3-4):930–950, 2012.

45 Ming-Shing Yu and Cheng-Hsing Yang. An $\mathcal{O}$(n) time algorithm for maximum matching on cographs. *Information Processing Letters*, 47(2):89–93, 1993.

46 Raphael Yuster. Maximum matching in regular and almost regular graphs. *Algorithmica*, 66(1):87–92, 2013.

47 Raphael Yuster and Uri Zwick. Maximum matching in graphs with an excluded minor. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 108–117. Society for Industrial and Applied Mathematics, 2007.

Lower Bounds on Balancing Sets and Depth-2 Threshold Circuits

Pavel Hrubeš
Institute of Mathematics of ASCR, Prague
pahrubes@gmail.com

Sivaramakrishnan Natarajan Ramamoorthy
Paul G. Allen School of Computer Science & Engineering, University of Washington, USA
sivanr@cs.washington.edu

Anup Rao
Paul G. Allen School of Computer Science & Engineering, University of Washington, USA
anuprao@cs.washington.edu

Amir Yehudayoff
Department of Mathematics, Technion-IIT, Haifa, Israel
amir.yehudayoff@gmail.com

Abstract

There are various notions of balancing set families that appear in combinatorics and computer science. For example, a family of proper non-empty subsets $S_1, \ldots, S_k \subset [n]$ is balancing if for every subset $X \subset \{1, 2, \ldots, n\}$ of size $n/2$, there is an $i \in [k]$ so that $|S_i \cap X| = |S_i|/2$. We extend and simplify the framework developed by Hegedűs for proving lower bounds on the size of balancing set families. We prove that if $n = 2p$ for a prime p, then $k \geq p$. For arbitrary values of n, we show that $k \geq n/2 - o(n)$.

We then exploit the connection between balancing families and depth-2 threshold circuits. This connection helps resolve a question raised by Kulikov and Podolskii on the fan-in of depth-2 majority circuits computing the majority function on n bits. We show that any depth-2 threshold circuit that computes the majority on n bits has at least one gate with fan-in at least $n/2 - o(n)$. We also prove a sharp lower bound on the fan-in of depth-2 threshold circuits computing a specific weighted threshold function.

2012 ACM Subject Classification Mathematics of computing → Combinatorics; Theory of computation → Circuit complexity

Keywords and phrases Balancing sets, depth-2 threshold circuits, polynomials, majority, weighted thresholds

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.72

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://eccc.weizmann.ac.il/report/2019/026/`.

Funding This work was done while the authors were visiting the Simons Institute for the Theory of Computing.
Pavel Hrubeš: Supported by ERC grant FEALORA 339691 and the GACR grant 19-27871X.
Sivaramakrishnan Natarajan Ramamoorthy: Supported by the National Science Foundation under agreement CCF- 1420268.
Anup Rao: Supported by the National Science Foundation under agreement CCF- 1420268.
Amir Yehudayoff: Partially supported by ISF grant 1162/15.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 72; pp. 72:1–72:14

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

1 Introduction

1.1 Balancing Families

Balancing set families are families of proper non-empty subsets of a finite universe that satisfy a *discrepancy* type property. They are well studied objects in combinatorics [12, 10, 4, 15, 5, 14], and they have found many applications in computer science [4, 20, 16, 5, 14]. In this work we prove new lower bounds on the size of such families, and then use them to prove lower bounds on depth-2 *majority* and *threshold circuits* that compute the majority and *weighted threshold* functions. We establish new sharp lower bounds on the *fan-in* of the gates in such circuits.

A central contribution of this work is the following lemma that shows a lower bound on the degree of a special class of polynomials.

▶ **Lemma 1.** *Let p be prime, and let $f(x_1, \ldots, x_{2p})$ be a polynomial over $\mathbb{F}_p$, where $\mathbb{F}_p$ is the field with p elements. Let f be such that for every input $x \in \{0,1\}^{2p}$ with exactly p ones, we have $f(x) = 0$, and $f(x)$ is non-zero when $x_1 = x_2 = \ldots = x_{2p} = 0$. Then, the degree of f is at least p.*

Hegedűs [15] used a similar lemma to prove lower bounds for balancing sets (in his lemma there are $4p$ variables, and the focus is on inputs with $3p$ ones). Hegedűs's proof uses Gröbner basis methods and linear algebra. Srinivasan found a simpler proof of Hegedűs's lemma that is based on Fermat's little theorem and linear algebra. Alon [3] gave an alternate proof of Hegedűs's lemma using the Combinatorial Nullstellensatz. The above lemma is inspired by Srinivasan's proof of Hegedűs's lemma [21, 5]. Our simple proof is presented in Section 3.

For a positive integer n, let $[n]$ denote the set $\{1, 2, \ldots, n\}$. Various notions of balancing set families have been considered in the past [12, 10, 4, 15, 5] with various terminologies. We use the following definition in this work.

▶ **Definition 2.** *Let k be a positive integer and n be a positive even integer. We say that proper non-empty subsets $S_1, \ldots, S_k \subset [n]$ are a* balancing set family *if for every $X \subset [n]$ of size $n/2$ there is an $i \in [k]$ such that $|S_i \cap X| = |S_i|/2$.*

Given any even n, let $\mathsf{B}(n)$ denote the minimum k for which a balancing set family of size k exists. Our first result gives tight bounds on $\mathsf{B}(n)$:

▶ **Theorem 3.** *If $n = 2p$ for a prime p, then $\mathsf{B}(n) = n/2 = p$.*

Moreover, if n is divisible by 4, we give an example of a balancing set family establishing that $\mathsf{B}(n) \leq n/2 - 1$. If n is divisible by 2, we show that $\mathsf{B}(n) \leq n/2$ by constructing a balancing set family of size $n/2$, in which each set is of size 2. We also show that this is tight when each set in the family is of size 2 (see the full version for a proof). Previously, for arbitrary values of n, Alon, Kumar and Volk [5] showed that $\mathsf{B}(n) \geq \Omega(n)$. We show

▶ **Theorem 4.** *If n is an even integer, then $\mathsf{B}(n) \geq n/2 - O\left(n^{0.98}\right)$.*

Our lower bounds on $\mathsf{B}(n)$ are the most interesting and they are proved using Lemma 1. See Section 4 and Section 5 for a full exposition of the proofs. We also apply our techniques to other questions about balancing sets in the literature and improve some of the previous bounds. We now briefly discuss two such notions from the literature.

(a) Galvin's question [12, 10, 15] asks for the smallest balancing family, denoted by $\mathsf{G}(n)$, where each set in the family is of size $n/2$, and n is a positive integer that is a multiple of 4.

(b) Jansen [16] and Alon, Kumar, and Volk [5] studied a variant where the size of each set in the family must satisfy $2\tau \leq |S_i| \leq n - 2\tau$ for a positive integer τ, and for every $X \subset [n]$ of size $n/2$, there is a set in the family such that $|S_i|/2 - \tau < |S_i \cap X| < |S_i|/2 + \tau$. Denote by $\mathsf{J}(n, \tau)$ to be the family of smallest size satisfying the above conditions.

We defer the discussion of previous known bounds on the quantities $\mathsf{G}(n)$ and $\mathsf{J}(n, \tau)$ to Section 2. We prove the following lower bounds on $\mathsf{G}(n)$ and $\mathsf{J}(n, \tau)$.

▶ **Theorem 5.** *If n is divisible by 4, then $\mathsf{G}(n) \geq n/2 - O\left(n^{0.53}\right)$.*

▶ **Theorem 6.**

1. *If $n = 2p$ for a prime p then $\mathsf{J}(n, \tau) \geq \frac{n}{4\tau - 2}$.*
2. $\mathsf{J}(n, \tau) \geq \frac{n - O\left(n^{0.98}\right)}{7\tau}$.

We proceed to define the notion of unbalancing set families used in this work.

▶ **Definition 7.** *Let n be a positive even integer, and $k \geq 0, 0 \leq t \leq n/2$ be integers. We say that subsets $S_1, \ldots, S_k \subset [n]$ are an* unbalancing set family *if for every $X \subset [n]$ of size $n/2 - t$, there is an $i \in [k]$ such that $|S_i \cap X| > |S_i|/2$.*

Given any even n, let $\mathsf{U}(n, t)$ denote the minimum k for which an unbalancing set family of size k exists. For unbalancing set families, we determine $\mathsf{U}(n, t)$ exactly:

▶ **Theorem 8.** $\mathsf{U}(n, t) = 2t + 2$.

Again, the lower bound here is more interesting than the upper bound. It is proved by showing a connection between $\mathsf{U}(n, t)$ and the chromatic number of an appropriately defined Kneser graph [18].

1.2 Threshold Circuits

We now discuss our results on depth-2 majority and threshold circuits. The majority function, $\mathsf{MAJ}(x)$ for $x \in \{0,1\}^n$, is defined as

$$\mathsf{MAJ}(x_1, \ldots, x_n) = \begin{cases} 1 & \sum_{i=1}^n x_i \geq n/2, \\ 0 & \text{otherwise.} \end{cases}$$

The unweighted threshold function, $T_t(x)$ for $x \in \{0,1\}^n$, is defined as

$$T_t(x_1, \ldots, x_n) = \begin{cases} 1 & \sum_{i=1}^n x_i \geq t, \\ 0 & \text{otherwise,} \end{cases}$$

for some non-negative integer t. In the rest of the paper, unless stated otherwise, we refer to threshold functions when we mean unweighted threshold functions.

A depth-2 circuit is defined by boolean functions $h, g_1, \ldots, g_k$, for some integer k, and the depth-2 circuit is said to compute a function f on input $x \in \{0,1\}^n$ if

$$f(x) = h(g_1(x), \ldots, g_k(x)).$$

Here $h, g_1, \ldots, g_k$ are called the *gates* of the circuit. h is referred to as the *top* gate, and $g_1, \ldots, g_k$ are referred to as the *bottom* gates of the circuit. Our lower bounds often hold even when h is allowed to be an arbitrary boolean function. The *fan-in* of a gate in the circuit measures the number of variables that need to be read for the gate to carry out its computation. The fan-in of the top gate in the circuit is defined to be k. The fan-in of each

of the gates g_i is r_i if g_i depends on r_i of the input variables. We sometimes refer to the top fan-in when we mean k and the bottom fan-in when we mean the maximum of $r_1, \ldots, r_k$. We say that the fan-in of the circuit is r, if r is the maximum of the top fan-in and bottom fan-in.

When functions $g_1, \ldots, g_k, h$ each compute majority, the circuit is called a majority circuit. Similarly, if all gates compute thresholds, then the circuit is called a threshold circuit. Kulikov and Podolskii [17] asked the following question: What is the minimum fan-in required to compute majority using a depth-2 majority circuit? Balancing set families are closely related to depth-2 majority circuits computing majority. One can prove that there is a depth-2 majority circuit computing majority of n bits with top fan-in at most $2 \cdot \mathsf{B}(n) + 2$, when n is even. Indeed, let $S_1, \ldots, S_k$ be the balancing set family. Define k majority gates, each on variables indexed by S_i, and another k majority gates, each on variables indexed by $[n] \setminus S_i$. The top majority gate, with fan-in $2k + 2$, reads these $2k$ gates along with two 0 inputs. It is easy to see that this circuit correctly computes the majority.

To obtain a lower bound on the fan-in of such circuits, a potential approach is to show that every depth-2 majority or threshold circuit corresponds to a balancing set family. We are able to leverage the ideas that are used to prove Theorem 3 to obtain lower bounds on the fan-in of these circuits. Moreover, our lower bounds are sharp up to a constant factor.

Let $n = 2p$ for a prime p. Note that the threshold function defined by the inequality $\sum_{i=1}^{n} x_i \geq p$ is the majority function on n bits, and yields a circuit with top fan-in 1. We prove a lower bound on the top fan-in of a depth-2 threshold circuit when the bottom gates do not have the threshold p:

▶ **Theorem 9.** *Suppose that $n = 2p$ for a prime p. Then in any depth-2 circuit computing the majority of n bits, if the bottom gates compute unweighted thresholds and read no constants, either the top fan-in is at least $n/2 = p$, or some gate at the bottom computes a threshold T_t with $t = p$.*

In fact, Theorem 9 implies a similar lower bound on the top fan-in when the bottom threshold gates read constants - see Section 6. Observe that in Theorem 9 we do not assume that the top gate h computes a threshold function. The lower bound holds with no restrictions on h.

Theorem 9 also gives tight lower bounds for the fan-in of threshold circuits computing majority. Firstly, any non-constant threshold function T_t reading at most r inputs must have $t \leq r$. Secondly, any bottom gate that computes a threshold function T_t by reading constants is equivalent to computing a threshold function $T_{t'}$ on the same input variables, for some $t' \leq t$, and $T_{t'}$ reads no constants. Here, $t' = t - \alpha$ where α is the number of ones read by T_t. Consequently, we get:

▶ **Corollary 10.** *Suppose that $n = 2p$ for a prime p. Then in any depth-2 circuit computing the majority of n bits, if the bottom gates compute unweighted thresholds, the fan-in of the circuit must be at least $n/2 = p$.*

Since majority is a special case of the threshold function, the above corollary implies the same lower bound on the fan-in of majority circuits that compute the majority. However, by directly invoking Theorem 9, we obtain a slightly stronger lower bound for majority circuits computing the majority:

▶ **Corollary 11.** *Suppose that $n = 2p$ for a prime p. Then in any depth-2 majority circuit computing the majority of n bits, either the bottom fan-in is more than $2p - 2 = n - 2$ or the top-fan in is at least $p = n/2$.*

This is because when the bottom fan-in of the majority circuit is at most $2p - 2$, the threshold of bottom gates are at most $p - 1$ and Theorem 9 applies.

Table 1 Summary of results on balancing and unbalancing families. p is a prime.

Balancing Sets	$\mathsf{B}(n) = n/2$ when $n = 2p$	Theorem 3
	$\mathsf{B}(n) \geq n/2 - o(n)$	Theorem 4
	$\mathsf{G}(n) \geq n/2 - o(n)$	Theorem 5
	$\mathsf{J}(n, \tau) \geq n/(4\tau - 2)$ when $n = 2p$	Theorem 6
	$\mathsf{J}(n, \tau) \geq n(1 - o(1))/7\tau$	Theorem 6
Unbalancing Sets	$\mathsf{U}(n, t) = 2t + 2$	Theorem 8

Theorem 9, Corollary 10 and Corollary 11 discuss the case when $n = 2p$ for a prime p. For arbitrary values of n, we can generalize Theorem 9 to show that either the top fan-in is at least $n/2 - o(n)$ or some gate at the bottom computes a threshold T_t with $t \geq p$, where p is the largest prime such that $p \leq n/2$ (see Section 6 for the proof). Naturally, this lower bound translates to Corollary 10 and Corollary 11. In particular, we get that any depth-2 majority circuit computing the majority of n bits must have that either the bottom fan-in at least $n - o(n)$ or the top fan-in at least $n/2 - o(n)$. This nearly matches Amano's [6] construction of a depth-2 majority circuit with bottom fan-in $n - 2$ and top fan-in $n/2 + 2$.

Another kind of result that we investigate is whether *weighted* threshold functions can be computed using unweighted thresholds of low fan-in. To that end, let $n = (3p - 1)/2$ for an odd prime p, and consider the weighted threshold function

$$T(x) = \begin{cases} 1 & \text{if } \sum_{i \leq p-1} x_i + 2 \sum_{i > p-1} x_i \geq p, \\ 0 & \text{otherwise.} \end{cases}$$

$T(x)$ is a weighted threshold function with weights 1 and 2.

▶ **Theorem 12.** *Any depth-2 circuit computing $T(x)$ where the bottom gates compute unweighted thresholds must have top fan-in at least $(p - 1)/2 = (n - 1)/3$.*

Observe that in Theorem 12 we do not assume an upper bound on the fan-in of the bottom gates. Our bounds are much stronger and significantly simpler than past lower bounds ([17, 9]) on such circuits. Our proofs of Theorem 3 and Theorem 9 are based on proving lower bounds on the degree of specific polynomials, using Lemma 1, that are constructed using the balancing set families and depth-2 threshold circuits, respectively.

Table 1 and Table 2 summarize all our results discussed in the introduction.

Outline

The rest of the paper is organized as follows. We discuss related work in Section 2. We prove Lemma 1 in Section 3. Theorem 3 is proved in Section 4, and the application of our techniques to generalizations of balancing set families are discussed in Section 5. In particular, Section 5 contains the proofs of Theorem 4, 5 and 6. Theorems 9 and 12 are proved in Sections 6 and 7 respectively. Theorem 8 is proved in Section 8.

Notation

$\mathbb{F}_p$ denotes the field with p elements, where p is a prime. For a positive integer n, $\mu(n)$ denotes the largest prime p so that $p \leq n$. For a natural number n, $[n]$ denotes the set $\{1, 2, \ldots, n\}$. For every $x \in \{0,1\}^n$ and $i \in [n]$, x_i denotes the i'th coordinate of x. For $x \in \{0,1\}^n$, when $x_1 = x_2 = \ldots = x_n = 0$, we refer to x as the all-zeros vector or the all-zeros input. The all-ones vector or all-ones input is defined similarly.

Table 2 Summary of results on depth-2 circuits. n is the number of input bits and p is a prime. k is the top fan-in and r is the maximum fan-in of the bottom gates. $\mu(n)$ denotes the largest prime that is no more than n.

Function	Bottom Gates	Result	
Majority	thresholds and reads no constants	$k \geq n/2$ or threshold $= p$ when $n = 2p$	Theorem 9
Majority	thresholds	$\max\{k, r\} \geq n/2$ when $n = 2p$	Corollary 10
Majority	majority	$k \geq n/2$ or $r > n - 2$ when $n = 2p$	Corollary 11
Majority	thresholds	$k \geq n/2 - o(n)$ or threshold $\geq \mu(n/2)$	Theorem 20
Majority	thresholds	$\max\{k, r\} \geq n/2 - o(n)$	Corollary 21
$T(x)$	unbounded fan-in thresholds	$k \geq (n-1)/3$	Theorem 12

Bounds on $\mu(n)$

Generalizations of Theorems 3 and 9 to the case when $n \neq 2p$ for a prime p are obtained by using a known lower bound on $\mu(n)$. Baker, Harman and Pintz [8] showed that the largest gap between consecutive primes is bounded by $O(n^{0.53})$. As a consequence, we can conclude that

▶ **Theorem 13** ([8]). $\mu(n) \geq n - O(n^{0.53})$.

2 Related Work

2.1 Balancing Families

Various notions of balancing set families have been studied. We first describe the question posed by Galvin [12, 10, 15].

▶ **Definition 14.** *Let n be a positive integer that is divisible by 4. A family of proper subsets $S_1, \ldots, S_k \subset [n]$ is* exactly balancing *if each S_i is of size $n/2$ and for every $X \subset [n]$ of size $n/2$ there is an $i \in [k]$ such that $|X \cap S_i| = |S_i|/2$.*

When n is divisible by 4, let $\mathsf{G}(n)$ denote the minimum k for which an exactly balancing set family of size k exists. Clearly, the family of all subsets of $[n]$ of size $n/2$ is exactly balancing, and any family with only one set is not exactly balancing. Therefore finding the minimum number of sets in any exactly balancing set family is interesting.

Galvin [12] observed that $\mathsf{G}(n) \leq n/2$; take $n/2$ consecutive intervals of length $n/2$. Frankl and Rödl [12] proved that $\mathsf{G}(n) \geq \Omega(n)$ if $n/4$ is odd, and later Enomote, Frankl, Ito and Nomura [10] proved that if $n/4$ is odd, then $\mathsf{G}(n) \geq n/2$. Proofs in [12, 10] are based on techniques from linear algebra and extremal set theory. Recently, Hegedűs [15] used algebraic techniques to proved that if $n/4$ is prime, then $\mathsf{G}(n) \geq n/4$. For arbitrary values of n, Alon, Kumar and Volk [5] proved that $\mathsf{G}(n) \geq \Omega(n)$. Theorem 5 improves the bound of Alon, Kumar and Volk.

Several natural variants of Galvin's problem have been studied. One such variant was studied by Jansen [16], and Alon, Kumar and Volk [5]:

▶ **Definition 15.** *Let n be an even integer, and let τ be a positive integer. Let $S_1, \ldots, S_k \subset [n]$ with $2\tau \leq |S_i| \leq n - 2\tau$. We say that $S_1, \ldots, S_k$ is a* τ-balancing set family *if for every $X \subset [n]$ of size $n/2$ there is an $i \in [k]$ such that*

$$|S_i|/2 - \tau < |X \cap S_i| < |S_i|/2 + \tau.$$

When n is even and τ is positive, let $\mathsf{J}(n, \tau)$ denote the minimum k for which such a family of size k exists. This variant allows the family to have sets with different sizes and the intersection sizes to take more than just one value. Alon, Kumar and Volk proved that $\mathsf{J}(n, \tau) \geq \frac{1}{10^5} \cdot (n/\tau)$. This lower bound is sharp up to a constant factor. Theorem 6 improves their bound to $\frac{n - o(n)}{7\tau}$.

Our techniques yield a quantitatively stronger lower bound on balancing set families. The improvement stems from the fact that the ratio of the degree of the polynomial to the number of variables of the polynomial increases from 1/4 to 1/2. Moreover, the application of Lemma 1 eliminates an additional argument using the probabilistic method employed in the work of Alon, Kumar and Volk.

There are many applications of balancing set families. Alon, Bergmann, Coppersmith and Odlyzko [4] studied a different version of balancing sets that has applications to optical data communication. Jansen [16] and Alon, Kumar, and Volk [5] showed applications to proving lower bounds for syntactic multilinear algebraic circuits (also see [20]).

2.2 Threshold Circuits

A depth-d majority circuit can be defined in analogy to depth-2 majority circuits. Let $\mathsf{M}_d(n)$ denote the minimum fan-in of a depth-d majority circuit that computes the majority of n bits. A long line of work has addressed the question of computing the majority function using majority circuits. Ajtai, Komlós and Szemerédi [1] showed that $\mathsf{M}_{c \cdot \log n}(n) = O(1)$, for some constant c. Using probabilistic arguments, Valiant [22] showed the existence of depth $O(\log n)$ majority circuit that computes the majority, where each gate has constant fan-in. Allender and Koucky [2] showed that $\mathsf{M}_c(n) = O(n^{\epsilon(c)})$, where c is a constant and $\epsilon(c)$ is a function of c. Kulikov and Podolskii proved that $\mathsf{M}_3(n) \leq \tilde{O}\left(n^{2/3}\right)$[1]. See [17, 9, 11] and references within for a detailed treatment.

We now discuss previous bounds on $\mathsf{M}_2(n)$. Kulikov and Podolskii [17] used probabilistic arguments to show that $\mathsf{M}_2(n) \geq \tilde{\Omega}\left(n^{7/10}\right)$. They also proved that $\mathsf{M}_2(n) \geq \tilde{\Omega}\left(n^{13/19}\right)$ when the gates are not required to read distinct variables. Amano and Yoshida [7] showed that for every odd $n \geq 7$, $\mathsf{M}_2(n) \leq n - 2$, where they allowed some of the gates to read variables multiple times. Later, Engles, Garg, Makino and Rao [9] used ideas from discrepancy theory to prove that $\mathsf{M}_2(n) \geq \Omega(n^{4/5})$ when the gates do not read constants. Posobin [19] showed that majority can be computed by a depth-2 majority circuit of fan-in at most $2n/3 + 4$ (this was also proved independently by Bauwens [19]). Very recently, Amano [6] gave a construction of a depth-2 majority circuit computing majority with bottom fan-in $n - 2$ and top fan-in $n/2 + 2$.

Kulikov and Podolskii [17] studied and proved lower bounds on other variants of depth-2 majority circuits. In particular, they consider circuits in which each majority gate can read a variable multiple times. Let W be the maximum over the number of times a variable is read. They prove that $\mathsf{M}_2(n) \geq \min\left\{\tilde{\Omega}\left(n^{13/19}\right), \tilde{\Omega}\left(\frac{n^{7/10}}{W^{3/10}}\right)\right\}$. In this case, our techniques yield a lower bound of $\mathsf{M}_2(n) \geq \Omega\left(\frac{n}{W}\right)$. Essentially, their lower bound is stronger when $W \geq n^{6/19}$ and our bound is stronger when $W \leq n^{6/19}$.

[1] In the rest of the paper, $\tilde{O}(a)$ and $\tilde{\Omega}(a)$ mean that $\mathsf{polylog}(a)$ factors are ignored.

The question of computing weighted thresholds using a depth-2 threshold function is connected to the study of exact threshold circuits initiated by Hansen and Podolskii [13]. It may also be useful in studying the expressibility of general functions using threshold or *ReLu* gates; see the work of Williams [23].

We would like to emphasize that the lower bounds in Theorems 9 and 12 are tight and only off by constant factors. In addition, most functions considered in past work on majority and threshold circuit lower bounds do not admit depth-2 majority or threshold circuits with linear fan-in on the gates. In fact, one can prove exponential lower bounds on the size of circuits computing these functions (see [13]).

3 Proof of Lemma 1

Let f be as in the assumption of Lemma 1. Consider the polynomial

$$g(x_1, \ldots, x_{2p}) = (1 - x_1) \cdot \prod_{i=1}^{p-1} \left(i - \sum_{i=1}^{2p} x_i \right),$$

which has degree p. For $x \in \{0,1\}^{2p}$, observe that $g(x) = 0$ if the number of ones in x is not a multiple of p or x is the all-ones input, and $g(x) \neq 0$ if x is the all-zeros input. Therefore, $f \cdot g$ is non-zero on the all-zeros input and 0 elsewhere in $\{0,1\}^{2p}$.

We will now show that the degree of $f \cdot g$ is at least $2p$. Consider the polynomial h that is obtained by multilinearizing $f \cdot g$. In other words, replace every power x_i^k with x_i in $f \cdot g$, for $k \geq 1$, Observe that the degree of h is at most the degree of f. Define $\alpha = h(0, \ldots, 0)$. Recall that there is a one-to-one correspondence between multilinear polynomials over $\mathbb{F}_p$ on $2p$ variables and the set of all functions from $\{0,1\}^{2p} \to \mathbb{F}_p$. Since h is the same as the function that is α on the all-zeros input and 0 elsewhere in $\{0,1\}^{2p}$, we can use this correspondence to conclude that

$$h(x_1, \ldots, x_{2p}) = \alpha \cdot \prod_{i=1}^{2p} (1 - x_i).$$

Therefore the degree of h is $2p$.

Hence the degree of $f \cdot g$ is at least $2p$, implying that the degree of f is at least p.

4 Upper and Lower Bounds on B(n)

In this section, we describe some explicit balancing set families.

▶ **Lemma 16.**
1. *If n is divisible by 4 and $n \neq 4$, then* $\mathsf{B}(n) \leq n/2 - 1$.
2. *If n is divisible by 2 and $n \neq 2$, then* $\mathsf{B}(n) \leq n/2$.

Proof. When 4 divides n, there is a family of $k = \frac{n}{2} - 1$ sets that are balancing: take any k sets, each of size 4, satisfying $S_i \cap S_j = \{1, 2\}$ for all $i \neq j$. This family has the property that for any subset $X \subset [n]$ of size $n/2$, there is an $i \in [k]$ such that $|X \cap S_i| = 2$.

When 2 divides n, there is a family of $k = n/2$ sets that are balancing: take any k sets, each of size 2, satisfying $S_i \cap S_j = \{1\}$ for all $i \neq j$. This family has the property that for any subset $X \subset [n]$ of size $n/2$, there is an $i \in [k]$ such that $|X \cap S_i| = 1$. ◀

As implied by Theorem 3, when $n = 2p$ for a prime p, there is no construction with $k = \frac{n}{2} - 1$ sets; the minimum possible k in this case is $\frac{n}{2}$. We now prove Theorem 3.

Proof of Theorem 3. Lemma 16 implies that $\mathsf{B}(n) \leq p = n/2$. We now proceed to show that $\mathsf{B}(n) \geq p = n/2$. Let $S_1, \ldots, S_k$ be the balancing set family. Without loss of generality each $|S_i|$ is even, and therefore $1 \leq |S_i|/2 \leq p-1$ for all $i \in [k]$. We will now construct a polynomial that is non-zero on the all-zeros input and vanishes on all $x \in \{0,1\}^{2p}$ with p ones. Define the polynomial

$$f(x_1, \ldots, x_{2p}) = \prod_{i=1}^{k} \left(|S_i|/2 - \sum_{j \in S_i} x_j \right),$$

over $\mathbb{F}_p$ that has degree k. Since $1 \leq |S_i|/2 \leq p-1$ for all $i \in [k]$, $f(0) \neq 0$. We will show that $f(x) = 0$, for $x \in \{0,1\}^{2p}$, when x exactly has p ones. This is because the input x to f with exactly p ones corresponds to a set $X \subset [2p]$ of size p. The fact that there is an $i \in [k]$ such that $|S_i \cap X| = |S_i|/2$, implies that $|S_i|/2 - \sum_{j \in S_i} x_j = 0$. By applying Lemma 1, we can conclude that $k \geq p$. ◀

Remark

In Definition 2, since $|S_i \cap X| = |S_i|/2$, it is no loss of generality to assume that each S_i is even sized. The definition can be relaxed by having $|S_i \cap X| = \lceil |S_i|/2 \rceil$. In this relaxed definition, the family $\{1\}, \{2, \ldots, 2p\}$ is balancing and the size of the family is 2. However, if we impose an extra condition that each $|S_i| \geq 2$, then we can prove that the size of any such family is at least p.

5 Balancing Families: Generalizations and Improvements

In this section we prove Theorems 4, 5 and 6. The following lemma is crucial in the proofs these theorems.

▶ **Lemma 17.** *Let n be an even integer. Let $S_1, \ldots, S_k \subset [n]$ and $T_1, \ldots, T_k \subseteq [\mu(n/2) - 1]$. Suppose that there is a set $R \subseteq [n]$ of size $n - 2\mu(n/2)$ such that for every $i \in [k]$ and $t \in T_i$, $|S_i \cap R| < t$, and for every $X \subset [n]$ of size $n/2$ there is an $i \in [k]$ such that $|X \cap S_i| \in T_i$. Then $\sum_{i=1}^{k} |T_i| \geq \mu(n/2)$.*

Proof. Define the polynomial

$$F(x_1, \ldots, x_n) = \prod_{i=1}^{k} \prod_{t \in T_i} \left(t - \sum_{j \in S_i} x_j \right).$$

Let $p = \mu(n/2)$. Define the polynomial $f(x_1, x_2, \ldots, x_{2p})$ over $\mathbb{F}_p$ by setting in F half of the variables indexed by R to 0 and the other half to 1. The degree of f is at most $\sum_{i=1}^{k} |T_i|$. We claim that f takes the value 0 on all inputs with exactly p ones and f is non-zero on the all-zeros input. This is sufficient to prove the theorem as Lemma 1 implies that $\sum_{i=1}^{k} |T_i| \geq p$.

The former part of the claim is true because the input x to f with exactly p ones along with the variables in R that are set to 1 correspond to a set $X \subset [n]$ of size $n/2$. The fact that there is an $i \in [k]$ and $t \in T_i$ with $|S_i \cap X| = t$, implies $t - \sum_{j \in S_i} x_j = 0$.

We now proceed to show that f is non-zero on the all-zeros input. On the all-zeros input for f, we know that all variables indexed by $[n] \setminus R$ are set to 0 and we do not have any control on the assignment to the variables in R. However, since for every $i \in [k]$ and $t \in T_i$, $0 < t < p$ and $|S_i \cap R| < t$, f is non-zero on the all-zeros input. ◀

Implications of Lemma 17

We now discuss the implications of Lemma 17 to the questions about balancing set families discussed in Section 1 and Section 2. The choice of R in Lemma 17 depends on the context. We obtain an asymptotically sharp lower bound for Galvin's problem and an improvement over the lower bound of Alon, Kumar and Volk.

$\mathsf{B}(n)$

We prove Theorem 4 using the following claim (the proof of the claim is presented in the full version of the paper).

▷ Claim 18. Let n be a positive integer and $S_1, \ldots, S_k \subset [n]$ be a balancing set family. If n is large enough and $k < n/2 - 2n^{0.98}$, then there exists a $R \subset [n]$ of size $n - 2\mu(n/2)$ such that for every $i \in [k]$, $|S_i \cap R| < |S_i|/2$.

Proof of Theorem 4. Assume for contradiction that $\mathsf{B}(n) < n/2 - 2n^{0.98}$. Let R be the set given by Claim 18. By invoking Lemma 17 with R and each $T_i = \{|S_i|/2\}$, we get $\mathsf{B}(n) \geq \mu(n/2) \geq n/2 - O\left(n^{0.53}\right)$, where the last inequality follows from Theorem 13. This contradicts the assumption for large values of n. ◀

$\mathsf{J}(n, \tau)$

We prove Theorem 6. We have that each

$$T_i = \{|S_i|/2 - \tau + 1, \ldots, |S_i|/2, \ldots, |S_i|/2 + \tau - 1\}.$$

When $n = 2p$ for a prime p, $R = \emptyset$. Observing that each T_i is of size $2\tau - 1$, Lemma 17 implies Part 1 of Theorem 6.

We now proceed to prove Part 2 of Theorem 6. We need the following claim, and this claim is proved in the full version of the paper.

▷ Claim 19. Let n be a positive integer, τ be a positive integer, and $S_1, \ldots, S_k \subset [n]$ be τ-balancing set family. If n is large enough and $k < n/(7\tau) - n^{0.98}/(7\tau)$, then there exists a $R \subset [n]$ of size $n - 2\mu(n/2)$ such that for every $i \in [k]$, $|S_i \cap R| \leq |S_i|/2 - \tau$.

Proof of Part 2 of Theorem 6. Assume for contradiction that

$$\mathsf{J}(n, \tau) < n/(7\tau) - n^{0.98}/(7\tau).$$

Let R be the set given by Claim 19. By invoking Lemma 17 with R and each

$$T_i = \{|S_i|/2 - \tau + 1, \ldots, |S_i|/2, \ldots, |S_i|/2 + \tau - 1\},$$

we get $\mathsf{J}(n, \tau) \geq \frac{\mu(n/2)}{2\tau - 1} \geq \frac{n - O\left(n^{0.53}\right)}{4\tau - 2}$, where the last inequality follows from Theorem 13. This contradicts the assumption for large values of n. ◀

$\mathsf{G}(n)$

We prove Theorem 5. For Galvin's problem, n is divisible by 4, each S_i is of size $n/2$ and each $T_i = \{n/4\}$. R can be chosen to be any arbitrary set of size $n - 2\mu(n/2)$. For Lemma 17 to apply, we need that for each $i \in [k]$ and $t \in T_i$, $T_i \subseteq [\mu(n/2) - 1]$ and $|S_i \cap R| < t$. This translates in to the condition that $\mu(n/2) > 3n/8$. Lemma 17 in conjunction with Theorem 13 implies Theorem 5.

Specifically Theorem 5 shows that our lower bound is sharp up to an additive $o(n)$ term as $\mathsf{G}(n) \leq n/2$. It is worth noting that $\mathsf{G}(n) < n/2$ for $n \in \{8, 16\}$, so a general $n/2$ lower bound is false (see [5]).

6 Computing Majority using Depth-2 Threshold Circuits

We first prove Theorem 9.

Proof of Theorem 9. Let k be the top fan-in of the circuit, and let $g_1, \ldots, g_k$ be the threshold functions given by the bottom gates of the circuit. We know that g_i is defined by an inequality of the form $L_i(x) \geq t_i$ for a linear function L_i. Assume towards a contradiction that $k < p$ and each $t_i \neq p$.

Define the polynomial

$$f(x) = \prod_{i \in \{j \mid 0 < t_j < 2p\}} (L_i(x) - t_i)$$

over $\mathbb{F}_p$ that has degree at most k. By definition, $f(0)$ is non-zero. We claim that $f(x) = 0$ on every $x \in \{0,1\}^{2p}$ with p ones. Indeed, for such a x we have that $\mathsf{MAJ}(x) = 1$, but for x' that is obtained from x by flipping a coordinate with value 1 to 0, we have that $\mathsf{MAJ}(x') = 0$. Observe that each L_i is a linear function with coefficients in $\{0,1\}$. Since x and x' only differ in one coordinate, we have $L_i(x) - L_i(x') \in \{0,1\}$ for every $i \in [k]$. $\mathsf{MAJ}(x) = 1$ and $\mathsf{MAJ}(x') = 0$ implies that there is an $i \in [k]$ such that $g_i(x) = 1$ and $g_i(x') = 0$. This means that $L_i(x) = t_i$, but $L_i(x') = t_i - 1$. Moreover, this implies that $0 < t_i < 2p$. Hence, for every $x \in \{0,1\}^{2p}$ with p ones, there is an $i \in [k]$ such that $L_i(x) = t_i$ and $0 < t_i < 2p$, which makes $f(x) = 0$. Therefore Lemma 1 implies that the degree of f is at least p, which is a contradiction. ◀

We obtain the following theorem for arbitrary values of n, which is proved using Theorem 9.

▶ **Theorem 20.** *In any depth-2 circuit computing the majority of n bits, if the bottom gates compute unweighted thresholds, either the top fan-in is at least $\mu(n/2)$, or some gate at the bottom computes a threshold T_t with $t \geq \mu(n/2)$.*

Proof. Let k be the top fan-in of the circuit, and let $p = \mu(n/2)$. If there exists a bottom gate with threshold at least p, then we are done. So assume that all bottom gates have threshold less than p. Set half the variables in $x_{2p+1}, \ldots, x_n$ to 0 and the other half to 1. We get a new depth-2 circuit computing the majority of $x_1, \ldots, x_{2p}$. Any bottom threshold gate computing T_t that reads constants is equivalent to a threshold gate computing $T_{t'}$ on the same input variables with $t' \leq t < p$, and $T_{t'}$ reads no constants. Here, $t' = t - \alpha$, where α is the number of ones read by T_t. Replacing each bottom gate that reads constants with its equivalent gate that reads no constants, we obtain a depth-2 circuit in which each bottom gate computes a threshold function with threshold less than p and does not read constants. By applying Theorem 9, we can conclude that $k \geq p$. ◀

Using Theorem 13 we get a corollary to Theorem 20.

▶ **Corollary 21.** *In any depth-2 circuit computing the majority of n bits, if the bottom gates compute unweighted thresholds, then the fan-in is at least $n/2 - O(n^{0.53})$.*

7 Proof of Theorem 12

Let $g_1, \ldots, g_k$ be the threshold functions given by the bottom gates of the circuit. Let

$$L(x) = \sum_{i \leq p-1} x_i + 2 \sum_{i > p-1} x_i.$$

Note that L is a polynomial on $\frac{3p-1}{2}$ variables. For $i \in [k]$, we know that g_i is defined by an inequality of the form $L_i(x) \geq t_i$ for a linear function L_i with coefficients in $\{0, 1\}$.

Consider the polynomial

$$f(x) = \prod_{i \in \{j | 0 < t_j < p\}} (L_i(x) - t_i)$$

over $\mathbb{F}_p$ that has degree at most k. By definition, f is non-zero on the all-zeros input. We will show that $f(x) = 0$ on $x \in \{0,1\}^{\frac{3p-1}{2}}$ such that $L(x) = p$.

Let $x \in \{0,1\}^{\frac{3p-1}{2}}$ be such that $L(x) = p$. Note that for every such x, the number of ones in it is at most $p-1$ and at least 1. For every $x' \in \{0,1\}^{\frac{3p-1}{2}}$ that is obtained by flipping one of the coordinates of x with value 1 to 0, we have $T(x') = 0$. For such x, x', there must be an $i \in [k]$ such that $g_i(x) = 1$ and $g_i(x') = 0$. Moreover, L_i being a linear function with coefficients in $\{0, 1\}$ implies that $L_i(x) - L_i(x') \in \{0, 1\}$. Since $g_i(x) \neq g_i(x')$, we have $L_i(x) = t_i$. In addition, since the number ones in x is at most $p-1$ and at least 1, we get that $0 < t_i < p$. Hence we can conclude that $f(x) = 0$.

We now find a polynomial g that is 0 everywhere in $\{0,1\}^{\frac{3p-1}{2}}$, except on the all-zeros input and x such that $L(x) = p$. Define

$$g(x) = (1 - x_1) \cdot \prod_{i=1}^{p-1} (i - L(x)).$$

The degree of g is p, and $f \cdot g$ is non-zero on the all-zeros input and 0 elsewhere in $\{0,1\}^{\frac{3p-1}{2}}$. We will show that the degree of $f \cdot g$ is at least $(3p-1)/2$. As in the proof of Lemma 1, let h be the multilinearization of $f \cdot g$. Then h is non-zero on the all-zeros input and 0 elsewhere in $\{0,1\}^{\frac{3p-1}{2}}$. Therefore the degree of h is at least $(3p-1)/2$. Since the degree of h is at most the degree of $f \cdot g$, the degree of f is at least $(p-1)/2$.

8 Upper and Lower Bounds on U(n, t)

Theorem 8 is proved in this section. We first recall the definition of a Kneser graph. The Kneser graph $K_{n,\alpha}$ is a graph whose vertices are identified with the subsets of $[n]$ of size α, and there is an edge between two vertices if and only if the corresponding subsets are disjoint. We need the following theorem bounding the chromatic number of Kneser graphs.

▶ **Theorem 22** ([18]). *Consider the Kneser graphs in which the vertex set is given by subsets of $[n]$ of size α. Then the chromatic number of this graph is* $\max\{1, n - 2\alpha + 2\}$.

Proof of Theorem 8. We first prove the upper bound. The following $2t + 2$ sets form an unbalancing family:

$$\{1\}, \{2\}, \ldots, \{2t+1\}, \{2t+2, 2t+3, \ldots, n\}.$$

The above family has the property that for a given $X \subseteq [n]$ of size $n/2 - t$, either $X \subseteq \{2t+2, 2t+3, \ldots, n\}$ or not. In the former case,

$$|X \cap \{2t+2, 2t+3, \ldots, n\}| = n/2 - t > \frac{n - 2t - 1}{2}.$$

In the latter case, there will be an $i \in [2t+1]$ such that $i \in X$. Therefore, $|X \cap \{i\}| = 1 > \frac{1}{2}$.

We now prove the lower bound. Consider the Kneser graph in which the vertex set is given by subsets of $[n]$ of size $n/2 - t$. We claim that the chromatic number of this graph is at most k. The coloring is as follows: For every $X \subseteq [n]$ of size $n/2 - t$, we know that there is an $i \in [k]$ such that $|S_i \cap X| > |S_i|/2$. The vertex associated with X is given the color i. This is a proper coloring because for every $X, Y \subseteq [n]$, each of size $n/2 - t$ that are disjoint, it cannot be the case that $|X \cap S_i| > |S_i|/2$ and $|Y \cap S_i| > |S_i|/2$. Therefore by Theorem 22, we can conclude that $k \geq 2t + 2$. ◀

References

1. M. Ajtai, J. Komlós, and E. Szemerédi. Sorting in c logn parallel steps. *Combinatorica*, 3(1):1–19, March 1983. doi:10.1007/BF02579338.
2. Eric Allender and Michal Koucký. Amplifying lower bounds by means of self-reducibility. *Journal of the ACM*, 57(3):1–36, March 2010. doi:10.1145/1706591.1706594.
3. Noga Alon. Personal Communication, 2019.
4. Noga Alon, Ernest E. Bergmann, Don Coppersmith, and Andrew M. Odlyzko. Balancing sets of vectors. *IEEE Transactions on Information Theory*, 34(1):128–130, 1988.
5. Noga Alon, Mrinal Kumar, and Ben Lee Volk. Unbalancing sets and an almost quadratic lower bound for syntactically multilinear arithmetic circuits. *arXiv*, 2017. arXiv:1708.02037.
6. Kazuyuki Amano. Depth Two Majority Circuits for Majority and List Expanders. In Igor Potapov, Paul Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*, volume 117, pages 81:1–81:13, 2018. URL: http://drops.dagstuhl.de/opus/volltexte/2018/9663.
7. Kazuyuki Amano and Masafumi Yoshida. Depth Two (n-2)-Majority Circuits for n-Majority. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E101.A(9):1543–1545, 2018.
8. Roger C. Baker, Glyn Harman, and János Pintz. The difference between consecutive primes, II. *Proceedings of the London Mathematical Society*, 83(3):532–562, 2001.
9. Christian Engels, Mohit Garg, Kazuhisa Makino, and Anup Rao. On expressing majority as a majority of majorities. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 24, page 174, 2017.
10. Hikoe Enomoto, Peter Frankl, Noboru Ito, and Kazumasa Nomura. Codes with given distances. *Graphs and Combinatorics*, 3(1):25–38, 1987.
11. David Eppstein and Daniel S. Hirschberg. From discrepancy to majority. *Algorithmica*, 80(4):1278–1297, 2018.
12. Peter Frankl and Vojtěch Rödl. Forbidden intersections. *Transactions of the American Mathematical Society*, 300(1):259–286, 1987.
13. Kristoffer Arnsfelt Hansen and Vladimir V. Podolskii. Exact threshold circuits. In *2010 25th Annual IEEE Conference on Computational Complexity*, pages 270–279. IEEE, 2010.
14. Johan Håstad, Guillaume Lagarde, and Joseph Swernofsky. d-Galvin families. *arXiv*, January 2019. arXiv:1901.02652.
15. Gábor Hegedűs. Balancing sets of vectors. *Studia Scientiarum Mathematicarum Hungarica*, 47(3):333–349, 2009.
16. Maurice J. Jansen. Lower bounds for syntactically multilinear algebraic branching programs. In *International Symposium on Mathematical Foundations of Computer Science*, pages 407–418, 2008.

17 Alexander S. Kulikov and Vladimir V. Podolskii. Computing majority by constant depth majority circuits with low fan-in gates. *arXiv*, 2016. arXiv:1610.02686.

18 L. Lovász. Kneser's conjecture, chromatic number, and homotopy. *Journal of Combinatorial Theory, Series A*, 25(3):319–324, 1978.

19 Gleb Posobin. Computing majority with low-fan-in majority queries. *arXiv*, 2017. arXiv:1711.10176.

20 Ran Raz, Amir Shpilka, and Amir Yehudayoff. A lower bound for the size of syntactically multilinear arithmetic circuits. *SIAM Journal on Computing*, 38(4):1624–1647, 2008.

21 Srikanth Srinivasan. Personal Communication, 2018.

22 L.G. Valiant. Short monotone formulae for the majority function. *Journal of Algorithms*, 5(3):363–366, 1984. doi:10.1016/0196-6774(84)90016-6.

23 R. Ryan Williams. Limits on representing Boolean functions by linear combinations of simple functions: thresholds, ReLUs, and low-degree polynomials. *arXiv*, 2018. arXiv:1802.09121.

Scalable and Jointly Differentially Private Packing

Zhiyi Huang
The University of Hong Kong
zhiyi@cs.hku.hk

Xue Zhu
The University of Hong Kong
xuezhu26@hku.hk

Abstract

We introduce an (ϵ, δ)-jointly differentially private algorithm for packing problems. Our algorithm not only achieves the optimal trade-off between the privacy parameter ϵ and the minimum supply requirement (up to logarithmic factors), but is also scalable in the sense that the running time is linear in the number of agents n. Previous algorithms either run in cubic time in n, or require a minimum supply per resource that is $\sqrt{n}$ times larger than the best possible.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Packing and covering problems

Keywords and phrases Joint differential privacy, packing, scalable algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.73

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at [18], `https://arxiv.org/abs/1905.00767`.

Funding *Zhiyi Huang*: This work is supported in by a RGC grant HKU17203717E.

1 Introduction

Suppose a trusted principal has b copies of some privacy-sensitive good, and there are n agents interested in getting a copy of it. Each agent has some value for receiving a copy of the good. The principal would like to choose a subset of up to b agents to receive the good so that sum of their values is maximized.[1] However, one of the agents, Alice, gets paranoid that the others may be able to learn a lot of information about her value for the sensitive good. In particular, here is a hypothetical scenario that Alice worries about. Suppose the principal simply allocate to the k agents with the largest values, with Alice being one of them. Then, all the other $n-1$ agents may exchange information and figure out that only $b-1$ of them get a copy and, hence, Alice must also get one. Further suppose that the b-th highest value among them is, say, \$1,000; they would also learn that Alice's value for the sensitive good is at least \$1,000. *Is there an allocation algorithm that addresses Alice's concerns without losing too much in the objective?*

This problem has been studied in a series of works in the last few years [14, 15, 17]. More broadly, let us consider a general packing problem with m resources and n agents. Each agent demands a bundle consists of a certain amount of each resource, and has a certain value for getting it. The goal is to pick a subset of the agents such that granting them the corresponding bundles approximately maximizes to sum of the values, subject to the supply constraints of the resources, while protecting the privacy of any individual agent. This line of works focus on a specific notation of privacy called joint differential privacy. In a nutshell,

[1] This is also known as social welfare maximization in the literature of mechanism design.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 73; pp. 73:1–73:12


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

it requires that for any individual agent, say, Alice in our example, an adversary shall not be able to learn more than a negligible amount of information about the agent's private information, i.e., her value and demands, from the allocations and prices of the other agents.

How can packing algorithms guarantee joint differential privacy? At a high level, such algorithms leave some amount of supply of each resource unallocated in some meticulous and randomized way, so that even if someone knows the allocations and prices of all other agents as in Alice's hypothetical scenario, he will not be able to learn with certainty whether Alice gets a copy of the item, or her value for it. Further, to ensure that the objective is approximately optimal even with the unallocated supplies, all jointly private algorithms require the supply of each resource to be sufficiently large. Hence, the literature measures how good a jointly private algorithm is by the trade-off between the privacy level, quantified by a parameter $\epsilon > 0$, and the minimum supply requirement, subject to getting an additive αn approximation. To this end, Huang and Zhu [17] show that a supply of $\frac{\sqrt{m}}{\epsilon\alpha}$ per resource, up to logarithmic factors, is both sufficient and necessary.

Another important consideration is the running time of the algorithms. This is particularly relevant for (jointly) differentially private algorithms, since they generally require the size of the dataset, i.e., n, to be sufficiently large to achieve good approximation in the objective. We argue that practical (jointly) differentially private algorithms must be scalable in the sense by Teng [26], i.e., the running time shall be quasi-linear in n or better. However, the aforementioned algorithm by Huang and Zhu [17], which achieves the optimal $\tilde{O}\big(\frac{\sqrt{m}}{\epsilon\alpha}\big)$ supply requirement, is not scalable, as its running time depends cubically in n. Neither are the earlier algorithms by Hsu et al. [14, 15]. Although Huang and Zhu [17] also propose an alternative algorithm that is scalable, it requires a much larger supply of $\tilde{O}\big(\frac{\sqrt{mn}}{\epsilon\alpha}\big)$ per resource. The following question is explicitly left open [17]:

> *Are there jointly differentially private algorithm that are scalable and at the same time only require a minimum supply of $\tilde{O}\big(\frac{\sqrt{m}}{\epsilon\alpha}\big)$ per resource?*

1.1 Our Contributions

We introduce a jointly differentially private packing algorithm that answers the above open question affirmatively. The main theorem of this paper is the following:

▶ **Theorem 1.** *There is an (ϵ, δ)-jointly differentially private algorithm such that:*

1. *it returns with high probability a feasible packing solution that is optimal up to an αn additive factor, provided that the supply per resource is at least $\tilde{O}\big(\frac{\sqrt{m}}{\alpha\epsilon}\big)$;*
2. *it stops in $O(n)$ time, omitting dependence in other parameters, with high probability.*

The algorithm follows the same high-level framework as the previous ones, which we summarize below. It maintains for each resource a price (per unit of the resource), which can be viewed as a dual variable that Lagrangianizes the corresponding resource constraint of the packing problem. We shall imagine that the prices are posted on a public billboard for everyone to see, including the agents and the adversary. Given the current prices, each agent gets the bundle if and only if her value is higher than the total price of the bundle. The agents' decisions induce a total demand on each resource, which can be viewed as a subgradient for the dual prices w.r.t. some dual objective. The algorithm then increases the prices of the overdemanded resources and decreases those of the underdemended ones. This process repeats for a certain number of rounds; the final allocation is obtained by averaging over all the rounds. Since the agents' allocation is coordinated only through the prices, it suffices to ensure that the sequence of prices is privacy-preserving. This is formulated as the billboard lemma by Hsu et al. [14].

What are the main differences between our algorithm and the existing ones? The previous non-scalable algorithms are essentially noisy versions of some existing optimization algorithms, including gradient descent [15] and multiplicative weight update [17], run on the dual space with a *fix step size*. They are not scalable because (1) the number of rounds needed by such algorithm generally depends on how large each coordinate of the subgradient could be (a.k.a., the width of the problem), which is roughly n in our problem, and (2) the time needed to compute the subgradient in each round is linear in n.

The scalable algorithm by Huang and Zhu [17], on the other hand, is a noisy version of the online multiplicative weight update algorithm (e.g., [2]), which use the demand of a single agent as an estimator of the overall demand in each round, iterating through all agents once in a random order. However, it does not seem plausible to avoid having an extra $\sqrt{n}$ factor in the minimum supply requirement using this approach, as it is not only jointly differentially private, but also locally private,[2] in the sense that it can be implemented in a way such that the agents add noises themselves so that even the algorithm never accesses any non-private version of the data. The extra $\sqrt{n}$ factor is ubiquitous in the literature of locally private algorithms [3, 9].

In contrast, our algorithm is a noisy version of the multiplicative weight update algorithm run on the dual space with *different step sizes*, which are optimized according on the scale of the subgradient in each round. Intuitively, it chooses a small step size when the scale of the subgradient is large, to avoid dramatic changes in prices, and a large step size when the scale of the subgradient is small, to ensure a good enough progress. The idea of using different step sizes is widely used in non-private packing algorithms to get width-independent running time (e.g., [20]). Despite being standard in non-private packing, a direct combination of it and how the existing approaches add noises to the subgradients lead to suboptimal minimum supply requirement and/or super-linear running time in n. Instead, we need to further use different noise scales in different rounds that are tailored to the scales of the subgradients and, by induction, the corresponding step sizes. In a round where the scale of the subgradient is large (respectively, small) and the step size is small (respectively, large), the algorithm adds noises at a larger (respectively, smaller) scale to the subgradient and, in some sense, uses up less (respectively, more) of the privacy budget. Setting noise scales adaptively over time introduces several technical difficulties which we will address in details in the technical sections. To get the results in Theorem 1, the noise scale is inversely proportional to the square root of the step size. Hence, our algorithm is more precisely characterized as a noisy version of the multiplicative weight update algorithm run on the dual space with *different step sizes and noise scales*, both of which are optimized according on the scale of the subgradient in each round. Table 1 provides a brief comparison of our algorithm and the existing ones.

1.2 Related Work

The notion of differential privacy is introduced by Dwork et al. [10]. It has evolved through a long line of works to become a standard notion of privacy in theoretical computer science. See Dwork and Roth [12] for a textbook introduction. A particularly related line of works study differentially private algorithms for combinatorial optimization problems and mathematical programs. McSherry and Talwar [22] introduce a generic (yet computationally inefficient) method called the exponential mechanism for privately solving optimization problems whose

[2] This is not explicitly stated in Huang and Zhu [17]. Nonetheless, it follows straightforwardly from the definition of the algorithm.

Table 1 A comparison of the algorithm in this paper and those in previous works. n and m denote the number of agents and number of resources respectively. ϵ and α quantify the privacy and approximation guarantees respectively.

Reference	Algorithm	Min Supply	Running Time (in n)
Hsu et al. [15]	Dual GD	$\tilde{O}\left(\frac{m}{\alpha\epsilon}\right)$	$O(n^3)$
Huang and Zhu [17]	Dual MWU (fixed step size)	$\tilde{O}\left(\frac{\sqrt{m}}{\alpha\epsilon}\right)$	$O(n^3)$
	Dual Online MWU	$\tilde{O}\left(\frac{\sqrt{mn}}{\alpha\epsilon}\right)$	$O(n)$
This paper	Dual MWU (different step sizes)	$\tilde{O}\left(\frac{\sqrt{m}}{\alpha\epsilon}\right)$	$O(n)$

feasible set of output is independent on the dataset (e.g., k-means clustering). It is not applicable to the packing problem considered in this paper, since the set of feasible allocations crucially rely on the dataset. Hsu et al. [16] study what linear programs can be solved in a differentially private manner. Unfortunately, packing linear programs are not among the solvable ones [14].

Subsequently, Kearns et al. [19] introduce a relaxed notion called joint differential privacy. It is still strong enough to provide provable privacy guarantees, but is also flexible enough to allow positive results for problems that cannot be solved under the original notion of differential privacy. This relaxed notion is widely used not only in resource allocation problems [14, 15, 17] such as the packing problem considered in this paper, but also in coordinating large games [24, 6, 23, 19, 7], privacy-presering learning [5, 25], privacy-preserving surveys [13], privacy-preserving prediction markets [8], etc.

Recently, jointly differentially private resource allocation algorithms find further applications in regularizing strategic behaviors in the problem of learning reserve prices online in strategic environments [21].

2 Model

For any positive integer ℓ, let $[\ell]$ denote the set of integers between 1 and ℓ, i.e., $\{1, 2, \ldots, \ell\}$.

Packing

Consider a packing problem with n agents and m resources. Each agent $i \in [n]$ demands a bundle of resources; let a_{ij} denote her demand for each resource $j \in [m]$. Further, agent i has value v_i for getting the bundle, and 0 for not getting it. We assume that a_{ij}'s and v_i's are bounded between 0 and 1, which is standard in the literature of differential privacy. For any agent $i \in [n]$, the demands a_{ij}'s and the value v_i are her private data. Let $U = \{(v, a_1, a_2, \ldots, a_m) \in [0, 1]^{m+1}\}$ denote the data universe. Let $\mathcal{D} \in U^n$ denote a dataset of n agents. For any resource $j \in [m]$, let b_j denote its supply. The goal is then to choose a subset of the agents who get their demanded bundles, such that the sum of the values of the chosen agents is maximized, subject to that the total demand on each resource does not exceed the corresponding supply.

We remark that we can make two simplifying assumptions because the focal point of the jointly differentially private packing problem lies in whether the minimum supply is sufficiently large. First, we may assume without loss of generality (wlog) that the supplies of all resources are equal; otherwise, we may rescale the larger ones down to be equal to the

minimum one. Second, we may focus on fractional solutions wlog; any fractional solution can be converted into an integral one using independent rounding with essentially the same objective and total demands on the resources, since we are in the large supply regime.

Therefore, the problem can be formulated as the following packing linear program:

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_{i\in[n]} v_i x_i \\ \text{subject to} \quad & \textstyle\sum_{i\in[n]} a_{ij} x_i \le b && \forall j \in [m] \\ & 0 \le x_i \le 1 && \forall i \in [n] \end{aligned}$$

Differential Privacy and Joint Differential Privacy

Next, we formally define differential privacy and joint differential privacy with respect to the packing problem. Two datasets $\mathcal{D}, \mathcal{D}' \in U^n$ are i-neighbors if they differ only in the data of the i-th agent, that is, if $\mathcal{D}_j = \mathcal{D}'_j$ for all $j \neq i$. We simply say that they are neighbors if they are i-neighbors for some $i \in [n]$. Further, let $X_i = [0, 1]$ denote the set of feasible decision to each agent $i \in [n]$. Let $X = X_1 \times X_2 \times \cdots \times X_n$ denote the set of feasible outcomes, ignoring the supply constraints. The notion of differential privacy by Dwork et al. [10] requires that the allocation of all agents is chosen from similar distributions for any neighboring datasets in the following sense:

▶ **Definition 2** (Differential Privacy). *A mechanism $\mathcal{M} : U^n \mapsto X$ is (ϵ, δ)-differentially private if for any neighbors $\mathcal{D}, \mathcal{D}' \in U^n$, and any subset of feasible allocations $S \subseteq X$:*

$$\mathbf{Pr}\big[\mathcal{M}(\mathcal{D}) \in S\big] \le \exp(\epsilon) \cdot \mathbf{Pr}\big[\mathcal{M}(\mathcal{D}') \in S\big] + \delta \ .$$

The notion of joint differential privacy by Kearns et al. [19], on the other hand, allows the allocation to each agent to depend non-privately on her own data, so long the allocation to the other agents does not. More precisely, the notion is defined as follows:

▶ **Definition 3** (Joint Differential Privacy). *A mechanism $\mathcal{M} : U^n \mapsto X$ is (ϵ, δ)-jointly differentially private if for any $i \in [n]$, any i-neighbors $\mathcal{D}, \mathcal{D}' \in U^n$, and any subset of feasible allocations to agents other than i, $S_{-i} \subseteq X_{-i}$:*

$$\mathbf{Pr}\big[\mathcal{M}(\mathcal{D})_{-i} \in S_{-i}\big] \le \exp(\epsilon) \cdot \mathbf{Pr}\big[\mathcal{M}(\mathcal{D}')_{-i} \in S_{-i}\big] + \delta \ .$$

The technical connections between the two notions are best explained by the following billboard lemma by Hsu et al. [14].

▶ **Lemma 4** (Billboard Lemma). *Suppose $\mathcal{M} : U^n \mapsto Y$ is (ϵ, δ)-differentially private. Then, for any collection of functions $f_i : U \times Y \mapsto X_i$, $i \in [n]$, the mechanism $\mathcal{M}'$ that allocates to each agent i with $f_i\big(\mathcal{D}_i, \mathcal{M}(\mathcal{D})\big)$ is (ϵ, δ)-jointly differentially private.*

3 Technical Preliminaries

3.1 Lagrangian

The Lagrangian of the packing linear program is:

$$\max_{x\in X} \min_{p\in[0,\infty)^m} \textstyle\sum_{i\in[n]} v_i x_i - \sum_{j\in[m]} p_j \big(\sum_{i\in[n]} a_{ij} x_i - b_j\big) \ .$$

Let $L(x, p)$ denote the partial Lagrangian objective, that is:

$$\begin{aligned} L(x,p) &= \textstyle\sum_{i\in[n]} v_i x_i - \sum_{j\in[m]} \big(\sum_{i\in[n]} a_{ij} x_i - b\big) p_j \\ &= \textstyle\sum_{j\in[m]} b p_j + \sum_{i\in[n]} \big(v_i - \sum_{j\in[m]} a_{ij} p_j\big) x_i \ . \end{aligned}$$

Let $D(p) = \max_{x\in[X]} L(x,p)$ denote the dual objective. We shall interpret p_j as the unit price of resource j for any $j \in [m]$. Given any prices p, an optimal solution $x^*(p)$ of the optimization problem $\max_{x\in[X]} L(x,p)$ is defined as, for any $i \in [n]$:

$$x_i^*(p) = \begin{cases} 1, & \text{if } v_i - \sum_{j\in[m]} a_{ij}p_j \geq 0; \\ 0, & \text{otherwise.} \end{cases}$$

We have the following envelope theorem (see, e.g., Afriat [1]).

▶ **Lemma 5** (Envolope Theorem)**.** *Given any prices p, the derivatives of the Lagrangian objective w.r.t. the prices, i.e., $\nabla_p L(x,p)$, is a sub-gradient of $D(p)$ when $x = x^*(p)$.*

3.2 Truncated Laplacian Distributions

The Laplacian distribution, given mean $\mu \in R$ and scale parameter $b > 0$, is a continuous distribution defined on $(-\infty, +\infty)$ such that the probability density of any $x \in \mathbf{R}$ is:

$$\frac{1}{2b}\exp\left(-\frac{|x-\mu|}{b}\right) .$$

Let $\mathrm{Lap}(\mu, b)$ denote this distribution. It has mean μ and variance $2b^2$.

Further, we will consider the truncated Laplacian distribution with support $[\mu - 1 + \alpha, \mu + 1 - \alpha]$, denoted as $\mathrm{Lap}_{1-\alpha}(\mu, b)$. The probability density of any x in the interval is proportional to that of $\mathrm{Lap}(\mu, b)$; the density is 0 if it is outside the interval. $\mathrm{Lap}_{1-\alpha}(\mu, b)$ also has mean μ, and variance $O(b^2)$. (See below for a formal statement.)

Given any mean $\mu \in [-\alpha, \alpha]$ and target standard deviation (up to a constant factor) $0 < \sigma \leq \alpha$, let $\mathcal{N}(\mu, \sigma) = \mathrm{Lap}_{1-\alpha}(\mu, \sigma)$. We use this notation to emphasize that the family of noise distributions, $\mathcal{N}(\mu, \sigma)$'s, can be replaced by distributions other than the truncated Laplacian distributions, as long as they satisfy the following properties.The proofs are deferred to the full version.

▶ **Lemma 6.** *The noise distributions $\mathcal{N}(\mu, \sigma)$'s satisfy that:*
1. *the mean of $\mathcal{N}(\mu, \sigma)$ is μ;*
2. *the variance of $\mathcal{N}(\mu, \sigma)$ is at most $2\sigma^2$.*

▶ **Lemma 7.** *Suppose $-\alpha \leq \mu_1, \mu_2 \leq \alpha$, and $0 < \sigma_1, \sigma_2 \leq \alpha$ satisfy that for some $0 < \eta \leq \alpha$, $\sigma = \max\{\sigma_1, \sigma_2\}$, and $\delta > 0$: (1) $\left|\frac{1}{\sigma_1^2} - \frac{1}{\sigma_2^2}\right| \leq \frac{\eta}{\alpha\sigma^2}$; (2) $\left|\mu_1 - \mu_2\right| \leq \eta$; and $\delta \leq \frac{\eta \ln(2/\delta)}{\sigma}$. Then, for any $S \subseteq \mathbf{R}$, we have:*

$$\mathbf{Pr}_{z\sim\mathcal{N}(\mu_1,\sigma_1)}\left[z \in S\right] \leq \exp\left(\frac{4\eta\ln(2/\delta)}{\sigma}\right) \cdot \mathbf{Pr}_{z\sim\mathcal{N}(\mu_2,\sigma_2)}\left[z \in S\right] + \delta .$$

4 Our Algorithm

The algorithm follows a primal dual approach, running best response on the primal (i.e., allocation x) and a noisy version of the multiplicative weight update (MWU) method on the dual (i.e., the prices p). Similar approaches are also used in the previous works that study jointly differentially private packing problem (e.g., [15, 17]). The new ingredients of our algorithm are the use of nonuniform step sizes as well as nonuniform noise scales, both of which are meticulously optimized to achieve both scalable running time and the optimal trade-off between privacy and the minimum supply requirement. In contrast, all previous algorithms use a fix step size and a fix noise scale across different rounds.

Algorithm 1 Private Dual MWU with Optimized Step Sizes and Noise Scales.

Input:
- Dataset $\mathcal{D} \in U^n$, represented by a_{ij}'s and v_i's;
- Supply (per resource) b.

Assumptions:
- $b \geq \tilde{O}\left(\frac{\sqrt{m}}{\alpha\epsilon}\right)$;
- We also assume $n \geq b$ as the problem is trivial otherwise.

Parameters:
- Upper bound on dual prices $p_{\max} = \frac{2n}{b}$;
- Initial dual prices p^1 such that $p_j^1 = \frac{p_{\max}}{m+1}$ for any $j \in [m+1]$;
- Upper bound on the sum of step sizes $\eta_{\text{sum}} = \frac{\ln(m+1)}{\alpha b}$.

1: **for** $t = 1, 2, \ldots,$ **until** $\sum_t \eta^t \geq \eta_{\text{sum}}$ (via a private counter) **do**
2: Let $x^t = x^*(p^t)$, i.e., the best response to p^t from the primal viewpoint.
3: Let $\nabla_j D(p^t) = b - \sum_{i \in [n]} a_{ij} x_i^t$ for all $j \in [m]$; let $\nabla_{m+1} D(p^t) = 0$.
4: Let the *step size* and the *noise scale* be:

$$\eta^t = \min\left\{\frac{\alpha}{b}, \frac{\alpha}{\nabla_1 D(p^t)}, \frac{\alpha}{\nabla_2 D(p^t)}, \ldots, \frac{\alpha}{\nabla_m D(p^t)}\right\} \quad , \quad \sigma^t = \frac{\sqrt{m\eta_{\text{sum}}\eta^t \ln(Tm/\delta)}}{\epsilon} \ .$$

5: Draw $\delta_j^t \sim \mathcal{N}\left(\mu_j^t, \sigma^t\right)$ where $\mu_j^t = \eta^t \nabla_j D(p^t)$ for all $j \in [m]$; let $\delta_{m+1}^t = 0$.
6: Let $\hat{p}_j^{t+1} = p_j^t \cdot \exp(-\delta_j^t)$ for all $j \in [m+1]$.
7: Let p^{t+1} be such that $p_j^{t+1} \propto \hat{p}_j^{t+1}$ and $\sum_{j \in [m+1]} p_j^{t+1} = p_{\max}$.
8: **end for**
9: Let T be the number of iterations in the for loop.

Output: $\bar{x} = \frac{1}{\eta_{\text{sum}}} \sum_{t=1}^{T} \eta^t x^t$, where agent i observes $\bar{x}_i$.

See Algorithm 1 for an exposition by pseudocode.

We now describe the algorithm in more details. Let us first explain the basic dynamic of the dual MWU algorithm. For technical reasons, we add a dummy resource with 0 supply, and 0 demands from all agents, and assume that the dual prices sum to a fix and sufficiently large number $p_{\max}$. Starting from some initial guess p^1 of the prices, say, with $p_{\max}$ uniformly distributed among the $m+1$ coordinates, the MWU algorithm repeatedly calculates:

$$x^t = x^*\left(p^t\right) \quad ; \quad p_j^{t+1} \propto p_j^t \cdot \exp\left(-\eta^t \nabla_j D\left(p^t\right)\right) \ , \ \forall j \in [m+1]$$

where $\eta^t > 0$ is the step size of round t. That is, the allocation x^t is the best response to p^t, which induces a subgradient of the dual objective at p^t by Lemma 5. Then, from p^t to p^{t+1}, each coordinate $j \in [m+1]$ decreases exponentially by an amount proportional to corresponding subgradient, before they are rescaled to sum to $p_{\max}$. With appropriate step sizes, standard analysis shows that the weighted average allocation across different rounds, where the weight of each round is its step size, converges to an optimal allocation.[3]

To obtain the desired privacy guarantee, our algorithm updates the prices with zero-mean noises added to the subgradients. Below we discuss the choice of step sizes and noise scales.

Step Sizes. Some standard choices of step sizes include uniform step sizes, i.e., $\eta^t = \eta$, which is used in previous works on jointly private packing algorithms [15, 17], decreasing step sizes, e.g., $\eta^t = \frac{\eta}{t}$, and step sizes inversely proportional to the magnitude of the subgradient, which

[3] The prices also converge, although this is not relevant for our analysis.

are what our algorithm uses (see, e.g., Koufogiannakis and Young [20], for an application in the non-private packing problem). Intuitively, our choice of step sizes ensure that at least one coordinate will be updated by an η amount, which in turns lower bounds the amount of progress made in each round. Our algorithm has one caveat, however, as it further caps the step size by an upper bound, which is set to $\frac{\alpha}{b}$ for technical reasons, so that the noise added in any single round does not affect the result by too much.

Noise Scales. Not surprisingly, our first attempt is to add a uniform amount of noise to every round like the previous jointly private algorithms in the literature [14, 15, 17]. This, however, either requires the minimum supply to be much larger than the best possible, or is not scalable. To see why, let us first consider an overly-simplified argument of why uniform noise scales work in the previous algorithms. Suppose the algorithm takes T rounds in total. By a standard composition theorem of differential privacy (see, e.g., Dwork and Roth [12]), adding noises at a uniform scale $\tilde{O}\big(\frac{\sqrt{T}}{\epsilon}\big)$ is sufficient for achieving (ϵ, δ)-joint differential privacy. Then, if the algorithm uses a uniform step size η, by the standard concentration bound, the cumulative noise summing over T rounds is roughly $\tilde{O}\big(\frac{\eta T}{\epsilon}\big)$. After averaging, this is essentially a fixed amount of noise $\tilde{O}\big(\frac{1}{\epsilon}\big)$ independent of T and η!

With non-uniform step sizes, however, this is no longer true. As a thought experiment, suppose there are $T' \ll T$ rounds that have large step sizes, say, all equal to η; the rest of the rounds can be omitted due to negligible step sizes. Note that the uniform noise scale, i.e., $\tilde{O}\big(\frac{\sqrt{T}}{\epsilon}\big)$, is still determined by the total number of rounds, repeating the above calculation gives that the amount of noise after averaging is $\tilde{O}\big(\frac{\sqrt{T}}{\epsilon\sqrt{T'}}\big) \gg \tilde{O}\big(\frac{1}{\epsilon}\big)$.

The lesson we learned from this though experiment is that the algorithm must choose non-uniform noise scales: smaller noise scales for more important rounds that have larger step sizes; and larger noise scales for less important rounds that have smaller step sizes. More precisely, we optimize the noise scale in each round t to be inversely proportional to the square root of the step size, i.e, $\sqrt{\eta^t}$; this is derived form a Cauchy-Schwarz inequality to balance different aspects of the analysis. (Note that δ_j^t's in Algorithm 1 denote the noises added to the subgradients *multiplied by the corresponding step size* and, hence, its scale is proportional to $\sqrt{\eta^t}$, rather than inversely proportional to it.)

Privacy-preserving Stopping Criteria. Finally, note that the stopping criteria of $\sum_t \eta^t \geq \eta_{\text{sum}}$ must be implemented approximately in a privacy-preserving manner as well. This can be done by maintaining $\sum_t \eta^t$ using a standard technique called private counter (e.g., Chan et al. [4], Dwork et al. [11]). For simplicity of exposition, we will omit this standard component and analyze the algorithm assuming the stopping criteria is implemented exactly.

5 Utility and Time Complexity

This section sketches the analysis of utility guarantees provided by Algorithm 1 and its time complexity, under the assumption that the supply is sufficiently large, i.e., $b \geq \tilde{O}\big(\frac{\sqrt{m}}{\alpha\epsilon}\big)$.

5.1 No-regret Lemma

We first introduce a technical lemma that will serve as the overarching tool in the analysis of the algorithm's approximation guarantees in terms of the objective and constraint violations, and its running time. It states that if we compare the Lagrangian objective achieved by the sequence of x^t's and p^t's computed in the algorithm, and what could have be achieved by replacing p^t's with an arbitrary but fix p, the difference can be bounded. In other words, the dual price sequence has no regret in the terminology of online learning.

▶ **Lemma 8.** *For any p such that $\|p\|_1 = p_{max}$, with high probability, we have:*

$$\sum_{t=1}^{T} \eta^t \big(L(x^t, p^t) - L(x^t, p)\big) \le D_{\mathrm{KL}}(p\|p^1) + \eta_{\mathrm{sum}} \cdot O\big(\alpha n\big) \ .$$

Proof Sketch of Lemma 8. We present a proof sketch of a weaker claim that the inequality holds in expectation, which captures the bottleneck of the analysis. Further showing the stronger claim in the lemma takes a (slightly nonstandard) concentration inequality for martingales. See the full version for a complete proof of the lemma.

Readers familiar with this kind of analysis will find it standard and may directly jump to the end to verify that the contribution from the variance term can be bounded given the noise scales chosen in the algorithm. Fix any step t, we have the followings:

$$\begin{aligned}
\mathbf{E}\big[\eta^t \big(L(x^t, p^t) - L(x^t, p)\big)\big] &= \mathbf{E}\big[\langle \eta^t \nabla D(p^t), p^t - p\rangle\big] \\
&= \mathbf{E}\big[\langle \delta^t, p^t - p\rangle\big] \\
&= \mathbf{E}\left[\left\langle \ln\left(\tfrac{\hat{p}^{t+1}}{p^t}\right), p - p^t\right\rangle\right] \\
&= \mathbf{E}\big[D_{\mathrm{KL}}\big(p\|p^t\big) - D_{\mathrm{KL}}\big(p\|\hat{p}^{t+1}\big) + D_{\mathrm{KL}}\big(p^t\|\hat{p}^{t+1}\big)\big] \\
&\le \mathbf{E}\big[D_{\mathrm{KL}}\big(p\|p^t\big) - D_{\mathrm{KL}}\big(p\|p^{t+1}\big) + D_{\mathrm{KL}}\big(p^t\|\hat{p}^{t+1}\big) - D_{\mathrm{KL}}\big(p^{t+1}\|\hat{p}^{t+1}\big)\big] \\
&\le \mathbf{E}\big[D_{\mathrm{KL}}\big(p\|p^t\big) - D_{\mathrm{KL}}\big(p\|p^{t+1}\big) + D_{\mathrm{KL}}\big(p^t\|\hat{p}^{t+1}\big)\big] \ .
\end{aligned}$$

We abuse notation and let $\ln\left(\frac{\hat{p}^{t+1}}{p^t}\right)$ denote a vector whose j-th coordinate is $\ln\left(\frac{\hat{p}_j^{t+1}}{p_j^t}\right)$ for any $j \in [m+1]$, in the 4th line of the above equation. The last two inequalities follow by the generalized Pythagorean theorem and the non-negativity of divergences, respectively.

Summing over $t \in [T]$, the first two terms form a telescopic sum; it is bounded by the first term on the right-hand-side of the inequality stated in the lemma. For the last term, we have:

$$\begin{aligned}
\mathbf{E}\big[D_{\mathrm{KL}}\big(p^t\|\hat{p}^{t+1}\big)\big] &= \mathbf{E}\big[\textstyle\sum_{j=1}^{m+1} \left(p_j^t \ln\left(\frac{p_j^t}{\hat{p}_j^{t+1}}\right) - p_j^t + \hat{p}_j^{t+1}\right)\big] \\
&= \mathbf{E}\big[\textstyle\sum_{j=1}^{m+1} p_j^t\big(\delta_j^t - 1 + \exp\big(-\delta_j^t\big)\big)\big] \\
&\le \mathbf{E}\big[\textstyle\sum_{j=1}^{m+1} p_j^t\big(\delta_j^t\big)^2\big] = \mathbf{E}\left[\textstyle\sum_{j=1}^{m+1} p_j^t\big(\mathbf{E}\big[\delta_j^t\big]^2 + \mathbf{Var}\big[\delta_j^t\big]\big)\right] \ .
\end{aligned}$$

Note that the step sizes ensure $\mathbf{E}\big[\delta_j^t\big] \le \alpha$. The first part on the right-hand-side sums to:

$$\begin{aligned}
\mathbf{E}\left[\textstyle\sum_{j=1}^{m+1}\sum_{t=1}^{T} p_j^t \cdot \mathbf{E}\big[\delta_j^t\big]^2\right] &\le \alpha\mathbf{E}\left[\textstyle\sum_{j=1}^{m+1} p_j^t \cdot \sum_{t=1}^{T} \big|\mathbf{E}\big[\delta_j^t\big]\big|\right] \\
&\le \alpha\mathbf{E}\left[\textstyle\sum_{j=1}^{m+1} p_j^t \cdot \Big(2b\sum_{t=1}^{T}\eta^t - \sum_{t=1}^{T}\eta^t\mathbf{E}\big[\delta_j^t\big]\Big)\right] \\
&= \eta_{\mathrm{sum}} \cdot O(\alpha n) - \alpha\mathbf{E}\left[\textstyle\sum_{j=1}^{m+1} p_j^t \sum_{t=1}^{T}\eta^t\mathbf{E}\big[\delta_j^t\big]\right] \\
&\le \eta_{\mathrm{sum}} \cdot O(\alpha n) - \alpha\mathbf{E}\left[\textstyle\sum_{j=1}^{m+1} p_j^t \sum_{t=1}^{T}\eta^t\nabla_j D(p^t)\right] \\
&= \eta_{\mathrm{sum}} \cdot O(\alpha n) - \alpha\mathbf{E}\left[\textstyle\sum_{j=1}^{m+1} \eta^t\big(L(x^t, p^t) - \sum_{i=1}^{n} v_i x_i^t\big)\right] \\
&\le \eta_{\mathrm{sum}} \cdot O(\alpha n) - \alpha\mathbf{E}\left[\textstyle\sum_{j=1}^{m+1} \eta^t L(x^t, p^t)\right] \le \eta_{\mathrm{sum}} \cdot O(\alpha n) \ .
\end{aligned}$$

Finally, the contribution from the variance part sums to:

$$\begin{aligned}
\mathbf{E}\left[\sum_{j=1}^{m+1}\sum_{t=1}^{T} p_j^t \cdot \mathbf{Var}\left[\delta_j^t\right]\right] &\le \mathbf{E}\left[\sum_{j=1}^{m+1}\sum_{t=1}^{T} p_j^t \cdot 4\left(\sigma^t\right)^2\right] && \text{(Lemma 7)}\\
&\le \tilde{O}\left(\sum_{t=1}^{T} \tfrac{p_{\max} m \eta_{\text{sum}} \eta^t}{\epsilon^2}\right) && \text{(Definition of } \sigma^t\text{'s)}\\
&\le \tilde{O}\left(\tfrac{p_{\max} m \eta_{\text{sum}}^2}{\epsilon}\right)\\
&= \eta_{\text{sum}} \cdot \tilde{O}\left(\tfrac{n}{\alpha\epsilon^2 b^2}\right) \le \eta_{\text{sum}} \cdot O(\alpha n)\ . && \text{(Assumption on } b\text{)}
\end{aligned}$$

Putting them together proves the lemma. ◀

By the choice of p^1, $p_{\max}$, η_{sum}, the fact that $L(x^t, p^t) \ge \text{OPT}$ since x^t's are best responses, and the definition of $\bar{x}$, we further have it in a simpler form as a corollary.

▶ **Lemma 9.** *For any p such that $\|p\|_1 = p_{max}$, with high probability, we have:*

$$\text{OPT} - L\left(\bar{x}, p\right) \le O\left(\alpha n\right)\ .$$

5.2 Approximate Optimality

We now argue that the algorithm gets an objective that is optimal up to an $O(\alpha n)$ additive factor, with the understanding that further improving it to an αn factor does not affect any of the asymptotic bound. To do so, simply let p be such that the first m coordinates are all equal to 0, and the last dummy coordinate equals $p_{\max}$. Then, we have that $L(\bar{x}, p) = \sum_{i=1}^{n} v_i \bar{x}_i = \text{ALG}$. The claim then follows from Lemma 9.

5.3 Feasibility

Next, we argue that the algorithm provides an allocation $\bar{x}$ that is approximately feasible in the sense that the total demand for each resource is at most $\left(1 + O(\alpha)\right)b$. To convert it into an exactly feasible solution as stated in Theorem 1, we can simply scale the allocation down by a $1 + O(\alpha)$ factor, at the cost of further reducing the objective by at most $O(\alpha n)$.

Suppose resource j^* has the largest demand. Let $s = \sum_{i=1}^{n} a_{ij^*}\bar{x}_i - b$ the gap between this demand and the supply b. Let p be such that all but the j^*-th coordinate are equal to 0, and the j^*-th coordinate is equal to $p_{\max}$. Then, we have:

$$L\left(\bar{x}, p\right) = \text{ALG} - p_{\max} s \le \left(1 + \tfrac{s}{b}\right)\text{OPT} - p_{\max} s\ .$$

By Lemma 9, we get that:

$$\left(p_{\max} - \tfrac{\text{OPT}}{b}\right) s \le O\left(\alpha n\right)\ .$$

Putting together with our choice of $p_{\max} = \frac{2n}{b}$, and that $\text{OPT} \le n$, we get $s \le O\left(\alpha b\right)$.

5.4 Time Complexity

We next show that the number of iterations of the for loop is upper bounded by a polynomial of the parameters other than n. More precisely, we show that $T \le O\left(\frac{m \ln(m+1)}{\alpha^2}\right)$.

To do so, we classify the iterations into $m+1$ types, according to how the step size is chosen. If the step size is $\eta^t = \frac{\alpha}{b}$, we call it type 0. Otherwise, if the step size is $\eta^t = \frac{\alpha}{\nabla_j D(p^t)}$ for some $j \in [m]$, we call it type j.

First, the number of type 0 iterations is at most $\frac{\ln(m+1)}{\alpha^2}$, before the sum of the step sizes exceeds $\eta_{\text{sum}} = \frac{\ln(m+1)}{\alpha b}$.

Next, consider the iterations of type j, for each resource $j \in [m]$. In each iteration t of type j, we have that $\eta^t \nabla_j D(p^t) = \alpha$. We argue that there cannot be too many such iterations, because $\sum_{t=1}^{T} \eta^t \nabla_j D(p^t)$ is upper bounded as a result of the approximate feasibility of $\bar{x}$, which follows from Lemma 9 as argued in the Section 5.3.

On the one hand, we have:

$$\textstyle\sum_{t=1}^{T} \eta^t \nabla_j D(p^t) = \sum_{t=1}^{T} \eta^t \big(\sum_{i=1}^{n} a_{ij} \bar{x}_i - b \big) \le \eta_{\text{sum}} \cdot O(\alpha b) = \tilde{O}(1) \ .$$

On the other hand, suppose there are T_j such iterations, we have:

$$\begin{aligned} \textstyle\sum_{t=1}^{T} \eta^t \nabla_j D(p^t) &= \textstyle\sum_{t \in T_j} \alpha + \sum_{t \notin T_j} \eta^t \nabla_j D(p^t) \\ &\ge \textstyle\sum_{t \in T_j} \alpha + \sum_{t \notin T_j} \eta^t (-b) \ge \alpha |T_j| - 2b\eta_{\text{sum}} = \alpha |T_j| - \frac{2\ln(m+1)}{\alpha} \ . \end{aligned}$$

Putting together gives $|T_j| \le \frac{3\ln(m+1)}{\alpha^2}$.

6 Privacy: Proof Sketch

This section sketches the proof that Algorithm 1 is (ϵ, δ)-jointly differentially private. See the full version for a complete argument.

First, consider any fix step t. Suppose $\mathcal{D}$ and $\widetilde{\mathcal{D}}$ are two neighboring datasets. Further suppose the random realizations of the first $t-1$ iterations are such that $x^1, x^2, \ldots, x^{t-1}$ and $p^1, p^2, \ldots, p^{t-1}$ are identical on $\mathcal{D}$ and $\widetilde{\mathcal{D}}$. Let us consider the privacy of step t alone.

In fact, let us focus on a resource $j \in [m]$. How does the distribution from which δ_j^t is drawn differ on the two datasets? Suppose the step sizes are η and $\tilde{\eta}$ respectively. Then, by the choice of step sizes in the algorithm and that $\nabla_j D(p^t)$ differs by at most 1, we have:

$$\big| \tfrac{1}{\eta} - \tfrac{1}{\tilde{\eta}} \big| \le \tfrac{1}{\alpha} \ .$$

Suppose the standard deviations are σ and $\tilde{\sigma}$, and the means are μ and $\tilde{\mu}$, respectively on the two datasets. They satisfy that:

$$\begin{aligned} \big| \tfrac{1}{\sigma^2} - \tfrac{1}{\tilde{\sigma}^2} \big| &= \tfrac{\epsilon^2}{m\eta_{\text{sum}} \ln(T/\delta)} \big| \tfrac{1}{\eta} - \tfrac{1}{\tilde{\eta}} \big| \le \tfrac{\epsilon^2}{\alpha m \eta_{\text{sum}} \ln(Tm/\delta)} = \tfrac{\eta}{\alpha \sigma^2} \ , \\ \big| \mu - \tilde{\mu} \big| &\le \eta \big| \tfrac{\mu}{\eta} - \tfrac{\tilde{\mu}}{\tilde{\eta}} \big| + \tilde{\mu} \eta \big| \tfrac{1}{\eta} - \tfrac{1}{\tilde{\eta}} \big| \le \eta + \alpha \eta \alpha^{-1} = 2\eta \ , \end{aligned}$$

where $\big| \frac{\mu}{\eta} - \frac{\tilde{\mu}}{\tilde{\eta}} \big| \le 1$ because it measures the difference in $\nabla_j D(p^t)$.

Also it's easy to verify that $\eta^t \le \alpha$. Therefore, by Lemma 7, we have that the update from price p_j^t to $\tilde{p}_j^{t+1}$ posted in round t for resource j is $(\epsilon^t, \frac{\delta}{Tm})$-differentially private for:

$$\epsilon^t = O\big(\tfrac{\eta^t}{\sigma^t}\big) = \tfrac{\epsilon \sqrt{\eta^t \ln(Tm/\delta)}}{\sqrt{m \eta_{\text{sum}}}} \ .$$

Then, intuitively by the composition theorem (see, e.g., Dwork and Roth [12]), the price sequence is (ϵ, δ)-differentially private because:

$$\sum_{t=1}^{T} \sum_{j=1}^{m} \big(\epsilon^t\big)^2 = O\big(\tfrac{\epsilon^2 \eta^t \ln^2(Tm/\delta)}{m \eta_{\text{sum}}}\big) = O\big(\tfrac{\epsilon^2}{\ln(2/\delta)}\big) \ .$$

However, note that the standard statement of the composition theorem does not directly apply here since the privacy budget ϵ^t in each step is chosen adaptively. Fortunately, the underlying argument still goes through. Again, see the full version for details.

Finally, the privacy guarantee of Algorithm 1 follows by the billboard lemma (Lemma 4).

References

1 SN Afriat. Theory of maxima and the method of Lagrange. *SIAM Journal on Applied Mathematics*, 20(3):343–357, 1971.

2 Shipra Agrawal and Nikhil R Devanur. Fast algorithms for online stochastic convex programming. In *SODA*, pages 1405–1424. SIAM, 2014.

3 Raef Bassily and Adam Smith. Local, private, efficient protocols for succinct histograms. In *STOC*, pages 127–135. ACM, 2015.

4 TH Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. In *ICALP*, pages 405–417. Springer, 2010.

5 Rachel Cummings, Stratis Ioannidis, and Katrina Ligett. Truthful linear regression. In *COLT*, pages 448–483, 2015.

6 Rachel Cummings, Michael Kearns, Aaron Roth, and Zhiwei Steven Wu. Privacy and truthful equilibrium selection for aggregative games. In *WINE*, pages 286–299. Springer, 2015.

7 Rachel Cummings, Katrina Ligett, Jaikumar Radhakrishnan, Aaron Roth, and Zhiwei Steven Wu. Coordination complexity: Small information coordinating large populations. In *ITCS*, pages 281–290. ACM, 2016.

8 Rachel Cummings, David M Pennock, and Jennifer Wortman Vaughan. The possibilities and limitations of private prediction markets. In *EC*, pages 143–160. ACM, 2016.

9 John C Duchi, Michael I Jordan, and Martin J Wainwright. Local privacy and statistical minimax rates. In *FOCS*, pages 429–438. IEEE, 2013.

10 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284. Springer, 2006.

11 Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. Differential privacy under continual observation. In *STOC*, pages 715–724. ACM, 2010.

12 Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

13 Arpita Ghosh, Katrina Ligett, Aaron Roth, and Grant Schoenebeck. Buying private data without verification. In *EC*, pages 931–948. ACM, 2014.

14 Justin Hsu, Zhiyi Huang, Aaron Roth, Tim Roughgarden, and Zhiwei Steven Wu. Private matchings and allocations. *SIAM Journal on Computing*, 45(6):1953–1984, 2016.

15 Justin Hsu, Zhiyi Huang, Aaron Roth, and Zhiwei Steven Wu. Jointly private convex programming. In *SODA*, pages 580–599. SIAM, 2016.

16 Justin Hsu, Aaron Roth, Tim Roughgarden, and Jonathan Ullman. Privately solving linear programs. In *ICALP*, pages 612–624. Springer, 2014.

17 Zhiyi Huang and Xue Zhu. Near optimal jointly private packing algorithms via dual multiplicative weight update. In *SODA*, pages 343–357. SIAM, 2018.

18 Zhiyi Huang and Xue Zhu. Scalable and Jointly Differentially Private Packing. *CoRR*, abs/1905.00767, 2019. `arXiv:1905.00767`.

19 Michael Kearns, Mallesh Pai, Aaron Roth, and Jonathan Ullman. Mechanism design in large games: incentives and privacy. In *ITCS*, pages 403–410. ACM, 2014.

20 Christos Koufogiannakis and Neal E Young. A nearly linear-time PTAS for explicit fractional packing and covering linear programs. *Algorithmica*, 70(4):648–674, 2014.

21 Jinyan Liu, Zhiyi Huang, and Xiangning Wang. Learning Optimal Reserve Price against Non-myopic Bidders. In *NeurIPS*, pages 2042–2052, 2018.

22 Frank McSherry and Kunal Talwar. Mechanism Design via Differential Privacy. In *FOCS*, pages 94–103. IEEE Computer Society, 2007.

23 Ryan Rogers, Aaron Roth, Jonathan Ullman, and Zhiwei Steven Wu. Inducing approximately optimal flow using truthful mediators. In *EC*, pages 471–488. ACM, 2015.

24 Ryan M Rogers and Aaron Roth. Asymptotically truthful equilibrium selection in large congestion games. In *EC*, pages 771–782. ACM, 2014.

25 Roshan Shariff and Or Sheffet. Differentially Private Contextual Linear Bandits. In *NeurIPS*, pages 4301–4311, 2018.

26 Shang-Hua Teng. Scalable algorithms for data and network analysis. *Foundations and Trends® in Theoretical Computer Science*, 12(1–2):1–274, 2016.

Local Search Breaks 1.75 for Graph Balancing

Klaus Jansen
Department of Computer Science, Christian-Albrechts-Universität, Kiel, Germany
kj@informatik.uni-kiel.de

Lars Rohwedder
Department of Computer Science, Christian-Albrechts-Universität, Kiel, Germany
lro@informatik.uni-kiel.de

Abstract

Graph Balancing is the problem of orienting the edges of a weighted multigraph so as to minimize the maximum weighted in-degree. Since the introduction of the problem the best algorithm known achieves an approximation ratio of 1.75 and it is based on rounding a linear program with this exact integrality gap. It is also known that there is no $(1.5 - \epsilon)$-approximation algorithm, unless P = NP. Can we do better than 1.75?

We prove that a different LP formulation, the configuration LP, has a strictly smaller integrality gap. Graph Balancing was the last one in a group of related problems from literature, for which it was open whether the configuration LP is stronger than previous, simple LP relaxations. We base our proof on a local search approach that has been applied successfully to the more general Restricted Assignment problem, which in turn is a prominent special case of makespan minimization on unrelated machines. With a number of technical novelties we are able to obtain a bound of 1.749 for the case of Graph Balancing. It is not clear whether the local search algorithm we present terminates in polynomial time, which means that the bound is non-constructive. However, it is a strong evidence that a better approximation algorithm is possible using the configuration LP and it allows the optimum to be estimated within a factor better than 1.75.

A particularly interesting aspect of our techniques is the way we handle small edges in the local search. We manage to exploit the configuration constraints enforced on small edges in the LP. This may be of interest to other problems such as Restricted Assignment as well.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases graph, approximation algorithm, scheduling, integrality gap, local search

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.74

Category Track A: Algorithms, Complexity and Games

Related Version A full version can be found at `https://arxiv.org/abs/1811.00955`.

Funding Research was supported by German Research Foundation (DFG) project JA 612/15-2.

1 Introduction

In this paper we consider weighted, undirected multigraphs that may contain loops. We write such a multigraph as $G = (V, E, r, w)$, where V is the set of vertices, E is the set of edge identities, and r is a function $E \to \{\{u, v\} : u, v \in V\}$ that defines the endpoints for every edge. Note that in the definition above we allow $u = v$, which describes a loop. E is often defined as a set of vertex pairs. We use the function r instead, since it avoids some issues due to multigraphs. The weight function $w : E \to \mathbb{R}_{>0}$ assigns positive weights to the edges. In the Graph Balancing problem we want to compute an orientation of the edges, i.e., one of the ways to turn the graph into a directed graph. The goal is to minimize the maximum weighted in-degree over all vertices, that is $\max_{v \in V} \sum_{e \in \delta^-(v)} w(e)$, where $\delta^-(v)$ are the incoming edges of vertex v in the resulting digraph. Apart from being an arguably natural problem, Graph Balancing has been of particular interest to the scheduling community.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 74; pp. 74:1–74:14

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

It is one of the simplest special cases of makespan minimization on unrelated machines for which an inapproximability bound of $1.5-\epsilon$ is known, which is already the best that is known in the general problem. In the interpretation as a scheduling problem, machines correspond to vertices and jobs to edges, i.e., each job has only two potential machines to which it can be assigned. The problem was introduced by Ebenlendr, Krčál, and Sgall [9] and they gave a polynomial time 1.75-approximation for it. Indepenently and under a different name, Asahiro et al. [4] studied the problem and showed that no $(1.5-\epsilon)$-approximation is possible unless P = NP. The algorithm by Ebenlendr et al. rounds the solution of a particular linear programming formulation. This appears to be the best one can hope for using their techniques, since the ratio between integral optimum and fractional optimum of the LP, the integrality gap, can be arbitrarily close to 1.75 [9]. Using a completely different approach to [9], Huang and Ott developed a purely combinatorial algorithm for the problem [10]. With $5/3+4/21 \approx 1.857$, however, their approximation ratio is inferior to the original algorithm. Another algorithm for Graph Balancing, developed by Wang and Sitters [21], achieves an approximation ratio of $11/6 \approx 1.833$, i.e., also worse than the original, but notable for being simpler. For the special case of only two different edge weights, three independent groups found a tight 1.5-approximation [7, 10, 17].

A good candidate for a stronger linear program to that from [9] is the configuration LP. It was introduced by Bansal and Sviridenko for the more general problem Scheduling on Unrelated Machines and the closely related Santa Claus problem [5]. It is easy to show that this LP is at least as strong as the LP from [9] (see the same paper), i.e., the integrality gap must be at most 1.75 as well. The best lower bound known is 1.5 (see for instance [11], this holds even for the case of Graph Balancing). In recent literature, the configuration LP has enabled breakthroughs in the restricted variants for both of the problems above [3, 19]. The restricted variant of Scheduling on Unrelated Machines (also known as Restricted Assignment) can be seen as Graph Balancing with hyperedges. In particular, it contains the Graph Balancing problem as a special case. In this setting, the configuration LP was shown first to have an integrality gap of at most $33/17 \approx 1.941$ [19], which was improved to $11/6 \approx 1.833$ by us [13]. This non-constructive proof is by a local search algorithm that is not known to terminate in polynomial time. In this paper, we present a sophisticated local search algorithm for Graph Balancing and obtain the following result.

▶ **Theorem 1.** *The integrality gap of the configuration LP is at most* 1.749 *for* Graph Balancing.

In other words, it is stronger than the LP from [9]. Although this does not give a polynomial time approximation algorithm, it is strong evidence that such an algorithm can be developed using the configuration LP. Furthermore, the optimal solution can be estimated in polynomial time within a factor of $1.749+\epsilon$ for any $\epsilon > 0$ by approximating the configuration LP. We emphasize that the purpose of this paper is to show a separation between the configuration LP and the previously used LP relaxation. The constants in the proof are not optimized. We chose to keep the case analysis (which is already difficult) and constants as simple as possible instead of improving the third decimal place. A summary of results regarding the configuration LP is given in Table 1. In fact, for all of the problems except for Graph Balancing it was known whether the configuration LP improves over the previous state-of-the-art. Our work in [15] indicates that earlier bounds on the integrality gap of the configuration LP in related problems disregard many constraints enforced on small edges/jobs, but that without them the LP might be much weaker. More precisely, these proofs used only properties that are already enforced by a weaker configuration LP that allows small edges/jobs to appear fractionally in a configuration. This weaker LP, however, has an integrality gap strictly

Table 1 Integrality gap of the configuration LP for various problems.

	Lower bound	Upper bound
Scheduling on Unrelated Machines	2	2 [16]
⊃ Restricted Assignment	1.5	1.833.. [13]
⊃ Graph Balancing	1.5	**1.749**
⊃ Unrelated Graph Balancing	2 [20]	2
Santa Claus	∞ [5]	∞ [5]
⊃ Restricted Santa Claus	2	3.833.. [14, 8]
⊃ Max-Min Unrelated Graph Balancing	2	2 [20]

higher than 1.5, whereas for the configuration LP it is still open whether 1.5 is the correct answer. The backbone of our new proof is the utilization of these small edge constraints (see end of Section 3.1). This may be relevant to other related local search based proofs as well.

Other related work. The problem of minimizing the maximum out-degree is equivalent to the maximum in-degree. The very similar problem of maximizing the minimum in- or out-degree has been settled by Wiese and Verschae [20]. They gave a 2-approximation and this is the best possible assuming P $\neq$ NP. Surprisingly, this holds even in the unrelated case when the value of an edge may be different on each end. They do not use the configuration LP, but it is easy to also get a bound of 2 on its integrality gap using their ideas. For the restricted case (a special case of the unrelated one) this bound of 2 was already proven by [6]. We are not aware of any evidence that Graph Balancing is easier on simple weighted graphs (without multiedges and loops). The same reduction for the state-of-the-art lower bound holds even in that case. A number of recent publications deal with the important question on how related local search algorithms can be turned into efficient algorithms [1, 2, 12, 18].

2 Preliminaries

Notation. For some $v \in V$ we will denote by $\delta(v)$ the incident edges, i.e. those $e \in E$ with $v \in r(e)$. When a particular orientation is clear from the context, we will write $\delta^-(v)$ for the incoming edges and $\delta^+(v)$ for the outgoing edges of a vertex. For some $F \subseteq E$ we will denote by $\delta_F(v)$ the incident edges of v restricted to F and $\delta_F^-(v)$, $\delta_F^+(v)$ accordingly. For some $e \in E$ we will describe by $t(e) \in r(e)$ the vertex it is oriented towards and by $s(e) \in r(e)$ the vertex it is leaving. For a loop e, i.e., $r(e) = \{v\}$ for some $v \in V$, it always holds that $t(e) = s(e)$. For a subset of edges $S \subseteq E$ we will write $w(S)$ for $\sum_{e \in S} w(e)$ and similar for other functions over the edges.

LP relaxations. The following linear programs have no objective functions. Instead they parameterized by τ, the makespan. The optimum is the lowest τ for which it is feasible. This will be denoted by OPT^* (referring to the configuration LP in the rest of the paper). First we look at the assignment LP by Lenstra, Shmoys, and Tardos [16]. It has a variable $x_{e,v}$ for every vertex v and incident edge $e \in \delta(v)$, which indicates whether e is oriented towards v.

The assignment LP.

$$\sum_{e \in \delta(v)} w(e) \cdot x_{e,v} \leq \tau \quad \forall v \in V, \qquad \sum_{v \in r(e)} x_{e,v} = 1 \quad \forall e \in E, \qquad x_{e,v} \in [0,1]$$

The first constraint ensures that no vertex has more than weight τ of edges oriented towards it. The second one describes that each edge is oriented towards one vertex. The assignment LP has an integrality gap of 2. Let B denote the big edges e, which have $w(e) > 0.5 \cdot \tau$. It is clear that an integral orientation can assign at most one such edge to each vertex. Ebenlendr et al. [9] show that adding the constraint $\sum_{e \in \delta_B(v)} x_{e,v} \leq 1 \quad \forall v \in V$ improves the integrality gap to 1.75. They also give other LP relaxations, but show that none of them have an integrality gap strictly better than 1.75.

Now we will introduce the configuration LP. A configuration is a subset of edges that can be oriented towards a particular vertex without exceeding a particular makespan τ. Formally, we define the configurations of a vertex v and a makespan τ as $\mathcal{C}(v, \tau) := \{C \subseteq \delta(v) : w(C) \leq \tau\}$. The configuration LP now assigns fractions of configurations to each machine.

Primal of the configuration LP.	Dual of the configuration LP.
$\sum_{v \in V} \sum_{C \in \mathcal{C}(v,\tau)} x_{v,C} \leq 1 \quad \forall v \in V$	$\min \sum_{v \in V} y_v - \sum_{e \in E} z_e$
$\sum_{v \in r(e)} \sum_{C \in \mathcal{C}(v,\tau): e \in C} x_{v,C} \geq 1 \quad \forall e \in E$	s.t. $\sum_{e \in C} z_e \leq y_v \quad \forall v \in V, C \in \mathcal{C}(v,\tau)$
$x_{v,C} \geq 0$	$y, z \geq 0$

Although the configuration LP has exponential size, a $(1+\epsilon)$-approximation can be computed in polynomial time for every $\epsilon > 0$ [5]. We are particularly interested in the dual of the configuration LP (which is constructed after adding the objective function $\min (0, \ldots, 0)^T x$). Note that τ is considered a constant in both the primal and the dual. A common idea for proving τ is lower than the optimum is to show that the dual is unbounded for τ (instead of directly showing that the primal is infeasible for τ).

▶ **Lemma 2.** *If there exists $y : V \to \mathbb{R}_{\geq 0}$, and $z : E \to \mathbb{R}_{\geq 0}$, such that $\sum_{e \in C} z(e) \leq y(v)$ for all $v \in V, C \in \mathcal{C}(v, \tau)$ and $\sum_{v \in V} y(v) < \sum_{e \in E} z(e)$, then $\tau < \mathrm{OPT}^*$.*

This holds because y, z is a feasible solution with negative objective value for the dual and so are the same values scaled by any $\alpha > 0$. This way an arbitrarily low objective value can be obtained. Lemma 2 can be seen as a generalization of a space argument: Consider $y(v) = \tau$ and $z(e) = w(e)$. Then for all $v \in V, C \in \mathcal{C}(v, \tau)$, $\sum_{e \in C} z(e) = \sum_{e \in C} w(e) \leq \tau = y(v)$. Thus, by the Lemma we have that $|V| \cdot \tau = \sum_{v \in V} y(v) < \sum_{e \in E} z(e) = w(E)$ implies $\tau < \mathrm{OPT}^*$.

3 Graph Balancing in a special case

To introduce our techniques, we first consider a simplified case where $w(e) \in (0, 0.5] \cup \{1\}$ for each $e \in E$ and the configuration LP is feasible for 1. We will show that there exists an orientation with maximum weighted in-degree $1 + R$ where $R = 0.74$. The proof for the general case (with a slightly worse rate) can be found in the full version of the paper.

▶ **Definition 3** (Tiny, small, big edges). *We call an edge e* tiny, *if $w(e) \leq 1 - R$;* small, *if $1 - R < w(e) \leq 1/2$; and* big, *if $w(e) = 1$. We will write for the tiny, small, and big edges $T \subseteq E, S \subseteq E$, and $B \subseteq E$, respectively.*

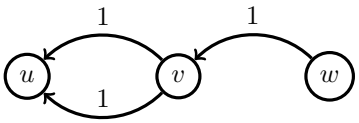
▶ **Definition 4** (Good and bad vertices). *For a given orientation, we call a vertex v* good, *if $w(\delta^-(v)) \leq 1 + R$. A vertex is* bad, *if it is not good.*

The local search algorithm starts with an arbitrary orientation and flips edges until all vertices are good. During this process, a vertex that is already good will never be made bad. It is then proved that (1) the algorithm terminates and (2) when it cannot find a useful edge to flip, the configuration LP also cannot distribute the edges well, i.e., $\mathrm{OPT}^* > 1$, a contradiction.

3.1 Informal overview

Before we give a formal definition of the local search algorithm, we devote this section to giving intuition. We start with very easy algorithms and give challenging instances that motivate the more advanced ideas.

As a toy algorithm consider the following: Start with an arbitrary orientation and repeat until all vertices are good. Whenever there is an edge oriented towards a bad vertex such that its other vertex is good and would remain good even if the edge was flipped (this is called a valid flip), flip this edge. In the following example, both LP and integral optimum are 1. Suppose the algorithm tries to obtain a solution of makespan $2-\epsilon$ with $\epsilon > 0$.



u is bad, v and w are good. However, the algorithm cannot flip one of the edges between u and v, because this would make v bad. It will not flip the edge between v and w, because v is already good. Hence, the algorithm fails. Obviously, in this example we should flip the edge between v and w and then fix u. Let us try to integrate this in the algorithm. We introduce the concept of *pending flips*. When the algorithm wants to flip an edge, but it cannot, because this would make a vertex bad, we add this edge to a list of pending flips. These will be executed once the flip is valid. In the example above, the algorithm could add the edges between u and v to the pending flips, but would not change their orientation, yet. For a pending flip e let us call $s(e)$ the *prospect vertex* and $t(e)$ the *current vertex*. As seen in the example above a sensible local search algorithm should try to move edges away from pending flips' prospect vertices as well (in addition to the bad vertices).

We now state the second toy algorithm. Initialize the pending flips as an empty list. Repeat the following until all vertices are good. Let U be the set of vertices that are either bad or prospect vertices of pending flips. Find an edge from V/U to U and add it to the pending flips. As long as there is a valid pending flip, (1) execute it and (2) delete all pending flips added after it. (2) is to ensure obsolete pending flips are removed. Without it there can be situations where a pending flip has its current vertex in $V \setminus U$, because the pending flip that initially led us to adding it has been executed. This algorithm always succeeds for makespan 1.75 in the special case where weights are in $(0, 0.5] \cup \{1\}$ and $\mathrm{OPT}^* = 1$. We will quickly go over the arguments, since it gives a good idea on how to use the dual of the configuration LP. At this point we will only argue that the algorithm does not get stuck. Normally, we would also have to prove that it terminates (we omit this for sake of brevity). Suppose toward contradiction that the algorithm gets stuck, i.e., there is no valid pending flip and there are no edges from V/U to U. Let F be the big edges e with $t(e) \in U$. We set

$$z(e) = \begin{cases} w(e) & \text{if } t(e) \in U, \\ 0 & \text{otherwise.} \end{cases}$$

$$y(v) = \begin{cases} w(\delta^-(v)) + \frac{1}{4}|\delta_F^+(v)| - \frac{1}{4}|\delta_F^-(v)| & \text{if } v \in U \text{ and } v \text{ is good,} \\ w(\delta^-(v)) + \frac{1}{4}|\delta_F^+(v)| - \frac{1}{4}|\delta_F^-(v)| - 0.1 & \text{if } v \in U \text{ and } v \text{ is bad,} \\ \frac{1}{4}|\delta_F^+(v)| - \frac{1}{4}|\delta_F^-(v)| & \text{if } v \in V \setminus U. \end{cases}$$

What is left to do is to check that for these values the premise of Lemma 2 is fulfilled. Let $v \in V$ and $C \in \mathcal{C}(v, 1)$. Recall that by definition, $C \subseteq \delta(v)$ and $w(C) \leq 1$. We have to verify that $z(C) \leq y(v)$.

Case 1: $v \in V \setminus U$. Since the algorithm is stuck, there is no edge $e \in \delta^+(v)$ with $t(e) \in U$. Hence, $z(e) = 0$ for all $e \in \delta(v)$ and $z(C) = 0$. By definition of F we have that $\delta_F^-(v) = \emptyset$. Thus, $y(v) \geq 0 = z(C)$.

Case 2: $v \in U$. If v is bad, then $w(\delta^-(v)) > 1.75$ and

$$y(v) \geq \begin{cases} w(\delta_F^-(v)) - \frac{1}{4}|\delta_F^-(v)| - 0.1 \geq \frac{3}{4}|\delta_F^-(v)| - 0.1 \geq 1.4 > z(C) & \text{if } |\delta_F^-(v)| \geq 2, \\ w(\delta^-(v)) - \frac{1}{4}|\delta_F^-(v)| - 0.1 > 1.75 - \frac{1}{4} - 0.1 \geq 1.4 > z(C) & \text{if } |\delta_F^-(v)| \leq 1. \end{cases}$$

If $v \in U$ is good, then it is the prospect vertex of some pending flip e. An invariant of the algorithm is that all pending flips have their current vertex in U. In particular, $z(e) = w(e)$. Furthermore, e is not a valid flip. Thus, $w(\delta^-(v)) + w(e) > 1.75$. Also note that $|\delta_F^-(v)| \leq 1$, since v is good. If $w(e) \leq 0.5$, then

$$y(v) \geq w(\delta^-(v)) - \frac{1}{4}|\delta_F^-(v)| > 1.75 - w(e) - \frac{1}{4} \geq 1 \geq z(C).$$

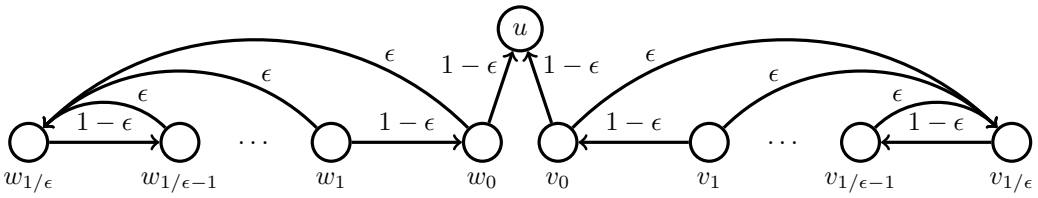
If $w(e) = 1$, then $e \in \delta_F^+(v)$. Hence,

$$y(v) \geq \begin{cases} w(\delta^-(v)) + \frac{1}{4}|\delta_F^+(v)| - \frac{1}{4}|\delta_F^-(v)| \geq 1.75 - w(e) + \frac{1}{4} \geq 1 \geq z(C) & \text{if } |\delta_F^-(v)| = 0, \\ w(\delta_F^-(v)) + \frac{1}{4}|\delta_F^+(v)| - \frac{1}{4}|\delta_F^-(v)| \geq 1 + \frac{1}{4} - \frac{1}{4} \geq 1 \geq z(C) & \text{if } |\delta_F^-(v)| = 1. \end{cases}$$

The second condition of Lemma 2 is that $z(E) > y(V)$. This holds because

$$z(E) = \sum_{v \in U} w(\delta^-(v)) = \sum_{v \in U} w(\delta^-(v)) + \frac{1}{4} \sum_{v \in V} [|\delta_F^+(v)| - |\delta_F^-(v)|] > y(V).$$

The strict inequality follows from the fact that there is at least one bad vertex. In the general case this algorithm does not get better than 2 as can be seen in the example below. In a similar, but more complicated way one can also show that in the special case with weights in $(0, 0.5] \cup \{1\}$, the algorithm does not succeed for $1.75 - \epsilon$, where $\epsilon > 0$ is arbitrary. In other words, the analysis for 1.75 is tight.



The LP and integral optima are again 1. Suppose the algorithm tries to find a solution with makespan $2 - 3\epsilon$. The only bad vertex is u. Hence, the algorithm will add the $(1 - \epsilon)$-edges to the pending flips one after another. However, at the time it reaches $v_{1/\epsilon}$ or $w_{1/\epsilon}$ it will get stuck, since there is a load of $1/\epsilon \times \epsilon$ on them. The way to fix this is to allow edges (in this case those of weight ϵ) to also be flipped towards vertices in U, i.e., vertices where we wanted to reduce the load. We cannot simply allow arbitrary flips back and forth between U,

because we have to take care that the algorithm eventually terminates. This is where the concept of vertices *repelling* edges comes in: Depending on the current orientation of the edges and the list of pending flips we will define a binary relation between vertices and edges. The exact definition has to be chosen carefully. For a pair (v, e) of this relation we write vertex v repels edge e. When a vertex repels an edge, it means it is undesirable that the edge is oriented towards this vertex. When it does not repel the edge, we do not care. This will be used in the algorithm when adding pending flips: An edge e may only be added to the pending flips, when it is repelled by its current vertex and not repelled by its prospect vertex.

In the earlier toy algorithms a vertex either repels all edges (when it is in U) or none (when it is not in U). By adding a fine-grained strategy of repelled edges, we gain much more flexibility. A strategy that appears particularly simple and powerful is the following. (1) We let bad vertices repel all edges. Moreover, (2) for every pending flip e with prospect vertex v we find maximum threshold W such that all edges in $\delta^-(v)$ with weight at least W are already enough to prevent the flip from being executed, i.e., their total weight is greater than $1 + R - w(e)$. We let v repel all edges of weight at least $\min\{w(e), W\}$.

Now the third toy algorithm is to repeat the following until all vertices are good. Find an edge e that is repelled by $t(e)$, but not by $s(e)$ and add it to the list of pending flips. As long as there is a valid pending flip, execute it and delete all pending flips added after it. This algorithm succeeds in the previous example. It will add the $(1-\epsilon)$-edges to the pending flips, but the vertices $u, v_1, \ldots, v_{1/\epsilon-1}, w_1, \ldots, w_{1/\epsilon-1}$ repel only the $(1-\epsilon)$-edges and not the ϵ-edges. Once the pending flips reach $w_{1/\epsilon}$ or $v_{1/\epsilon}$, these vertices will repel the ϵ-edges and start flipping them. Finally, there will be enough space on $w_{1/\epsilon}$ or $v_{1/\epsilon}$ and the $1-\epsilon$ edges can be flipped one after another.

For the simple case at least this is very close to the algorithm that gives us the bound of 1.74. However, to make the analysis work, we add a couple of tweaks. One of them is the idea of critical vertices. When the threshold of repelled edges, i.e., $\min\{w(e), W\}$, reaches a low value, e.g., 0.26, we make the vertex repel all edges. We call such a vertex a *critical* vertex. This tweak (together with some technicalities) allows us to argue that at least half of the critical vertices are prospect vertices of pending flips for tiny edges. This is helpful, since (unless the pending flip is valid) such prospect vertices have a lot of weight oriented towards them and this makes it easier to argue that the configuration LP also cannot distribute this weight very well; thereby reaching a contradiction.

An important novelty in this paper compared to earlier local search algorithms is that we are able to repel only a subset of edges small/tiny edges. In the Restricted Assignment proofs [19, 13] it is always the case that a machine (vertex) repels all small/tiny jobs (edges) or none. To utilize the flexibility in the small/tiny edges we scale the z values of tiny edges up by a factor $\beta > 1$. It is not obvious at all that this works out and the proof is quite tricky.

In a sense, we push the threshold for repelling all edges down to some value less than 0.5 (see description of critical vertices above). A logical conclusion to draw would be that pushing it down even more (or removing it completely) should give an even better algorithm. This would bring us back to toy algorithm 3. In fact, we are not aware of bad instances. It is an intriguing question whether this can also be proved that this simple algorithm is good.

3.2 Algorithm

The central data structure we use is an ordered list of pending flips $P = (e_1^P, e_2^P, \ldots, e_\ell^P)$. Here, every component e_k^P, stands for an edge the algorithm wants to flip. If P is clear from the context, we simply write *flip* when we speak of a pending flip e_k^P. A *tiny flip* is a flip where e_k^P is tiny. In the same way we define *small* and *big flips*. The prospect vertex of a

flip e_k^P is the vertex $s(e_k^P)$ to which we want to orient it. The algorithm will not perform the flip, if this would create a bad vertex, i.e., $w(\delta^-(s(e_k^P))) + w(e_k^P) > 1 + R$. If it does not create a bad vertex, we say that the flip e_k^P is a *valid flip*. For every $0 \leq k \leq \ell$ define $P_{\leq k} := (e_1^P, \ldots, e_k^P)$, i.e., the first k elements of P (with $P_{\leq 0}$ being the empty list).

At each point during the execution of the algorithm, the vertices repel certain edges. This can be thought of as a binary relation between vertices and their incident edges, i.e., a subset of $\{(v, e) : v \in r(e)\}$, and this relation changes dynamically as the current orientation or P change. The definition of which vertices repel which edges is given in Section 3.3. The algorithm will only add a new pending flip e to P, if e is repelled by the vertex it is oriented towards and not repelled by the other.

Algorithm 1: Local search algorithm for simplified Graph Balancing.

Input : Weighted multigraph $G = (V, E, r, w)$ with $\mathrm{OPT}^* = 1$ and $w(e) \in (0, 0.5] \cup \{1\}$ for all $e \in E$
Result: Orientation $s, t : E \to V$ with maximum weighted in-degree $1 + R$
let $s, t : E \to V$ map arbitrary source and target vertices to each edge ;
// i.e., $\{s(e), t(e)\} = r(e)$ for all $e \in E$
$\ell \leftarrow 0$; // number of pending edges P to flip
while *there is a bad vertex* **do**
  if *there exists a valid flip* $e \in P$ **then**
    let $0 \leq k \leq \ell$ be minimal such that e is repelled by $t(e)$ w.r.t. $P_{\leq k}$;
    exchange $s(e)$ and $t(e)$;
    $P \leftarrow P_{\leq k}$; $\ell \leftarrow k$; // Forget pending flips $e_{k+1}^P, \ldots, e_\ell^P$
  else
    choose an edge $e \in E \setminus P$ with $w(e)$ minimal and e is repelled by $t(e)$ and not repelled by $s(e)$ w.r.t. P ;
    $P_{\ell+1} \leftarrow e$; $\ell = \ell + 1$; // Append e to P
  end
end

3.3 Repelled edges

Consider the current list of ℓ pending flips $P_{\leq \ell}$. The repelled edges are defined inductively. For some $k \leq \ell$ we will now define the repelled edges w.r.t. $P_{\leq k}$.

(initialization) If $k = 0$, let every bad vertex v repel every edge in $\delta(v)$. Furthermore, let every vertex v repel every loop e where $\{v\} = r(e)$.

(monotonicity) If $k > 0$ and v repels e w.r.t. $P_{\leq k-1}$, then let v repel e w.r.t. $P_{\leq k}$.

The rule on loops is only for a technical reasons. Loops will never appear in the list of pending flips. The remaining rules regard $k > 0$ and the last pending flip in $P_{\leq k}$, e_k^P. The algorithm should reduce the load on $s(e_k^P)$ to make it valid.

Which edges exactly does the prospect vertex of e_k^P, i.e., $s(e_k^P)$, repel? First, we define $\tilde{E}(P_{\leq k-1}) \subseteq E$ where $e \in \tilde{E}(P_{\leq k-1})$ if and only if e is repelled by $s(e)$ w.r.t. $P_{\leq k-1}$. We will omit $P_{\leq k-1}$ when it is clear from the context. $\tilde{E}$ are edges that we do not expect to be able to flip: Recall that when an edge e is repelled by $s(e)$, it cannot be added to P. Moreover, for every W define $E_{\geq W} = \{e \in E : w(e) \geq W\}$. We are interested in values W such that

$$w(\delta^-_{\tilde{E} \cup E_{\geq W}}(s(e_k^P))) + w(e_k^P) > 1 + R. \tag{1}$$

Let $W_0 \in (0, w(e_k^P)]$ be maximal such that (1) holds. To be well-defined, we set $W_0 = 0$, if no such W exists. In that case, however, it holds that $w(\delta^-(s(e_k^P))) + w(e_k^P) \leq 1 + R$. This means that e_k^P is valid and the algorithm will remove it from the list immediately. Hence, the case is not particularly interesting. We define the following edges to be repelled by $s(e_k^P)$:
(uncritical) If $W_0 > 1 - R$, let $s(e_k^P)$ repel every edge in $\delta_{\tilde{E} \cup E_{\geq W_0}}(s(e_k^P))$.
(critical) If $W_0 \leq 1 - R$, let $s(e_k^P)$ repel every edge in $\delta(s(e_k^P))$.
Note that in the cases above there is not a restriction to incoming edges like in (1).

▶ **Fact 1.** *$s(e_k^P)$ repels e_k^P w.r.t. $P_{\leq k}$.*

This is because of $W_0 \leq w(e_k^P)$ in the rules for $P_{\leq k}$. An important observation is that repelled edges are stable under the following operation.

▶ **Fact 2.** *Let $e \notin P_{\leq k}$ be an edge that is not repelled by any vertex w.r.t. $P_{\leq k-1}$, and possibly by $t(e)$ (but not $s(e)$) w.r.t. $P_{\leq k}$. If the orientation of e changes and this does not affect the sets of good and bad vertices, the edges repelled by some vertex w.r.t. $P_{\leq k}$ will still be repelled after the change.*

Proof. We first argue that the repelled edges w.r.t. $P_{\leq k'}$, $k' = 0, \ldots, k-1$ have not changed. This argument is by induction. Since the good and bad vertices do not change, the edges repelled w.r.t. $P_{\leq 0}$ do not change. Let $k' \in \{1, \ldots, k-1\}$ and assume that the edges repelled w.r.t. $k'-1$ have not changed. Moreover, let W_0 be as in the definition of repelled edges w.r.t. $P_{\leq k'}$ before the change. We have to understand that e is not and was not in $\delta^-_{\tilde{E} \cup E_{\geq W_0}}(s(e_{k'}^P))$ (with $\tilde{E} = \tilde{E}(P_{\leq k'-1})$). This means that flipping it does not affect the choice of W_0 and, in particular, not the repelled edges. Let v and v' denote the vertex e is oriented towards before the flip and after the flip, respectively. Since e was not repelled by v and v' w.r.t. $P_{\leq k'}$, it holds that $v, v' \neq s(e_k^P)$ or $w(e) < W_0$: $s(e_k^P)$ repelled all edges greater or equal W_0 in both the case (uncritial) and (critical), but e was not repelled by v or v'.

Therefore $e \notin \delta^-_{E_{\geq W_0}}(s(e_{k'}^P))$ before and after the change. Moreover, since e was not repelled by any vertex w.r.t. $P_{\leq k'-1}$ (and by induction hypothesis, it still is not), it follows that $e \notin \tilde{E}(P_{\leq k'-1})$. By this induction we have that edges repelled w.r.t. $P_{\leq k-1}$ have not changed and by the same argument as before, e is not in $\delta^-_{\tilde{E} \cup E_{\geq W_0}}(s(e_k^P))$ after the change. This means W_0 has not increased and edges repelled w.r.t. $P_{\leq k}$ are still repelled. It could be that W_0 decreases, if e was in $\delta^-_{\tilde{E} \cup E_{\geq W_0}}(s(e_k^P))$ before the flip. This would mean that the number of edges repelled by $s(e_k^P)$ increases. ◀

We note that W_0 (in the definition of $P_{\leq k}$) is either equal to $w(e_k^P)$ or it is the maximal value for which (1) holds. Furthermore,

▶ **Fact 3.** *Let W_0 be as in the definition of repelled edges w.r.t. $P_{\leq k}$. If $W_0 < w(e_k^P)$, then there is an edge of weight exactly W_0 in $\delta^-(s(e_k^P))$. Furthermore, it is not a loop and it is not repelled by its other vertex, i.e., not $s(e_k^P)$, w.r.t. $P_{\leq k}$.*

Proof. We prove this for $P_{\leq k-1}$. Since the change from $P_{\leq k-1}$ to $P_{\leq k}$ only affects $s(e_k^P)$, this suffices. All edges that are repelled by their other vertex w.r.t. $P_{\leq k-1}$ (in particular, loops) are in $\tilde{E}$. Recall that $w(\delta^-_{\tilde{E} \cup E_{\geq W_0}}(s(e_k^P))) + w(e_k^P) > 1 + R$. Assume toward contradiction there is no edge of weight W_0 in $\delta^-(s(e_k^P))$, which is not in $\tilde{E}$. This means there is some $\epsilon > 0$ such that $\delta^-_{\tilde{E} \cup E_{\geq W_0 + \epsilon}}(s(e_k^P)) = \delta^-_{\tilde{E} \cup E_{\geq W_0}}(s(e_k^P))$. Hence, W_0 is not maximal. ◀

3.4 Analysis

The following analysis holds for the values $R = 0.74$ and $\beta = 1.1$. β is a central parameter in the proof. These can be slightly improved, but we refrain from this for the sake of simplicity. The proof consists of two parts. We need to show that the algorithm terminates and that there is always either a valid flip in P or some edge can be added to P.

▶ **Lemma 5.** *The algorithm terminates after finitely many iterations of the main loop.*

Proof. We consider the potential function $s(P) = (g, |\tilde{E}(P_{\leq 0})|, \ldots, |\tilde{E}(P_{\leq \ell})|, -1)$, where g is the number of good vertices. We will argue that this vector increases lexicographically after every iteration of the main loop. Intuitively $|\tilde{E}(P_{\leq k})|$, $k \leq \ell$, is an measure for progress. When an edge e is repelled by a vertex v, then we want that $s(e) = v$. $|\tilde{E}(P_{\leq k})|$ counts exactly these situations. Since ℓ is bounded by $|E|$, there can be at most $|V| \cdot |E|^{O(|E|)}$ possible values for the vector. Thus, the algorithm must terminate after at most this many iterations. In an iteration either a new flip is added to P or a flip is executed.

If a flip e is added as the $(\ell+1)$-th element of P, then clearly $\tilde{E}(P_{\leq i})$ does not change for $i \leq \ell$. Furthermore, the last component of the vector is replaced by some non-negative value.

Now consider a flip $e \in P$ that is executed. If this flip turns a bad vertex good, we are done. Hence, assume otherwise and let v and v' be the vertex it was previously and the one it is now oriented towards. Furthermore, let ℓ' be the length of P after the flip. Recall that ℓ' was chosen such that before the flip is executed v repels e w.r.t. $P_{\leq \ell'}$, but not w.r.t. $P_{\leq k}$ for any $k \leq \ell' - 1$. Also, e is not repelled by v' w.r.t. $P_{\leq k}$ for any $k \leq \ell'$ or else the flip would not have been added to P in the first place. By Fact 2 this means that repelled edges w.r.t. $P_{\leq k}$, $k \leq \ell'$, are still repelled after the flip. Because of this and because the only edge that changed direction, e, is not in $\tilde{E}(P_{\leq k})$ for any $k \leq \ell'$, $|\tilde{E}(P_{\leq k})|$ has not decreased. Finally, e has not been in $\tilde{E}(P_{\leq \ell'})$ before the flip, but now is. Thus, the first $\ell' - 1$ components of the vector have not decreased and the ℓ'-th one has increased. ◀

▶ **Lemma 6.** *If there at least one bad vertex remaining, then there is either a valid flip in P or a flip that can be added to P.*

Proof. We assume toward contradiction that there exists a bad vertex, no valid pending flip and no edge that can be added to P. We will show that this implies $\mathrm{OPT}^* > 1$.

Like above we denote by $\tilde{E} = \tilde{E}(P)$ those edges e that are repelled by $s(e)$. In particular, if an edge e is in P or repelled by $t(e)$ it must also be in $\tilde{E}$: If such an edge is in P, this follows from Fact 1. Otherwise, such an edge must be repelled also by $s(e)$ or else it could be added to P. For every $e \in E \setminus \tilde{E}$, set $z(e) = 0$. For every $e \in \tilde{E}$, set

$$z(e) = \begin{cases} 1 & \text{if } w(e) = 1, \\ w(e) & \text{if } 1 - R < w(e) \leq 1/2, \text{ and} \\ \beta w(e) & \text{if } w(e) \leq 1 - R. \end{cases}$$

In general, we would like to set each $y(v)$ to $z(\delta^-(v))$ (or equivalently, $z(\delta^-_{\tilde{E}}(v))$). However, there are two kinds of amortization between vertices that we include in the values of y.

As can be seen in the definition of repelled edges, a vertex v repels either edges with weight at least a certain threshold $W > 1 - R$ and edges in $\delta^-(v)$ that are repelled by their other vertex; or they repel all edges. We will call vertices of the latter kind *critical*. In other words, a vertex v is critical, if in the inductive definition of repelled edges at some point $v = s(e_k^P)$ and the rule (critical) applies. There might be a coincidence where only rule (uncritical) applies for v, but this already covers all edges in $\delta^-(v)$. This is not a critical vertex. We note that every vertex that is prospect vertex of a tiny flip e_k^P must be critical: In the inductive definition when considering e_k^P we have that $W_0 \leq w(e_k^P) \leq 1 - R$ by definition.

Critical vertex amortization. If v is a good vertex and critical, but is not a prospect vertex of a tiny flip, set $a_v := \beta - 1$. If v is a good vertex and prospect vertex of tiny flip (in particular, v is critical), set $a_v := -(\beta - 1)$. Otherwise, set $a_v = 0$.

Big edge amortization. Let $F \subseteq P$ denote the set of big flips. Then in particular $F \subseteq \tilde{E}$ (Fact 1). We define for all $v \in V$, $b_v := (|\delta_F^+(v)| - |\delta_F^-(v)|) \cdot (1 - R)$.

We conclude the definition of y by setting $y(v) = z(\delta^-(v)) + a_v + b_v$ for all good vertices v and $y(v) = z(\delta^-(v)) + a_v + b_v - \mu$ for all bad vertices v, where $\mu = 0.01$.

▷ Claim 7. It holds that $\sum_{v \in V} y(v) < \sum_{e \in E} z(e)$.

▷ Claim 8. $y(v), z(e) \geq 0$ f.a. $v \in V, e \in E$ and f.a. $v \in V$, $C \in \mathcal{C}(v, 1)$, $\sum_{e \in C} z(e) \leq y(v)$.

By Lemma 2 this implies that $\mathrm{OPT}^* > 1$. ◀

Proof of Claim 7. First we note that

$$\sum_{v \in V} b_v = (1 - R) \underbrace{\left(\sum_{v \in V} |\delta_F^+(v)| - \sum_{v \in V} |\delta_F^-(v)|\right)}_{=0} = 0.$$

Moreover, we have that $\sum_{v \in V} a_v \leq 0$: This is because at least half of all good vertices that are critical are prospect vertices of tiny flips. The proof for this is omitted due to space constraints. We conclude,

$$\sum_{e \in E} z(e) \geq \sum_{v \in V} \sum_{e \in \delta^-(v)} z(e) + \sum_{v \in V} [b_v + a_v] > \sum_{v \in V} y(v),$$

where the strict inequality holds because there exists at least one bad vertex. ◁

Proof of Claim 8. Let $v \in V$ and $C \in \mathcal{C}(v, 1)$. We need to show that $z(C) \leq y(v)$. Obviously the z values are non-negative. By showing the inequality above, we also get that the y values are non-negative. First, we will state some auxiliary facts.

▶ **Fact 4.** *For every edge flip $e \in P$, we have $w(\delta_{\tilde{E}}^-(s(e))) + w(e) > 1 + R$.*

This is due to the fact that e is not a valid flip and by definition of repelled edges.

▶ **Fact 5.** *If v is a good vertex and not prospect vertex of a tiny pending flip, then $y(v) \geq z(C) + w(\delta_{\tilde{E}}^-(v)) - 1 + b_v$. In other words, it is sufficient to show that $w(\delta_{\tilde{E}}^-(v)) + b_v \geq 1$.*

If v is critical, then $y(v) = z(\delta^-(v)) + \beta - 1 + b_v \geq w(\delta_{\tilde{E}}^-(v)) + z(C) - 1 + b_v$. If v is uncritical, all edges $e \in C$ with $z(e) > w(e)$ must be in $\delta^-(v)$. Otherwise, the flip e could be added to P. Therefore,

$$\begin{aligned} y(v) = z(\delta^-(v)) + a_v + b_v &\geq z(\delta_{\tilde{E}}^-(v)) - w(\delta_{\tilde{E}}^-(v)) + w(\delta_{\tilde{E}}^-(v)) + b_v \\ &\geq z(\tilde{C}) - w(\tilde{C}) + w(\delta_{\tilde{E}}^-(v)) + b_v \geq z(C) + w(\delta_{\tilde{E}}^-(v)) - 1 + b_v. \end{aligned}$$

▶ **Fact 6.** *For every vertex v it holds that (1) v repels no edges, (2) v repels all edges (critical), or (3) there exists a threshold $W > 1 - R$ such that v repels edges $e \in \delta^-(v)$ if $w(e) \geq W$ or they are also repelled by $s(e)$.*

Furthermore, in (3) we have that $W = \min_{e \in \delta_P^+(v)} w(e)$ or $W < \min_{e \in \delta_P^+(v)} w(e)$ and $W \in \{w(e) : e \in \delta_{\tilde{E}}^-(v)\}$ (see Fact 3).

Case 1: v is a bad vertex. If $|\delta_F^-(v)| \geq 2$,

$$y(v) \geq z(\delta^-(v)) - |\delta_F^-(v)| \cdot (1-R) - \mu \geq |\delta_F^-(v)| \cdot (1-(1-R)) - \mu \geq 2R - \mu \geq \beta \geq z(C).$$

Otherwise, $|\delta_F^-(v)| \leq 1$ and therefore

$$y(v) \geq z(\delta^-(v)) - 1 + R - \mu \geq w(\delta^-(v)) - 1 + R - \mu > 1 + R - 1 + R - \mu = 2R - \mu \geq \beta \geq z(C).$$

Here we use that $w(\delta^-(v)) > 1 + R$ by the definition of a bad vertex. Assume for the remainder that v is a good vertex, in particular that $|\delta_F^-(v)| \leq 1$.

Case 2: v is good and prospect vertex of a tiny flip. Using Fact 4 we get $w(\delta_{\tilde{E}}^-(v)) > 1 + R - (1-R) = 2R$. Since $z(e) \geq w(e)$ for all $e \in \tilde{E}$, we also have $z(\delta^-(v)) \geq 2R$. Moreover, since the vertex is good, $|\delta_F^-(v)| \leq 1$ and consequently $b_v \geq -(1-R)$. We conclude

$$y(v) \geq z(\delta^-(v)) - (\beta - 1) - (1-R) \geq 2R - (\beta - 1) - (1-R) = \beta + \underbrace{3R - 2\beta}_{\geq 0} \geq z(C).$$

Case 3: v is good, not prospect vertex of a tiny flip, but prospect vertex of a small flip. If $|\delta_F^-(v)| = 0$, again with Fact 4 we get $w(\delta_{\tilde{E}}^-(v)) + b_v \geq 1 + R - 0.5 + 0 > 1$. This suffices because of Fact 5. Moreover, if $|\delta_F^-(v)| = 1$ and $\delta_{\tilde{E}}^-(v)$ contains a small edge, it holds that

$$w(\delta_{\tilde{E}}^-(v)) + b_v \geq w(\delta_F^-(v)) + (1-R) + b_v \geq 1 + (1-R) - (1-R) = 1.$$

Here we use that the small edge must have a size of at least $1 - R$. The case that remains is where $\delta_{\tilde{E}}^-(v)$ contains one edge from F and tiny edges. Since the overal weight of $\delta_{\tilde{E}}^-(v)$ is at least $1 + R - 0.5$, the weight of tiny edges is at least $R - 0.5$. Thus, the z-value of the tiny edges is at least $\beta(R - 0.5)$. If v is critical,

$$y(v) \geq z(\delta^-(v)) - (1-R) + (\beta - 1) \geq 1 + \underbrace{\beta(R - 0.5) - (1-R)}_{\geq 0} + (\beta - 1) \geq z(C).$$

Notice that $\beta(R - 0.5) \geq 1 - R$ by choice of R and β. Assume for the remainder of this case that v is uncritical. If C contains a big edge, this must be the only element in C. Therefore,

$$y(v) \geq z(\delta^-(v)) - (1-R) \geq 1 + \beta(R - 0.5) - (1-R) \geq 1 = z(C).$$

Consider the case where $C \cap \tilde{E}$ contains at most one small edge and no big edge. Since v is uncritical, all tiny edges in C with positive z value must also be in $\delta^-(v)$. Therefore, $z(C) \leq 0.5 + z(\delta_T^-(v))$. Thus,

$$y(v) \geq z(\delta^-(v)) - (1-R) \geq 1 + z(\delta_T^-(v)) - (1-R) > 0.5 + z(\delta_T^-(v)) \geq z(C).$$

In the final case $C \cap \tilde{E}$ contains $k \in \{2, 3\}$ small edges. We let s denote the weight of the smallest edge with a pending flip whose prospect vertex is v. Since v is not critical and there is no small edge in $\delta^-(v)$, v repels exactly those edges with weight at least s. This means the small edges in $C \cap \tilde{E}$ must be of weight at least s: Otherwise, they could be added as pending flips. Therefore,

$$\begin{aligned} y(v) &\geq z(\delta^-(v)) - (1-R) \geq z(\delta_F^-(v)) + z(\delta_T^-(v)) - (1-R) \\ &\geq 1 + \beta(w(\delta_{\tilde{E}}^-(v)) - w(\delta_F^-(v))) - (1-R) = R + \beta(w(\delta_{\tilde{E}}^-(v)) - 1) \geq R + \beta(R - s). \end{aligned}$$

Note that $s \leq 1/k$ and, by choice of β, $k - \beta(k-1) \geq 0$. It follows that

$$\begin{aligned} z(C) &\leq k \cdot s + \beta(1 - k \cdot s) \\ &\leq y(v) + k \cdot s - R + \beta(1 - R - (k-1)s) \leq y(v) + k \cdot \frac{1}{k} - R + \beta\left(1 - R - \frac{k-1}{k}\right) \\ &= y(v) + 1 - R + \beta\left(\frac{1}{k} - R\right) \leq y(v) + 1 - R + \beta(0.5 - R) \leq y(v). \end{aligned}$$

Case 4: v is good and not prospect vertex of a small/tiny flip. If $|\delta_F^+(v)| = 0$, then v does not repel any edges. In particular, $\delta_F^-(v) = \emptyset$ and every $e \in \delta(v)$ with $z(e) > 0$ must be in $\delta^-(v)$. Therefore, $z(C) \leq z(\delta^-(v)) \leq y(v)$. We assume in the remainder that $|\delta_F^+(v)| = 1$.
If $|\delta_F^-(v)| = 1$, we get $w(\delta_{\tilde{E}}^-(v)) + b_v \geq 1 + 0$. Otherwise, it must hold that $|\delta_F^-(v)| = 0$ and $w(\delta_{\tilde{E}}^-(v)) + b_v \geq R + (1 - R) = 1$. ◁

References

1. Chidambaram Annamalai. Lazy Local Search Meets Machine Scheduling. *CoRR*, abs/1611.07371, 2016. arXiv:1611.07371.
2. Chidambaram Annamalai, Christos Kalaitzis, and Ola Svensson. Combinatorial Algorithm for Restricted Max-Min Fair Allocation. *ACM Transactions on Algorithms*, 13(3):37:1–37:28, 2017. Previously appeared in SODA'15. doi:10.1145/3070694.
3. Arash Asadpour, Uriel Feige, and Amin Saberi. Santa Claus meets hypergraph matchings. *ACM Transactions on Algorithms*, 8(3):24:1–24:9, 2012. See Asadpour's homepage for the bound of 4 instead of 5 as in the paper; Previously appeared in APPROX'08. doi:10.1145/2229163.2229168.
4. Yuichi Asahiro, Jesper Jansson, Eiji Miyano, Hirotaka Ono, and Kouhei Zenmyo. Approximation algorithms for the graph orientation minimizing the maximum weighted outdegree. *Journal of Combinatorial Optimization*, 22(1):78–96, 2011. doi:10.1007/s10878-009-9276-z.
5. Nikhil Bansal and Maxim Sviridenko. The Santa Claus problem. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 31–40, 2006. Previously appeared in STOC'06. doi:10.1145/1132516.1132522.
6. Deeparnab Chakrabarty, Julia Chuzhoy, and Sanjeev Khanna. On Allocating Goods to Maximize Fairness. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 107–116, 2009. doi:10.1109/FOCS.2009.51.
7. Deeparnab Chakrabarty and Kirankumar Shiragur. Graph Balancing with Two Edge Types. *CoRR*, abs/1604.06918, 2016. arXiv:1604.06918.
8. Siu-Wing Cheng and Yuchen Mao. Integrality Gap of the Configuration LP for the Restricted Max-Min Fair Allocation. *CoRR*, abs/1807.04152, 2018. arXiv:1807.04152.
9. Tomáš Ebenlendr, Marek Krčál, and Jiří Sgall. Graph Balancing: A Special Case of Scheduling Unrelated Parallel Machines. *Algorithmica*, 68(1):62–80, 2014. Previously appeared in SODA'08. doi:10.1007/s00453-012-9668-9.
10. Chien-Chung Huang and Sebastian Ott. A Combinatorial Approximation Algorithm for Graph Balancing with Light Hyper Edges. In *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, pages 49:1–49:15, 2016. doi:10.4230/LIPIcs.ESA.2016.49.
11. Klaus Jansen, Kati Land, and Marten Maack. Estimating The Makespan of The Two-Valued Restricted Assignment Problem. In *15th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2016, June 22-24, 2016, Reykjavik, Iceland*, pages 24:1–24:13, 2016. doi:10.4230/LIPIcs.SWAT.2016.24.

12 Klaus Jansen and Lars Rohwedder. A Quasi-Polynomial Approximation for the Restricted Assignment Problem. In *Integer Programming and Combinatorial Optimization - 19th International Conference, IPCO 2017, Waterloo, ON, Canada, June 26-28, 2017, Proceedings*, pages 305–316, 2017. doi:10.1007/978-3-319-59250-3_25.

13 Klaus Jansen and Lars Rohwedder. On the Configuration-LP of the Restricted Assignment Problem. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2670–2678, 2017. doi:10.1137/1.9781611974782.176.

14 Klaus Jansen and Lars Rohwedder. A note on the integrality gap of the configuration LP for restricted Santa Claus. *CoRR*, abs/1807.03626, 2018. arXiv:1807.03626.

15 Klaus Jansen and Lars Rohwedder. Compact LP Relaxations for Allocation Problems. In *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA*, pages 11:1–11:19, 2018. doi:10.4230/OASIcs.SOSA.2018.11.

16 Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation Algorithms for Scheduling Unrelated Parallel Machines. *Math. Program.*, 46:259–271, 1990. doi:10.1007/BF01585745.

17 Daniel R. Page and Roberto Solis-Oba. A 3/2-Approximation Algorithm for the Graph Balancing Problem with Two Weights. *Algorithms*, 9(2):38, 2016. doi:10.3390/a9020038.

18 Lukás Polácek and Ola Svensson. Quasi-Polynomial Local Search for Restricted Max-Min Fair Allocation. *ACM Transactions on Algorithms*, 12(2):13:1–13:13, 2016. Previously appeared in ICALP'12. doi:10.1145/2818695.

19 Ola Svensson. Santa Claus schedules jobs on unrelated machines. *SIAM Journal on Computing*, 41(5):1318–1341, 2012. Previously appeared in STOC'11. doi:10.1137/110851201.

20 José Verschae and Andreas Wiese. On the configuration-LP for scheduling on unrelated machines. *Journal of Scheduling*, 17(4):371–383, 2014. Previously appeared in ESA'11. doi:10.1007/s10951-013-0359-4.

21 Chao Wang and René Sitters. On some special cases of the restricted assignment problem. *Information Processing Letters*, 116(11):723–728, 2016. doi:10.1016/j.ipl.2016.06.007.

Near-Linear Time Algorithm for n-fold ILPs via Color Coding

Klaus Jansen
Department of Computer Science, Kiel University, Kiel, Germany
kj@informatik.uni-kiel.de

Alexandra Lassota
Department of Computer Science, Kiel University, Kiel, Germany
ala@informatik.uni-kiel.de

Lars Rohwedder
Department of Computer Science, Kiel University, Kiel, Germany
lro@informatik.uni-kiel.de

Abstract

We study an important case of ILPs $\max\{c^T x \mid \mathcal{A}x = b, l \leq x \leq u,\ x \in \mathbb{Z}^{nt}\}$ with $n \cdot t$ variables and lower and upper bounds $\ell, u \in \mathbb{Z}^{nt}$. In n-fold ILPs non-zero entries only appear in the first r rows of the matrix $\mathcal{A}$ and in small blocks of size $s \times t$ along the diagonal underneath. Despite this restriction many optimization problems can be expressed in this form. It is known that n-fold ILPs can be solved in FPT time regarding the parameters s, r, and Δ, where Δ is the greatest absolute value of an entry in $\mathcal{A}$. The state-of-the-art technique is a local search algorithm that subsequently moves in an improving direction. Both, the number of iterations and the search for such an improving direction take time $\Omega(n)$, leading to a quadratic running time in n. We introduce a technique based on Color Coding, which allows us to compute these improving directions in logarithmic time after a single initialization step. This leads to the first algorithm for n-fold ILPs with a running time that is near-linear in the number nt of variables, namely $(rs\Delta)^{\mathcal{O}(r^2 s + s^2)} L^2 \cdot nt \log^{\mathcal{O}(1)}(nt)$, where L is the encoding length of the largest integer in the input. In contrast to the algorithms in recent literature, we do not need to solve the LP relaxation in order to handle unbounded variables. Instead, we give a structural lemma to introduce appropriate bounds. If, on the other hand, we are given such an LP solution, the running time can be decreased by a factor of L.

2012 ACM Subject Classification Mathematics of computing → Integer programming

Keywords and phrases Near-Linear Time Algorithm, n-fold ILP, Color Coding

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.75

Category Track A: Algorithms, Complexity and Games

Related Version A full version can be found at `https://arxiv.org/abs/1811.00950`.

Funding This work was supported by DFG project JA 612/20-1.

1 Introduction

Solving integer linear programs of the form $\max\{c^T x \mid \mathcal{A}x = b, x \in \mathbb{Z}_{\geq 0}\}$ is one of the most fundamental tasks in optimization. This problem is very general and broadly applicable, but unfortunately also very hard. In this paper we consider n-fold ILPs, a class of integer linear programs with a specific block structure. This is, when non-zero entries appear only in the first r rows of $\mathcal{A}$ and in blocks of size $s \times t$ along the diagonal underneath. More precisely,

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 75; pp. 75:1–75:13

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

the contraint matrix in an n-fold ILP has the form

$$\mathcal{A} = \begin{pmatrix} A_1 & A_2 & \dots & A_n \\ B_1 & 0 & \dots & 0 \\ 0 & B_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & B_n \end{pmatrix},$$

where $A_1, \dots, A_n$ are $r \times t$ matrices and $B_1, \dots, B_n$ are $s \times t$ matrices. In n-fold ILPs we also allow upper and lower bounds on the variables. Throughout the paper we subdivide a solution x into bricks of length t and denote by $x^{(i)}$ the i-th one. The corresponding columns in $\mathcal{A}$ will be called blocks.

Lately, n-fold ILPs received great attention [2, 7, 12, 14, 16] and were studied intensively due to two reasons. Firstly, many optimization problems are expressible as n-fold ILPs [5, 10, 12, 14]. Secondly, n-fold ILPs indeed can be solved much more efficiently than arbitrary ILPs [7, 10, 16]. The previously best algorithm has a running time of $(rs\Delta)^{\mathcal{O}(r^2s+rs^2)}L \cdot (nt)^2 \log^2(n \cdot t) + \text{LP}$ and is due to Eisenbrand et al. [7]. Here LP is the running time required for solving the corresponding LP relaxation. This augmentation algorithm is the last one in a line of research, where local improvement/augmenting steps are used to converge to an optimal solution. Clever insights about the structure of the improving directions allow them to be computed fast. Nevertheless, the dependence on n in the algorithm above is still high. Indeed, in practice a quadratic running time is simply not suitable for large data sets [3, 6, 13]. For example when analyzing big data, large real world graphs as in telecommunication networks or DNA strings in biology, the duration of the computation would go far beyond the scope of an acceptable running time [3, 6, 13]. For this reason even problems which have an algorithm of quadratic running time are still studied from the viewpoint of approximation algorithms with the objective to obtain results in subquadratic time, even at the cost of a worse quality [3, 6, 13]. Hence, it is an intriguing question, whether the quadratic dependency on the number nt of variables can be eliminated. In this paper, we answer this question affirmatively. The technical novelty comes from a surprising area: We use a combinatorial structure called splitter, which has been used to derandomize Color Coding algorithms. It allows us to build a powerful data structure that is maintained during the local search and from which we can derive an improving direction in logarithmic time. Handling unbounded variables in an n-fold ILP is a non-trivial issue in the previous algorithms from literature. They had to solve the corresponding LP relaxation and use proximity results. Unfortunately, it is not known whether linear programming can be solved in near-linear time in the number of variables. Hence, it is an obstacle for obtaining a near-linear running time. We manage to circumvent the necessity of solving the LP by introducing artificial bounds as a function of the finite upper bounds and the right-hand side of the n-fold ILP.

Summary of Results

- We present an algorithm, which solves n-fold ILPs in time $(rs\Delta)^{\mathcal{O}(r^2s+s^2)}L \cdot nt \log^5(nt) + \text{LP}$, where LP is the time to solve the LP relaxation of the n-fold ILP. This is the first algorithm with a near-linear dependence on the number of variables. The crucial step is to speed up the computation of the improving directions.
- We circumvent the need for solving the LP relaxation. This leads to a purely combinatorial algorithm with running time $(rs\Delta)^{\mathcal{O}(r^2s+s^2)}L^2 \cdot nt \log^7(nt)$.
- In the running times above the dependence on the parameters, i.e., $(rs\Delta)^{\mathcal{O}(r^2s+s^2)}$, improves on the function $(rs\Delta)^{\mathcal{O}(r^2s+rs^2)}$ in the previous best algorithms.

Outline of New Techniques

We will briefly elaborate the main technical novelty in this paper. Let x be some feasible, non-optimal solution for the n-fold ILP. It is clear that when y^* is an optimal solution for $\max\{c^T y \mid \mathcal{A}y = 0, \ell - x \leq y \leq u - x, y \in \mathbb{Z}^{nt}\}$, then $x + y^*$ is optimal for the initial n-fold ILP. In other words, y^* is a particularly good improving step. A sensible approximation of y^* is to consider directions y of small size and multiplying them by some step length, i.e., find some $\lambda \cdot y$ with $\|y\|_1 \leq k$ for a value k depending only on Δ, r, and s. This implies that at most k of the n blocks are used for y. If we randomly color the blocks with k^2 colors, then with high probability at most one block of every color is used. This reduces the problem to choosing a solution of a single brick for every color and to aggregate them. We add data structures for every color to implement this efficiently. There is of course a chance that the colors do not split y perfectly. We handle this by using a deterministic structure of multiple colorings (instead of one) such that it is guaranteed that at least one of them has the desired property.

Related Work

The first XP-time algorithm for solving n-fold integer programs is due to De Loera et al. [5] with a running time of $n^{g(\mathcal{A})}L$. Here $g(\mathcal{A})$ denotes a so-called Graver complexity of the constraint matrix $\mathcal{A}$ and L is the encoding length of the largest number in the input. This algorithm already uses the idea of iterative converging to the optimal solution by finding improving directions. Nevertheless, the Graver complexity appears to be huge even for small n-fold integer linear programs and thus this algorithm was of no practical use [10]. The exponent of this algorithm was then greatly improved by Hemmecke et al. [10] to a constant factor yielding the first cubic time algorithm for solving n-fold ILPs. More precisely, the running time of their algorithm is $\Delta^{\mathcal{O}(t(rs+st))}L \cdot (nt)^3$, i.e., FPT-time parameterized over Δ, r, s, and t. Lately, two more breakthroughs were obtained. One of the results is due to Koutecký et al. [16], who gave a strongly polynomial algorithm with running time $\Delta^{\mathcal{O}(r^2s+rs^2)}(nt)^6 \cdot \log(nt) + LP$. Here LP is the running time for solving the corresponding LP relaxation, which is possible in strongly polynomial time, since the entries of the matrix are bounded. Simultaneously, Eisenbrand et al. reduced in [7] the running time from a cubic factor to a quadratic one by introducing new proximity and sensitivity results. This leads to an algorithm with running time $(\Delta rs)^{\mathcal{O}(r^2s+rs^2)}L \cdot (nt)^2 \log^2(nt) + LP$. Note that both results require only polynomial dependency on t.

As for applications, n-fold ILPs are broadly used to model various problems such as string, flow or scheduling problems. We refer to the works [5, 10, 11, 12, 15, 19] and the references therein for an overview.

Structure of the Document

In Section 2 we introduce the necessary preliminaries. Section 3 gives the algorithm for efficiently computing the augmenting steps. This is then integrated into an algorithm for n-fold ILPs in Section 4. At first we require finite variable bounds and then discuss how to eliminate this requirement using the solution of the LP relaxation. Finally, in Section 5 we discuss how to handle infinite variable bounds without the LP relaxation and give new structural results.

2 Preliminaries

In the following we introduce n-fold ILPs formally and state the main results regarding them. Further we familiarize splitters, a technique known from Color Coding.

▶ **Definition 1.** *Let $n, r, s, t \in \mathbb{N}$. Furthermore let $A_1, \ldots, A_n$ be $r \times t$ integer matrices and $B_1, \ldots, B_n$ be $s \times t$ integer matrices. Then the n-fold matrix $\mathcal{A}$ is of following form:*

$$\mathcal{A} = \begin{pmatrix} A_1 & A_2 & \ldots & A_n \\ B_1 & 0 & \ldots & 0 \\ 0 & B_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & B_n \end{pmatrix}.$$

The matrix $\mathcal{A}$ is of dimension $(r+n\cdot s)\times n\cdot t$. We will divide $\mathcal{A}$ into blocks of size $(r+n\cdot s)\times t$. Similarly, the variables of a solution x are partitioned into bricks of length t. This means each brick $x^{(i)}$ corresponds to the columns of one submatrix A_i and therefore also B_i. Given $c, \ell, u \in \mathbb{Z}^{n\cdot t}$ and $b \in \mathbb{Z}^{r+n\cdot s}$, the corresponding n-fold Integer Linear Programming problem is defined by:

$$\max\{c^T x \mid \mathcal{A}x = b, \ell \leq x \leq u, x \in \mathbb{Z}^{n\cdot t}\}.$$

The main idea for the state-of-the-art algorithms relies on some insight about the Graver basis of n-fold ILPs, which are special elements of the kern of $\mathcal{A}$. More formally, we introduce the following definitions:

▶ **Definition 2.** *The kern of a matrix A is defined as the set of integral vectors $x \in \mathbb{Z}^{\aleph\cdot\approx}$ with $Ax = 0$. We write* $\ker(A)$ *for them.*

▶ **Definition 3.** *A Graver basis element g is a minimal element of* $\ker(A)$*. An element is minimal, if it is not the sum of two sign-compatible elements $u, v \in \ker(A)$.*

Here, sign-compatible means that $u_i \cdot v_i \geq 0$ for every i.

▶ **Theorem 4** ([4])**.** *Let $A \in \mathbb{Z}^{n\times m}$ and let $x \in \ker(A)$. Then there exist $2n-1$ Graver basis elements $g_1, \ldots, g_{2n-1}$, which are sign-compatible with x such that*

$$x = \sum_{i=1}^{2n-1} \lambda_i g_i$$

for some $\lambda_1, \ldots, \lambda_{2n-1} \in \mathbb{Z}_{\geq 0}$.

Many results for n-fold ILPs rely on the fact that the ℓ_1-norm of Graver basis elements for n-fold matrices are small. The best bound known for the ℓ_1-norm is due to Eisenbrand et al. [7].

▶ **Theorem 5** ([7])**.** *The ℓ_1-norm of the Graver basis elements of an n-fold matrix $\mathcal{A}$ is bounded by $\mathcal{O}(rs\Delta)^{rs}$.*

Next, we will introduce a technique called splitters (see e.g. [17]), which has its origins in the FPT community and was used to derandomize the Color Coding technique [1]. So far it has not been used with n-fold ILPs. We refer the reader to the outline of techniques in the introduction for the idea on how we apply the splitters.

▶ **Definition 6.** *An (n, k, ℓ) splitter is a family of hash functions F from $\{1, \ldots, n\}$ to $\{1, \ldots, \ell\}$ such that for every $S \subseteq \{1, \ldots, n\}$ with $|S| = k$, there exists a function $f \in F$ that splits S evenly, that is, for every $j, j' \leq \ell$ we have $|f^{-1}(j) \cap S|$ and $|f^{-1}(j') \cap S|$ differ by at most 1.*

If $\ell \geq k$, the above means that there is some hash function that has no collisions when restricted to S. Interestingly, there exist splitters of very small size.

▶ **Theorem 7** ([1, 18])**.** *There exists an (n, k, k^2) splitter of size $k^6 \log(k) \log(n)$ which is computable in time $k^6 \log(k) \cdot n \log(n)$.*

We note that an alternative approach to the result above is to use FKS hashing. Although it has an extra factor of $\log(n)$, it is particularly easy to implement.

▶ **Theorem 8** (Corollary 2 and Lemma 2, [9])**.** *Define for every prime $q < k^2 \log(n)$ and prime $p < q$ the hash function $x \mapsto 1 + (p \cdot (x \mod q) \mod k^2)$. This is an (n, k, k^2) splitter of size $\mathcal{O}(k^4 \log^2(n))$.*

We remark that a hash function from $\{1, \ldots, n\}$ to $\{1, \ldots, \ell\}$ naturally corresponds to a partition of the set $\{1, \ldots, n\}$ into exactly ℓ subsets.

3 Efficient Computation of Improving Directions

The backbone of our algorithm is the efficient computation of augmenting steps. The important aspect is the fact that we can update the augmenting steps very efficiently if the input changes only slightly. In other words, whenever we change the current solution by applying an augmenting step, we do not have to recompute the next augmenting step from scratch. The augmenting steps depend on a partition of the bricks. In the following we define the notion of a best step based on a fixed partition. Later, we will independently find steps for a number of partitions and take the best among them.

▶ **Definition 9.** *Let P be a partition of the n bricks into k^2 disjoint sets $P_1, P_2, \ldots, P_{k^2}$. Let $\overline{u} \in \mathbb{Z}^{nt}_{\geq 0}$ and $\overline{\ell} \in \mathbb{Z}^{nt}_{\leq 0}$ be some upper and lower bounds on the variables (not necessarily the same as in the n-fold ILP). A (P, k)-best step is an optimal solution of the system below. We slightly abuse notation by using P_j or bricks $S_j \in P_j$ for the indices of variables contained in them.*

$$
\begin{aligned}
\max\ & c^T x && \\
\mathcal{A}x &= 0 && \\
\sum_{i \in S_j} |x_i| &\leq k && \forall j \in \{1, \ldots, k^2\} \\
x_i &= 0 && \forall j \in \{1, \ldots, k^2\}, i \in P_j \setminus \{S_j\} \\
\overline{\ell} \leq x &\leq \overline{u} && \\
x &\in \mathbb{Z}^{nt} && \\
S_j &\in P_j && \forall j \in \{1, \ldots, k^2\}.
\end{aligned}
$$

This means a (P, k)-best step is an element of $\ker(\mathcal{A})$, which uses only one brick of every $P_j \in P$. Within that brick the norm of the solution must be at most k. Later, we will choose k as the upper bound for the ℓ_1-norm of a Graver basis element, i.e., $k = \mathcal{O}(rs\Delta)^{rs}$.

▶ **Theorem 10.** *Consider the problem of finding a (P,k)-best step in an n-fold matrix where the lower and upper bounds $\overline{u}, \overline{\ell}$ can change. This problem can be solved initially in time $k^{\mathcal{O}(r)} \cdot \Delta^{\mathcal{O}(r^2+s^2)} \cdot nt$ and then in $k^{\mathcal{O}(r)} \cdot \Delta^{\mathcal{O}(r^2+s^2)} \cdot \log(nt)$ update time whenever the bounds of a single variable change.*

Proof. Let P be a partition of the bricks from matrix $\mathcal{A}$ into k^2 disjoint sets $P_1, P_2, \ldots, P_{k^2}$. Solving the (P,k)-best step problem requires that from each set $P_j \in P$ we choose at most one brick and set this brick's variables. All variables in other bricks of P_j must be 0.

Let x be a (P,k)-best step and let $x^{(j)}$ have the values of x in variables of P_j and 0 in all other variables. Then by definition, $\|x^{(j)}\|_1 \leq k$. This implies that the right-hand side regarding $x^{(j)}$, that is to say, $\mathcal{A}x^{(j)}$, is also small. Since the absolute value of an entry in $\mathcal{A}$ is at most Δ, we have that $\|\mathcal{A}x^{(j)}\|_\infty \leq k\Delta$. Let a_i be the i-th row of $\mathcal{A}$. If $i > r$, then $a_i x^{(j)} = 0$. This is because $\mathcal{A}x = 0$ and a_i has all its support either completely inside P_j or completely outside P_j. Meaning, the value of $\mathcal{A}x^{(j)}$ is one of the $(2k\Delta+1)^r$ many values we get by enumerating all possibilities for the first r rows. Furthermore, since P has only k^2 sets, the partial sum $\mathcal{A}(x^{(1)} + \cdots + x^{(j)})$ is always one of $(2k^3\Delta+1)^r = (k\Delta)^{\mathcal{O}(r)}$ many candidates.

Hence to find a (P,k)-best step we can restrict our search to solutions whose partial sums stay in this range. To do so, we set up a graph containing k^2+2 layers $L_0, L_1, \ldots L_{k^2}, L_{k^2+1}$. An example is given in figure 1. The first layer L_0 will consist of just one node marking the starting point with partial sum zero. Similarly, the last layer L_{k^2+1} will just contain the target point also having partial sum zero, since a (P,k)-best step is an element of $\ker(\mathcal{A})$. Each layer L_j with $1 \leq j \leq k^2$ will contain $(2k^3\Delta+1)^r$ many nodes, each representing one possible value of $\mathcal{A}(x^{(1)} + \cdots x^{(j)})$. Two points v, w from adjacent layers L_{j-1}, L_j will be connected if the difference of the corresponding partial sums, namely $w - v$, can be obtained by a solution y of variables from only one brick of P_j (with $\|y\|_1 \leq k$). The weight of the edge will be the largest gain for the objective function $c^T y$ over all possible bricks. Hence, it could be necessary to compute and compare up to n values for each P_j and each difference in the partial sums to insert one edge into the graph. Finally, we just have to find the longest path in this graph as it corresponds to a (P,k)-best step. The out-degree of each node is bounded by $(2k^3\Delta+1)^r$ since at most this many nodes are reachable in the next layer. Therefore the overall number of edges is bounded by

$$(k^2+2) \cdot (2k^3\Delta+1)^r \cdot (2k^3\Delta+1)^r = (k\Delta)^{\mathcal{O}(r)}.$$

Using the Bellman-Ford algorithm we can solve the Longest Path problem for a graph with N vertices and M edges in time $\mathcal{O}(N \cdot M)$ as the graph is directed and acyclic. This gives a running time of $(k\Delta)^{\mathcal{O}(r)} \cdot (k\Delta)^{\mathcal{O}(r)} = (k\Delta)^{\mathcal{O}(r)}$ for solving the problem. Constructing the graph, however, requires solving a number of IPs of the form

$$\begin{aligned} \max\ & c'^T x \\ \begin{pmatrix} A_j \\ B_j \end{pmatrix} x &= \begin{pmatrix} b' \\ 0 \end{pmatrix} \\ \|x\|_1 &\leq k \\ \ell' \leq x &\leq u' \\ x &\in \mathbb{Z}^t, \end{aligned}$$

where $b' \in \mathbb{Z}^r$ is the corresponding right-hand side of the top rows and ℓ', u', c' are the upper and lower bounds, and the objective of the block. This is an IP with $r+s+1$ constraints, t variables, lower and upper bounds, and entries of the matrix bounded by Δ in absolute value.

Using the algorithm by Eisenbrand and Weismantel [8], solving one of them requires time

$$t \cdot \mathcal{O}(r+s+1)^{r+s+4} \cdot \mathcal{O}(\Delta)^{(r+s+1)(r+s+4)} \cdot \log^2((r+s+1)\Delta) + \mathrm{LP} = t \cdot \Delta^{\mathcal{O}(r^2+s^2)} + \mathrm{LP},$$

where LP is the time for solving the LP relaxation. Note that strictly speaking the constraint $\|x\|_1 \leq k$ is not linear, but by standard construction the ILP can easily be transformed into an equivalent one replacing every variable x_i with a composition $x_i^+ - x_i^-$ of two positive variables. Then the ℓ_1-norm is simply the sum over all variables. This does not affect the asymptotic running time.

Furthermore, a little thought allows us to reduce the dependency on t to a logarithmic one: Since the number of constraints in the ILP above is very small, there are only $\Delta^{\mathcal{O}(r+s)}$ many different columns. Because of the cardinality constraint $\|x\|_1 \leq k$, we only have to consider $2k$ many variables of each type of column, namely: The k many with $u'_i > 0$ and maximal c'_i and the k many with $\ell'_i < 0$ and minimal c'_i.

If some solution uses a variable not in this set, then by pigeonhole principle there is a variable with the same column values and a superior objective value and which can be increased/decreased. We can reduce the variable outside this set and increase the corresponding variable inside this set until all variables outside the set are 0. We can use an appropriate data structure (e.g. AVL trees) to maintain a set of all variables with $u'_i > 0$ ($\ell'_i < 0$) such that we can find the k best among them in time $\mathcal{O}(k \log(t))$. Whenever the bounds of some variable change, we might have to add or remove entries, which also takes only logarithmic time. After initialization in time $\mathcal{O}(nt)$ (in total for all bricks) solving such an IP can therefore be implemented in time

$$k\log(t) + 2k\Delta^{\mathcal{O}(r+s)}\Delta^{\mathcal{O}(r^2+s^2)} + \mathrm{LP} \leq k\log(t)\Delta^{\mathcal{O}(r^2+s^2)} + \mathrm{LP} \leq k\log(t)\Delta^{\mathcal{O}(r^2+s^2)}.$$

The last inequality holds, because using Tardos' algorithm [20] LPs can be solved in time polynomial in the encoding size of the matrix, which can be bounded by $2k\Delta^{\mathcal{O}(r+s)} \cdot (r+s) \cdot \log(\Delta)$. This is dominated by the other term. The number of IPs to solve is at most n times the number of edges, since we have to compare the values of up to n bricks. This gives a running time of

$$\mathcal{O}(nt) + n \cdot (k\Delta)^{\mathcal{O}(r)} \cdot \log(t) \cdot \Delta^{\mathcal{O}(r^2+s^2)} \leq nt \cdot k^{\mathcal{O}(r)} \cdot \Delta^{\mathcal{O}(r^2+s^2)}$$

for constructing the graph. To obtain the update time from the premise of the theorem, it is perfectly fine to solve the Longest Path problem again, but we cannot construct the graph from scratch. However, in order to construct the graph we still have to find the best value over all bricks for each edge. Fortunately, if only a few bricks are updated (in their lower and upper bounds) it is not necessary to recompute all values. Each edge corresponds to a particular $P_j \in P$ and a fixed right-hand side (a possible value of $\mathcal{A}x^{(j)}$). We require an appropriate data structure $\mathcal{D}_e$ for every edge e, which supports fast computation of the operations FindMax, Insert, and Delete. Again, an AVL tree computes each of these operations in time $\mathcal{O}(\log(N))$, where N is the number of elements. In $\mathcal{D}_e$ we store pairs (v, i) where i is a brick in P_j and v is the maximum gain of brick i for the right-hand side of e. The pairs are stored in lexicographical order. Since there are at most n bricks in P_j, the data structure will have at most n elements. Initially, we can build $\mathcal{D}_e$ in time $nt \cdot \Delta^{\mathcal{O}(r^2+s^2)}$ (this is replicated for each edge). Now consider a change to the instance. Recall that we are looking at changes that affect only a single brick, namely the upper and lower bounds within that brick change. We are going to update the data structure $\mathcal{D}_e$ (for each edge) to reflect the changes and we are going to recompute the edge value of each edge e using $\mathcal{D}_e$. Then we

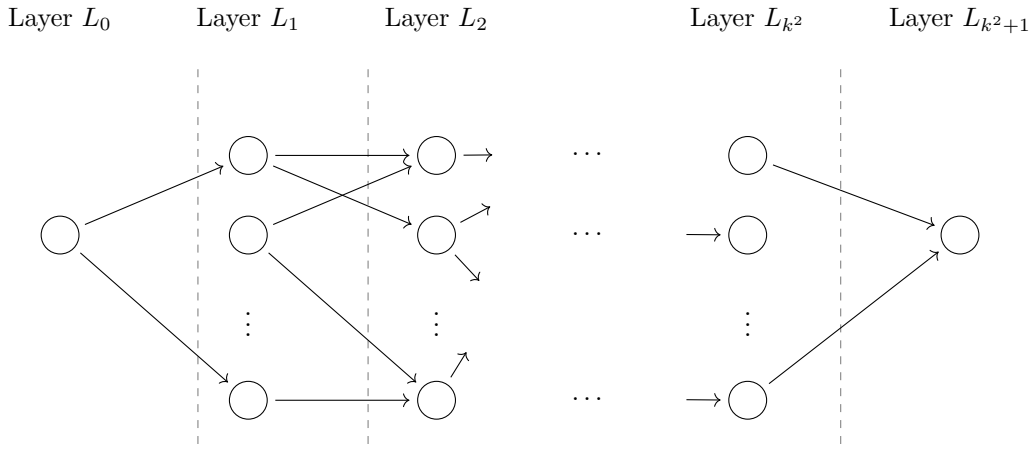


Figure 1 This figure shows an example for a layered graph obtained while solving the (P, k)-best step problem. There are $k^2 + 2$ layers, visually separated by gray dashed lines. This includes one source layer L_0, one target layer L_{k^2+1} both with just a single node representing the zero sum. Further there are k^2 layers with $(2k^3\Delta + 1)^r$ nodes each, where in one layer the nodes stand for all reachable partial sums. Two points v, w from adjacent layers L_{j-1}, L_j will be connected if the difference of the corresponding partial sums, namely $w - v$, can be obtained by a solution y of variables from only one brick of P_j (with $\|y\|_1 \leq k$). The weight of the edge will be the largest gain for the objective function $c^T y$ over all possible bricks. For the sake of clarity both the values of the nodes and the edges are not illustrated.

simply solve the Longest Path problem again. Let $P_j \in P$ be the set that contains the brick i that has changed in some variable. We only have to consider edges from L_{j-1} to L_j, since none of the other edges are affected by the change. For a relevant edge e we compute the previous value v and current value v' that the brick i would produce (before and after the bounds have changed). In $\mathcal{D}_e$ we have to remove (v, i) and insert (v', i). Both operations need only $O(\log(n))$ time. Then the running time to update $\mathcal{D}_e$ for one edge is

$$k \log(t) \cdot \Delta^{\mathcal{O}(r^2+s^2)} + \mathcal{O}(\log(n)) \leq k \log(nt) \cdot \Delta^{\mathcal{O}(r^2+s^2)}.$$

In order to update the edge value of e using $\mathcal{D}_e$, we simply have to find the maximum element in $\mathcal{D}_e$, which again takes time $\mathcal{O}(\log(n))$. To summarize, the total time to update the (P, k)-best step after a change to a single brick consists of (1) updating each $\mathcal{D}_e$, (2) finding the maximum in each $\mathcal{D}_e$, and (3) solving the Longest Path problem. We conclude that the update time is

$$k \log(nt) \cdot \Delta^{\mathcal{O}(r^2+s^2)} \cdot (k\Delta)^{\mathcal{O}(r)} + \log(n) \cdot (k\Delta)^{\mathcal{O}(r)} + (k\Delta)^{\mathcal{O}(r)} \leq k^{\mathcal{O}(r)} \Delta^{\mathcal{O}(r^2+s^2)} \cdot \log(nt).$$ ◀

4 The Augmenting Step Algorithm

In this section we will assume that all lower and upper bounds are finite and give a complete algorithm for this case. Later, we will explain how to cope with infinite bounds. We start by showing how to converge to an optimal solution when an initial feasible solution is given. To

compute the initial solution, we also apply this algorithm on a slightly modified instance. The approach resembles the procedure in previous literature, although we apply the results from the previous section to speed up the computation of augmenting steps.

Let x be a feasible solution for the n-fold ILP, in particular $\mathcal{A}x = b$. Let x^* be an optimal one. Theorem 4 states that we can decompose the difference vector $x' = x^* - x$ into at most $2nt - 1$ weighted Graver basis elements, that is

$$x' = x^* - x = \sum_{j=1}^{2nt-1} \lambda_j g_j.$$

For intuition, consider the following simple approach (this is similar to the algorithm by Hemmecke et al. [10]). Suppose we are able to guess the best vector $\lambda_i g_i = \operatorname{argmax}_j\{c^T(\lambda_j g_j)\}$ regarding the gain for the objective function. This pair of step length λ_i and Graver element g_i is called the Graver best step. Then we can augment the current solution x by adding $\lambda_i g_i$ to it, i.e., we set $x \leftarrow x + \lambda_i g_i$. Feasibility follows because all g_j are sign-compatible. This procedure is repeated until no improving step is possible and therefore x must be optimal. In each iteration this decreases the gap to the optimal solution by a factor of at least $1 - 1/(2nt)$ by the pigeonhole principle. It may be costly to guess the precise Graver best step, but for our purposes it will suffice to find an augmenting step that is approximately as good.

We will now describe how to guess λ_i. Since $x + \lambda_i g_i$ is feasible, we have that $\lambda_i g_i \leq u - x \leq u - \ell$ and $\lambda_i g_i \geq \ell - x \geq \ell - u$. Let $(g_i)_j \in \operatorname{supp}(g_i)$ be some non-zero variable. If $(g_i)_j > 0$, then $\lambda_i \leq (\lambda_i g_i)_j \leq u_j - \ell_j$. Otherwise, $(g_i)_j < 0$ and $\lambda_i \leq -(\lambda_i g_i)_j \leq -(\ell_j - u_j) = u_j - \ell_j$. Hence, it suffices to check all values in the range $\{1, \ldots, \Gamma\}$, where $\Gamma = \max_j\{u_j - \ell_j\}$. Proceeding like in [7], we lower the time a bit further by not taking every value into consideration. Instead, we look at guesses of the form $\lambda' = 2^k$ for $k \in \{0, \ldots, \lfloor \log(\Gamma) \rfloor\}$. Doing so we lose a factor of at most 2 regarding the improvement of the objective function, since $c^T(\lambda' g_i) > 0.5 \cdot c^T(\lambda_i g_i)$ when taking $\lambda' = 2^{\lfloor \log(\lambda_i) \rfloor} > \lambda_i/2$. Fix λ' to the value above. Next we describe how to compute an augmenting step that is at least as good as $\lambda' g_i$. Note that g_i is a solution of

$$\begin{aligned} \mathcal{A}y &= 0 \\ \|y\|_1 &\leq k \\ \lceil \frac{\ell - x}{\lambda'} \rceil \leq y &\leq \lfloor \frac{u - x}{\lambda'} \rfloor, \end{aligned}$$

where $k = \mathcal{O}(rs\Delta)^{rs}$ is the bound on the norm of Graver elements from Theorem 5. Suppose we have guessed some partition $P = \{P_1, \ldots, P_{k^2}\}$ of the bricks such that of each P_j only a single brick has non-zero variables in g_i. Clearly, the augmenting step $\lambda' y^*$, where y^* is a (P, k)-best step with bounds $\overline{\ell} = \lceil \frac{\ell - x}{\lambda'} \rceil$ and $\overline{u} = \lfloor \frac{u - x}{\lambda'} \rfloor$ would be at least as good as $\lambda' g_i$. Indeed Theorem 10 explains how to compute such a (P, k)-best step dynamically and when we add $\lambda' g_i$ to x we only change the bounds of at most k^3 many variables. Hence, it is very efficient to recompute (P, k)-best steps until we have converged to the optimal solution. However, valid choices of λ' and P might be different in every iteration. Regarding λ', we simply compute (P, k)-best steps for each of the $\mathcal{O}(\log(\Gamma))$ many guesses and take the best among them. We proceed similarly for P. We guess a small number of partitions and guarantee that always at least one of them is valid. For this purpose we employ splitters. More precisely, we compute a (n, k, k^2) splitter of the n bricks. Since g_i has a norm bounded by k, it can also only use at most k bricks. Therefore the splitter always contains a partition $P = \{P_1, \ldots, P_{k^2}\}$ where g_i only uses a single brick in every P_j.

To recap, in every iteration we solve a (P, k)-best step problem for every guess λ' and every partition P in the splitter and take the overall best solution as an improving direction $\lambda' y^*$. Then we update our solution x by adding $\lambda' y^*$ onto it. At most k^2 many bricks change

(and within each brick only k variables can change) and therefore we can efficiently recompute the (P, k)-best steps for every guess for the next iteration. This way we guarantee that we improve the solution by a factor of at least $1 - 1/(4nt)$ in every iteration. The explicit running time of these steps will be analyzed in the next theorem.

Recall that we still have to find an initial solution. This solution can indeed be computed by using the augmenting step algorithm described above. Roughly speaking, we modify our n-fold matrix by introducing new variables, such that it admits a trivial initial solution. Introducing a new objective function which penalizes using these new variables, an optimal solution clearly corresponds to an initial solution of the original n-fold ILP. The detailed procedure is given in the full version of this paper.

▶ **Theorem 11.** *The dynamic augmenting step algorithm described above computes an optimal solution for the n-fold Integer Linear Program problem in time* $(rs\Delta)^{\mathcal{O}(r^2s+s^2)} \cdot \mathcal{O}(L^2 \cdot nt \log^5(nt))$ *when finite variable bounds are given for each variable. Here L is the encoding length of the largest occurring number in the input.*

Proof. Due to Theorem 4 we know that the difference vector of an optimal solution x^* to our current solution x, i.e. $x' = x^* - x$, can be decomposed into $2nt - 1$ weighted Graver basis elements. Hence, if we adjust our solution x with the Graver best step, we reduce the gap between the value of an optimal solution and our current solution by a factor of at least $1 - 1/(2nt)$ due to the pigeonhole principle. Our algorithm finds an augmenting step that is at least half as good as the Graver best step. Therefore, the gap to the optimal solution is still reduced by at least a factor of $1 - 1/(4nt)$.

Regarding the running time we first have to compute the splitter. Theorem 7 says, that this can be done in time $k^{\mathcal{O}(1)} \cdot n \log(n) = (rs\Delta)^{\mathcal{O}(rs)} \cdot n \log(n)$. Next we have to try all values for the weight λ. Due to our step-length we get $\mathcal{O}(\log(\Gamma))$ guesses. Recall that Γ denotes the largest difference between an upper bound and the corresponding lower bound, i.e., $\Gamma = \max_j\{u_j - \ell_j\}$. Fixing one, we have to find the best improving direction regarding each of the $((rs\Delta)^{\mathcal{O}(rs)})^{\mathcal{O}(1)} \log(n) = (rs\Delta)^{\mathcal{O}(rs)} \log(n)$ partitions. In the first iteration we have to set up the tables in time $k^{\mathcal{O}(r)} \cdot \Delta^{\mathcal{O}(r^2+s^2)} \cdot nt = (rs\Delta)^{\mathcal{O}(r^2s)} \cdot \Delta^{\mathcal{O}(r^2+s^2)} \cdot nt$ by computing the gain for each possible summand for each set and setting up the data structure. In each following iteration we update each table and search for the optimum in time $k^{\mathcal{O}(r)} \cdot \Delta^{\mathcal{O}(r^2+s^2)} \cdot \log(nt) = (rs\Delta)^{\mathcal{O}(r^2s)} \cdot \Delta^{\mathcal{O}(r^2+s^2)} \cdot \log(nt)$. Now it remains to bound the number I of iterations needed to converge to an optimal solution. To obtain such a bound we calculate:

$$1 > (1 - 1/(4nt))^I |c^T(x^* - x)|.$$

By reordering the term, we get

$$I < \frac{-\log(|c^T(x^* - x)|)}{\log(1 - 1/(4nt))}.$$

As $\log(1 + x) = \Theta(x)$, we can bound $\log(1 - 1/(4nt))$ by $\Theta(-1/(4nt))$ and thus

$$I < \mathcal{O}(\frac{-\log(|c^T(x^* - x)|)}{-1/(4nt)}) \leq \mathcal{O}(4nt \log(|c^T(x^* - x)|)).$$

As the maximal difference between the current solution x and an optimal one x^* can be at most the maximal value of c times the largest number in between the bounds for each variable, we get $|c^T(x^* - x)| \leq nt \max_i |c_i| \cdot \Gamma$ and thus

$$I < \mathcal{O}(4nt \log(|c^T(x^* - x)|)) \leq \mathcal{O}(nt \log(nt \max_i |c_i| \cdot \Gamma)) \leq \mathcal{O}(nt \log(nt\Gamma \max_i |c_i|)).$$

Let L denote the encoding length of largest integer in the input. Clearly 2^L bounds the largest absolute value in c and thus we get

$$I < \mathcal{O}(nt\log(nt\Gamma \max_i |c_i|)) = \mathcal{O}(nt\log(nt\Gamma 2^L)) = \mathcal{O}(nt\log(nt\Gamma 2^L)).$$

Hence after this amount of steps by always improving the gain by a factor of at least $1-1/(4nt)$ we close the gap between the initial solution and an optimal one. Given this, we can now bound the overall running time with:

$$\underbrace{(rs\Delta)^{\mathcal{O}(rs)}\cdot n\log(n)}_{\text{Splitter}} + \underbrace{(rs\Delta)^{\mathcal{O}(rs)}\log(n)}_{\text{Partitions}}\cdot\underbrace{(rs\Delta)^{\mathcal{O}(r^2s)}\cdot(rs\Delta)^{\mathcal{O}(r^2+s^2)}\cdot nt}_{\text{First Iteration}} +$$
$$\underbrace{\mathcal{O}(nt\log(nt\Gamma 2^L))}_{I}\cdot\underbrace{\mathcal{O}(\log(\Gamma))}_{\lambda\text{ Guesses}}\cdot\underbrace{(rs\Delta)^{\mathcal{O}(rs)}\log(n)}_{\text{Partitions}}\cdot\underbrace{(rs\Delta)^{\mathcal{O}(r^2s)}\cdot(rs\Delta)^{\mathcal{O}(r^2+s^2)}\cdot\log(nt))}_{\text{Update Time}}$$
$$= \mathcal{O}((nt\log(nt\Gamma 2^L))\cdot\mathcal{O}(\log(\Gamma))\cdot(rs\Delta)^{\mathcal{O}(r^2s+s^2)}\cdot\log^2(nt)$$
$$= (rs\Delta)^{\mathcal{O}(r^2s+s^2)}\cdot\mathcal{O}(\log^2(\Gamma+2^L)\cdot nt\log^3(nt)).$$

Here *Splitter* denotes the time to compute the initial set $\mathcal{P}$ of partitions and *Partitions* denotes the cardinality of $\mathcal{P}$. *First Iteration* is the time to solve the first iteration of the (P,k)-best step problem. Further λ *Guesses* is the number of guesses we have to do to get the right weight and lastly *Update Time* is the time needed to solve each following (P,k)-best step including updating the bounds and data structures.

Note, that we still have to argue about finding the initial solution, since in the construction of the modified n-fold ILP the parameters slightly change. The length of a brick expand to $t' = t+r+s$, as shown in the full version of this paper. This, however, can be hidden in the $\mathcal{O}$-Notation of $(rs\Delta)^{\mathcal{O}(r^2s+s^2)}$. Further, Γ', the biggest difference in upper and lower bounds can change as we introduced new variables admitting new bounds. The difference between the bounds of old variables does not change. For the new variables, however, the difference can be as large as $\|b'\|_\infty$. Thus we bound this value by

$$\|b'\|_\infty = \|b-\mathcal{A}\ell\|_\infty \le \|b\|_\infty + \|\mathcal{A}\ell\|_\infty \le \|b\|_\infty + \Delta\cdot\|\ell\|_1 \le \mathcal{O}(\Delta\cdot nt\cdot 2^L).$$

We conclude that the running time for finding an initial solution (and also the overall running time) is

$$(rs\Delta)^{\mathcal{O}(r^2s+s^2)}\mathcal{O}(\log^2(\Gamma'+2^L)nt\log^3(nt)) = (rs\Delta)^{\mathcal{O}(r^2s+s^2)}\mathcal{O}(\log^2(\Delta 2^L nt)nt\log^3(nt))$$
$$= (rs\Delta)^{\mathcal{O}(r^2s+s^2)}\cdot L^2 nt\log^5(nt).$$ ◀

Handling Infinite Bounds

Remark, that if no finite bounds are given for all variables, we have to introduce some artificial bounds first. Here we can proceed as in [7], where first the LP relaxation is solved to obtain an optimal fractional solution z^*. Using the proximity results from [7], we know that an optimal integral solution x^* exists such that $\|x^*-z^*\|_1 \le nt(rs\Delta)^{\mathcal{O}(rs)}$. This allows us to introduce artificial upper bounds for the unbounded variables. Remark that this comes at the price of solving the corresponding relaxation of the n-fold Integer Linear Program problem. However we also lessen the dependency from L^2 to L as the finite upper and lower bounds can also be bounded more strictly due to the same proximity result. This yields an overall running time of $(rs\Delta)^{\mathcal{O}(r^2s+s^2)}\cdot L\cdot nt\log^5(nt)+\text{LP}$. Nevertheless, solving this LP can be very costly, indeed it is not clear if a potential algorithm even runs in time linear in n.

Thus, it may even dominate the running time of solving the n-fold ILP with finite upper bounds. Fortunately we can circumvent the necessity of solving the LP as we will describe in the following section using new structural results.

▶ **Theorem 12.** *The dynamic augmenting step algorithm described above computes an optimal solution for the n-fold Integer Linear Program problem in time $(rs\Delta)^{\mathcal{O}(r^2s+s^2)} \cdot L \cdot nt \log^5(nt) + LP$ when some variables have infinite upper bounds. Here LP is the running time to solve the corresponding relaxation of the n-fold ILP problem.*

5 Bounds on ℓ_1-norm

In the following, we state that even with infinite variable bounds in an n-fold ILP there always exists a solution of small norm (if the n-fold ILP has a finite optimum). Therefore, we can apply the algorithm for finite variable bounds by replacing every infinite one with this value. However, due to space restrictions, the proofs are omitted. They can be found in the full version of this paper.

▶ **Lemma 13.** *If the n-fold ILP is feasible and $\overline{y}$ is some vector satisfying the variable bounds, then there exists a feasible solution x with $\|x\|_1 \leq \mathcal{O}(rs\Delta)^{rs+1} \cdot (\|\overline{y}\|_1 + \|b\|_1)$*

▶ **Lemma 14.** *If the n-fold ILP is bounded and feasible, then there exists an optimal solution x with $\|x\|_1 \leq (rs\Delta)^{\mathcal{O}(rs)} \cdot (\|b\|_1 + nt\zeta)$, where ζ denotes the largest absolute value among all finite variable bounds.*

This yields an alternative approach to solving the LP relaxation, because now we can simply replace all infinite bounds with $\pm(rs\Delta)^{\mathcal{O}(rs)} \cdot nt \cdot 2^L$. Then we can apply the algorithm that works only on finite variable bounds. The new encoding length L' of the largest integer in the input can be bounded by

$$L' \leq \log((rs\Delta)^{\mathcal{O}(rs)} \cdot 2^L \cdot nt) \leq \mathcal{O}(rs \cdot \log(rs\Delta) \cdot L \cdot \log(nt)).$$

This way we obtain the following.

▶ **Corollary 15.** *We can compute an optimal solution for an n-fold ILP in time $(rs\Delta)^{\mathcal{O}(r^2s+s^2)} \cdot L^2 \cdot nt \log^7(nt)$.*

In a similar way, we can derive the following bound on the sensitivity of an n-fold ILP. This bound is not needed in our algorithm, but may be of independent interest, since it implies small sensitivity for problems that can be expressed as n-fold ILPs.

▶ **Theorem 16.** *Let x be an optimal solution of an n-fold ILP with right-hand side b, in particular, $\mathcal{A}x = b$. If the right hand side changes to b' and the n-fold ILP still has a finite optimum, then there exists an optimal solution x' for b' ($\mathcal{A}x' = b'$) with $\|x - x'\|_1 \leq \mathcal{O}(rs\Delta)^{rs} \cdot \|b - b'\|_1$.*

It is notable that this bound does not depend on n. This is in contrast to the known bounds for the distance between LP and ILP solutions of an n-fold ILP [7].

References

1 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995. Previously appeared in STOC 1994.

2 Katerina Altmanová, Dušan Knop, and Martin Koutecký. Evaluating and Tuning n-fold Integer Programming. In *17th International Symposium on Experimental Algorithms*, volume 103 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:14, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

3 Alexandr Andoni and Krzysztof Onak. Approximating edit distance in near-linear time. *SIAM Journal on Computing*, 41(6):1635–1648, 2012. Previously appeared in STOC 2009.

4 William Cook, Jean Fonlupt, and Alexander Schrijver. An integer analogue of Carathéodory's theorem. *Journal of Combinatorial Theory, Series B*, 40(1):63–70, 1986.

5 Jesús A. De Loera, Raymond Hemmecke, Shmuel Onn, and Robert Weismantel. N-fold integer programming. *Discrete Optimization*, 5(2):231–241, 2008.

6 Doratha E. Drake and Stefan Hougardy. A linear time approximation algorithm for weighted matchings in graphs. *Journal of the ACM Transactions on Algorithms*, 1(1):107–122, 2005.

7 Friedrich Eisenbrand, Christoph Hunkenschröder, and Kim-Manuel Klein. Faster Algorithms for Integer Programs with Block Structure. In *45th International Colloquium on Automata, Languages, and Programming*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 49:1–49:13, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

8 Friedrich Eisenbrand and Robert Weismantel. Proximity results and faster algorithms for integer programming using the steinitz lemma. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 808–816. Society for Industrial and Applied Mathematics, 2018.

9 Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a Sparse Table with $O(1)$ Worst Case Access Time. *Journal of the ACM*, 31(3):538–544, 1984. Previously appeared in FOCS 1982.

10 Raymond Hemmecke, Shmuel Onn, and Lyubov Romanchuk. N-fold integer programming in cubic time. *Mathematical Programming*, pages 1–17, 2013.

11 Raymond Hemmecke, Shmuel Onn, and Robert Weismantel. N-fold integer programming and nonlinear multi-transshipment. *Optimization Letters*, 5(1):13–25, 2011.

12 Klaus Jansen, Kim-Manuel Klein, Marten Maack, and Malin Rau. Empowering the Configuration-IP - New PTAS Results for Scheduling with Setups Times. In *ITCS*, volume 124 of *LIPIcs*, pages 44:1–44:19. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019.

13 Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 217–226. SIAM, 2014.

14 Dušan Knop and Martin Kouteckỳ. Scheduling meets n-fold integer programming. *Journal of Scheduling*, pages 1–11, 2017.

15 Dušan Knop, Martin Koutecký, and Matthias Mnich. Combinatorial n-fold Integer Programming and Applications. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, pages 54:1–54:14, 2017.

16 Martin Koutecký, Asaf Levin, and Shmuel Onn. A Parameterized Strongly Polynomial Algorithm for Block Structured Integer Programs. In *45th International Colloquium on Automata, Languages, and Programming*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 85:1–85:14, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

17 Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. Splitters and Near-Optimal Derandomization. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 182–191, 1995.

18 Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 182–191. IEEE, 1995.

19 Shmuel Onn and Pauline Sarrabezolles. Huge Unimodular n-Fold Programs. *SIAM J. Discrete Math.*, 29(4):2277–2283, 2015.

20 Éva Tardos. A Strongly Polynomial Algorithm to Solve Combinatorial Linear Programs. *Operations Research*, 34(2):250–256, 1986. doi:10.1287/opre.34.2.250.

An Improved FPTAS for 0-1 Knapsack

Ce Jin
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China
jinc16@mails.tsinghua.edu.cn

Abstract

The 0-1 knapsack problem is an important NP-hard problem that admits fully polynomial-time approximation schemes (FPTASs). Previously the fastest FPTAS by Chan (2018) with approximation factor $1+\varepsilon$ runs in $\tilde{O}(n+(1/\varepsilon)^{12/5})$ time, where $\tilde{O}$ hides polylogarithmic factors. In this paper we present an improved algorithm in $\tilde{O}(n+(1/\varepsilon)^{9/4})$ time, with only a $(1/\varepsilon)^{1/4}$ gap from the quadratic conditional lower bound based on $(\min,+)$-convolution. Our improvement comes from a multi-level extension of Chan's number-theoretic construction, and a greedy lemma that reduces unnecessary computation spent on cheap items.

2012 ACM Subject Classification Theory of computation → Algorithm design techniques

Keywords and phrases approximation algorithms, knapsack, subset sum

Category Track A: Algorithms, Complexity and Games

Acknowledgements Part of this research was done while visiting Harvard University. I would like to thank Professor Jelani Nelson for introducing this problem to me, advising this project, and giving many helpful comments on my writeup.

1 Introduction

1.1 Background

In the *0-1 knapsack* problem, we are given a set I of n items where each item $i \in I$ has weight w_i and profit p_i, and we want to select a subset $J \subseteq I$ such that $\sum_{j\in J} w_j \le W$ and $\sum_{j\in J} p_j$ is maximized.

The 0-1 knapsack problem is a fundamental optimization problem in computer science and is one of Karp's 21 NP-complete problems [8]. An important field of study on NP-hard problems is to find efficient approximation algorithms. A $(1+\varepsilon)$-approximation algorithm (for a maximization problem) outputs a value SOL such that $\mathrm{SOL} \le \mathrm{OPT} \le (1+\varepsilon)\cdot \mathrm{SOL}$, where OPT denotes the optimal answer. The 0-1 knapsack problem is one of the first problems that were shown to have fully polynomial-time approximation schemes (FPTASs), i.e., algorithms with approximation factor $1+\varepsilon$ for any given $0<\varepsilon<1$ and running time polynomial in both n and $1/\varepsilon$.

There has been a long line of research on finding faster FPTASs for the 0-1 knapsack problem, as summarized in Table 1. The first algorithm with only subcubic dependence on $1/\varepsilon$ was due to Rhee [15]. Very recently, Chan [3] gave an elegant algorithm for the 0-1 knapsack problem in deterministic $O(n\log\frac{1}{\varepsilon}+(\frac{1}{\varepsilon})^{5/2}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ via simple combination of the SMAWK algorithm [1] and a standard divide-and-conquer technique. The speedup of superpolylogarithmic factor $2^{\Omega(\sqrt{\log(1/\varepsilon)})}$ is due to recent progress on $(\min,+)$-convolution [2, 16, 4]. Using an elementary number-theoretic lemma, Chan further improved the algorithm to $O(n\log\frac{1}{\varepsilon}+(\frac{1}{\varepsilon})^{12/5}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time, and obtained two new algorithms running in $\tilde{O}(\frac{1}{\varepsilon}n^{3/2})$ and $O((\frac{1}{\varepsilon})^{4/3}n+(\frac{1}{\varepsilon})^2)/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time respectively, which are faster for small n.

FPTASs on several special cases of 0-1 knapsack are also of interest. For the *unbounded knapsack* problem, where every item has infinitely many copies, Jansen and Kraft [7] obtained an $O(n+(\frac{1}{\varepsilon})^2\log^3\frac{1}{\varepsilon})$-time algorithm; the unbounded version can be reduced

Table 1 FPTASs for 0-1 knapsack.

$O(n\log n + (\frac{1}{\varepsilon})^4 \log \frac{1}{\varepsilon})$	Ibarra and Kim [6]	1975
$O(n\log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^4)$	Lawler [13]	1979
$O(n\log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^3 \log^2 \frac{1}{\varepsilon})$	Kellerer and Pferschy [11]	2004
$O(n\log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^{5/2} \log^3 \frac{1}{\varepsilon})$ (randomized)	Rhee [15]	2015
$O(n\log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^{12/5}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$	Chan [3]	2018
$O(n\log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^{9/4}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$	**This work**	
$O(\frac{1}{\varepsilon}n^3)$	Textbook algorithm	
$O(\frac{1}{\varepsilon}n^2)$	Lawler [13]	1979
$O((\frac{1}{\varepsilon})^2 n\log \frac{1}{\varepsilon})$	Kellerer and Pferschy [10]	1999
$\tilde{O}(\frac{1}{\varepsilon}n^{3/2})$ (randomized, Las Vegas)	Chan [3]	2018
$O(((\frac{1}{\varepsilon})^{4/3}n + (\frac{1}{\varepsilon})^2)/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$	Chan [3]	2018
$O(((\frac{1}{\varepsilon})^{3/2}n^{3/4} + (\frac{1}{\varepsilon})^2)/2^{\Omega(\sqrt{\log(1/\varepsilon)})} + n\log \frac{1}{\varepsilon})$	**This work**	

to 0-1 knapsack with only a logarithmic blowup in the problem size [5]. For the *subset sum* problem, where every item has $p_i = w_i$, Kellerer et al. [9] obtained an algorithm with $O(\min\{n/\varepsilon, n + (\frac{1}{\varepsilon})^2 \log \frac{1}{\varepsilon}\})$ running time, which will be used in our algorithm as a subroutine. For the *partition* problem, which is a special case of the subset sum problem where $W = \frac{1}{2}\sum_{i\in I} w_i$, Mucha et al. [14] obtained an algorithm with a subquadratic $\tilde{O}(n + (\frac{1}{\varepsilon})^{5/3})$ running time.

On the lower bound side, recent reductions showed by Cygan et al. [5] and Künnemann et al. [12] imply that 0-1 knapsack and unbounded knapsack have no FPTAS in $O((n+\frac{1}{\varepsilon})^{2-\delta})$ time, unless $(\min,+)$-convolution has truly subquadratic algorithm [14]. It remains open whether 0-1 knapsack has a matching upper bound.

1.2 Our results

In this paper we present improved FPTASs for the 0-1 knapsack problem. Our results are summarized in the following two theorems.

▶ **Theorem 1.** *There is a deterministic $(1+\varepsilon)$-approximation algorithm for 0-1 knapsack with running time $O(n\log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^{9/4}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$.*

▶ **Theorem 2.** *For $n = O(\frac{1}{\varepsilon})$, there is a deterministic $(1+\varepsilon)$-approximation algorithm for 0-1 knapsack with running time $O\Big(\big(n^{3/4}(\frac{1}{\varepsilon})^{3/2} + (\frac{1}{\varepsilon})^2\big)/2^{\Omega(\sqrt{\log(1/\varepsilon)})}\Big)$.*

Theorem 2 gives the current best time bound for $(\frac{1}{\varepsilon})^{2/3} \ll n \ll \frac{1}{\varepsilon}$, improving upon the previous $O((\frac{1}{\varepsilon})^{4/3}n + (\frac{1}{\varepsilon})^2)/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ algorithm by Chan [3]. For $n \ll (\frac{1}{\varepsilon})^{2/3}$, Chan's $\tilde{O}(\frac{1}{\varepsilon}n^{3/2})$ time randomized algorithm [3] remains the fastest.

For $n \gg \frac{1}{\varepsilon}$, Theorem 1 gives a better time bound, improving upon the previous $O(n\log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^{12/5}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ algorithm by Chan [3].

1.3 Outline of our algorithm

We give an informal overview of our improved algorithm for 0-1 knapsack.

Using a known reduction [3], it suffices to solve an easier instance of 0-1 knapsack where profits of all items satisfy $p_i \in [1,2]$. Here "solving an instance" means approximating the function $f(x) :=$ [maximum total profit of items with at most x total weight] for all $x \geq 0$, rather than for just a single point $x = W$. In this restricted case, simple greedy (sorting

according to *unit profits* p_i/w_i) gives an additive error of at most $\max_j p_j = O(1)$, so it suffices to approximate the capped function $\min\{\varepsilon^{-1}, f(x)\}$ with approximation factor $1 + O(\varepsilon)$. Chan gave an algorithm that $(1+\varepsilon)$-approximates $\min\{B, f(x)\}$ in $\tilde{O}(n + \varepsilon^{-2}B^{1/2})$ time (implied by [3, Lemma 7]), which immediately implies an $\tilde{O}(n + \varepsilon^{-5/2})$ time FPTAS by setting $B = \varepsilon^{-1}$.

Greedy. Now we explain how to use a greedy argument (described in detail in Section 5) to improve this algorithm to $\tilde{O}(n + \varepsilon^{-7/3})$ time. We sort all items (with $p_i \in [1, 2]$) in non-increasing order of unit profits p_i/w_i, and divide them into three subsets H, M, L (items with high, medium, low unit profits), where H contains the top $\Theta(\varepsilon^{-1})$ items, and L contains all items i for which $p_i/w_i \le (1 - \varepsilon^{2/3}) \cdot \min_{h \in H}\{p_h/w_h\}$, so there is a gap between the unit profits of H-items and L-items. Intuitively, there are sufficiently many H-items available, so it's not optimal to include too many cheap L-items when the knapsack capacity is not very big. To be more precise, we prove that in any optimal solution we care about (i.e., having optimal total profit smaller than ε^{-1}), the total profit contributed by L-items cannot exceed $O(\varepsilon^{-2/3})$. Hence, for subset L we only need to approximate up to $B = \Theta(\varepsilon^{-2/3})$ in $\tilde{O}(n + \varepsilon^{-2}B^{1/2}) = \tilde{O}(n + \varepsilon^{-7/3})$ time. Subset H has only $O(\varepsilon^{-1})$ items and can be solved using Chan's $\tilde{O}(\varepsilon^{-4/3}n + \varepsilon^{-2})$ algorithm in $\tilde{O}(\varepsilon^{-7/3})$ time. To solve subset M, we round down the profit value p_i for every item $i \in M$, so that the unit profit p_i/w_i becomes a power of $(1+\varepsilon)$. Then there are $O(\varepsilon^{-1/3})$ distinct unit profit values in M. Items with the same unit profit can be solved together using the efficient FPTAS for *subset sum* by Kellerer et al. [9] in $\tilde{O}(n + \varepsilon^{-2})$ time. Finally we merge the results for H, M, L. The total time complexity is $\tilde{O}(n + \varepsilon^{-7/3})$.

Multi-level number-theoretic construction. The above approach invokes two of Chan's algorithms: an $\tilde{O}(n + \varepsilon^{-2}B^{1/2})$ algorithm (useful for small B) and an $\tilde{O}(\varepsilon^{-4/3}n + \varepsilon^{-2})$ algorithm (useful for small n). The key ingredient in these algorithms is a number-theoretic lemma: we can $(1+\varepsilon)$-approximate all profit values $p_i \in [1, 2]$ by multiples of elements from a small set $\Delta \subset [\delta, 2\delta]$ of size $|\Delta| = \tilde{O}(\frac{\delta}{\varepsilon})$ (small $|\Delta|$ can reduce the additive error incurred from rounding).

Chan obtained an $\tilde{O}(n + \varepsilon^{-2}B^{2/5})$ time algorithm using some additional tricks. First, evenly partition Δ into r subsets $\Delta^{(1)}, \ldots, \Delta^{(r)}$, and divide the items into $P = P^{(1)} \cup \cdots \cup P^{(r)}$ accordingly, so that profit values from $P^{(j)}$ are approximated by $\Delta^{(j)}$-multiples. To $(1+\varepsilon)$-approximate the profit function f_j for each $P^{(j)}$, pick a threshold $B_0 \ll B$, and return the combination of a $(1+\varepsilon)$-approximation of $\min\{f_j, B_0\}$ and an εB_0-additive-approximation of $\min\{f_j, B\}$. Since the size of $\Delta^{(j)}$ is only $|\Delta|/r$, the latter function can be approximated faster when $r \gg 1$. Finally, merge f_j over all $1 \le j \le r$. By fine-tuning the parameters r, δ, B_1, the time complexity is improved to $\tilde{O}(n + \varepsilon^{-2}B^{2/5})$.

Our new algorithm extends this technique to *multiple levels*. To $(1+\varepsilon)$-approximate the profit function f_j for each $P^{(j)}$, we will pick $B_0 \ll B_1 \ll \cdots \ll B_{d-1} \ll B_d \approx B$, and compute the εB_{i-1}-additive-approximation of $\min\{f_j, B_i\}$, for all $i \in [d]$. An issue of this multi-level approach is that, different levels have different optimal parameters δ_i and different $\Delta_i^{(1)}, \ldots, \Delta_i^{(r)}$, but we have to stick to the same partition of items $P = P^{(1)} \cup \cdots \cup P^{(r)}$ over all levels. We overcome this issue by enforcing that $\Delta_i^{(j)}$ at level i must be generated by multiples of elements from $\Delta_{i-1}^{(j)}$ at level $i-1$, so that $P^{(j)}$ can be approximated by $\Delta_i^{(j)}$-multiples for all levels. To achieve this, we need a multi-level version of the number-theoretic lemma. We will discuss this part in detail in Section 4.

Using this multi-level construction, we obtain algorithms in $\tilde{O}(n + \varepsilon^{-2}B^{1/3})$ time and $\tilde{O}(\varepsilon^{-3/2}n^{3/4} + \varepsilon^{-2})$ time. Combining these improved algorithms with the greedy argument previously described (the threshold which splits M and L needs to be adjusted accordingly), we obtain an algorithm in $\tilde{O}(n + \varepsilon^{-9/4})$ time as claimed in Theorem 1.

2 Preliminaries

Throughout this paper, $\log x$ stands for $\log_2 x$, and $\tilde{O}(f)$ stands for $O(f \cdot \operatorname{poly}\log(f))$.

We will describe our algorithm with approximation factor $1 + O(\varepsilon)$, which can be lowered to $1 + \varepsilon$ if we scale down ε by a constant factor at the beginning.

We are only interested in the case where $n = O(\varepsilon^{-4})$. For greater n, Lawler's $O(n \log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^4)$ algorithm [13] is already near-optimal. Hence we assume $\log n = O(\log \varepsilon^{-1})$.

Assume $0 < w_i \le W$ and $p_i > 0$ for every item i. Then a trivial lower bound of the maximum total profit is $\max_j p_j$. At the beginning, we discard all items i with $p_i \le \frac{\varepsilon}{n} \max_j p_j$. Since the total profit of discarded items is at most $\varepsilon \max_j p_j$, the optimal total profit is only reduced by a factor of $1 + O(\varepsilon)$. So we can assume that $\frac{\max_j p_j}{\min_j p_j} \le \frac{n}{\varepsilon}$.

We adopt Chan's terminology in presenting our algorithm. For a set I of items, define the profit function

$$f_I(x) = \max\left\{ \sum_{i \in J} p_i : \sum_{i \in J} w_i \le x, \;\; J \subseteq I \right\}$$

over non-negative real numbers $x \ge 0$. Note that f_I is a monotone (nondecreasing) step function. The *complexity* of a monotone step function refers to the number of its steps.

We say that a function $\tilde{f}$ approximates a function f with factor $1 + \varepsilon$ if $\tilde{f}(x) \le f(x) \le (1 + \varepsilon)\tilde{f}(x)$ for all $x \ge 0$. We say that $\tilde{f}$ approximates f with additive error δ if $\tilde{f}(x) \le f(x) \le \tilde{f}(x) + \delta$ for all $x \ge 0$. Our goal is to approximate f_I with factor $1 + O(\varepsilon)$ on the input item set I.

Let I_1, I_2 be two disjoint subsets of items, and $I = I_1 \cup I_2$. We have $f_I = f_{I_1} \oplus f_{I_2}$, where $\oplus$ denotes the $(\max, +)$-convolution, defined by $(f \oplus g)(x) = \max_{0 \le x' \le x}(f(x') + g(x - x'))$. If two non-negative monotone step functions f, g are approximated with factor $1 + \varepsilon$ by functions $\tilde{f}, \tilde{g}$ respectively, then $f \oplus g$ is also approximated by $\tilde{f} \oplus \tilde{g}$ with factor $1 + \varepsilon$.

For a monotone step function f with range[1] contained in $\{0\} \cup [A, B]$, we can obtain a function $\tilde{f}$ with complexity only $O(\varepsilon^{-1} \log(B/A))$ which approximates f with factor $1 + \varepsilon$, by simply rounding f down to powers of $(1 + \varepsilon)$. For our purposes, B/A will be bounded by polynomial of n and $1/\varepsilon$, hence we may always assume that the approximation results are monotone step functions with complexity $\tilde{O}(\varepsilon^{-1})$.

For an item set I with the same profit $p_i = p$ for every item $i \in I$, the step function f_I can be exactly computed in $O(n \log n)$ time by simple greedy: the function values are $0, p, 2p, \ldots, np$ and the x-breakpoints are $w_1, w_1 + w_2, \ldots, w_1 + \cdots + w_n$, after sorting all w_i's in nondecreasing order. We say that a monotone step function is *p-uniform* if its function values are of the form $0, p, 2p, \ldots, lp$ for some l. We say that a p-uniform function is *pseudo-concave* if the differences of consecutive x-breakpoints are nondecreasing from left to right. In the previous case where all p_i's are equal to p, f_I is indeed p-uniform and pseudo-concave.

[1] Here *range* refers to the set of possible output values of the function.

3 Chan's techniques

In this section we review several useful lemmas by Chan [3].

3.1 Merging profit functions

▶ **Lemma 3** ([3, Lemma 2(i)]). *Let $f_1, \dots, f_m$ be monotone step functions with total complexity $O(n)$ and ranges contained in $\{0\} \cup [A, B]$. Then we can compute a monotone step function that approximates $f_1 \oplus \cdots \oplus f_m$ with factor $1 + O(\varepsilon)$ and complexity $\tilde{O}(\frac{1}{\varepsilon} \log B/A)$ in $O(n) + \tilde{O}((\frac{1}{\varepsilon})^2 m / 2^{\Omega(\sqrt{\log(1/\varepsilon)})} \log B/A)$ time.*

▶ Remark 4. Lemma 3 is proved using a divide-and-conquer method, which was also used previously in [10]. The speedup of superpolylogarithmic factor $2^{\Omega(\sqrt{\log(1/\varepsilon)})}$ is due to recent progress on $(\min, +)$-convolution [2, 16, 4].

Lemma 3 enables us to focus on a simpler case where all $p_i \in [1, 2]$. For the general case, we divide the items into $O(\log \frac{\max_j p_j}{\min_j p_j}) = O(\log \varepsilon^{-1})$ groups, each containing items with $p_i \in [2^j, 2^{j+1}]$ for some j (which can be rescaled to $[1, 2]$), and finally merge the profit functions of all groups by using Lemma 3 in $\tilde{O}(n + \varepsilon^{-2})$ time.

Assuming ε^{-1} is an integer and every $p_i \in [1, 2]$, we can round every p_i down to a multiple of ε, introducing only a $1 + \varepsilon$ error factor. Then there are only $m = O(\varepsilon^{-1})$ distinct values of p_i. For every value of p_i, the corresponding profit function f_i is p_i-uniform and pseudo-concave, and can be obtained by simple greedy (as discussed in Section 2).

3.2 Approximating big profit values using greedy

When all p_i's are small, simple greedy gives good approximation guarantee when the answer is big enough.

▶ **Lemma 5.** *Suppose $p_i \in [1, 2]$ for all $i \in I$. For $B = \Omega(\varepsilon^{-1})$, the function f_I can be approximated with additive error $O(\varepsilon B)$ in $O(n \log n)$ time.*

Proof. Sort the items in nonincreasing order of unit profit p_i/w_i. Let $\tilde{f}$ be the monotone step function resulting from greedy, with function values $0, p_1, p_1 + p_2, \dots, p_1 + \cdots + p_n$ and x-breakpoints $0, w_1, w_1 + w_2, \dots, w_1 + \cdots + w_n$. It approximates f_I with an additive error of $\max_i p_i \le 2 \le O(\varepsilon B)$ for $B = \Omega(\varepsilon^{-1})$. ◀

When every $p_i \in [1, 2]$, we set $B = \varepsilon^{-1}$ and let f_H denote the result from greedy (Lemma 5). Then we only need to obtain a function f_L which $1 + O(\varepsilon)$ approximates $\min\{f_I, B\}$, and finally return $\max\{f_L, f_H\}$ as a $1 + O(\varepsilon)$ approximation of f_I (because when $f_I(x)$ exceeds B, an additive error $O(\varepsilon B)$ implies $1 + O(\varepsilon)$ approximation factor).

3.3 Approximation using Δ-multiples of small set Δ

For a set Δ of numbers, we say that p is a Δ-multiple if it is a multiple of δ for some $\delta \in \Delta$.

▶ **Lemma 6** ([3, Lemma 5]). *Let $f_1, \dots, f_m$ be monotone step functions with ranges contained in $[0, B]$. Let $\Delta \subset [\delta, 8\delta]$. If every f_i is p_i-uniform and pseudo-concave for some $p_i \in [1, 2]$ which is a Δ-multiple, then we can compute a monotone step function that approximates $\min\{f_1 \oplus \cdots \oplus f_m, B\}$ with additive error $O(|\Delta|\delta)$ in $\tilde{O}(Bm/\delta)$ time.*

▶ Remark 7. An intuition of Lemma 6 is as follows. When p_i's are exact multiples of δ, standard dynamic programming algorithm maintains a result array of length B/δ, and adding a new f_i to the result can be done in linear time (by exploiting the pseudo-concavity of f_i using the SMAWK algorithm[2]). Now if the next p_i to be considered is a multiple of $\delta' \neq \delta$, we first have to round down the current results to multiples of δ', introducing an additive error of δ'. We round our results for $|\Delta| - 1$ times, so smaller $|\Delta|$ implies smaller overall additive error.

▶ **Corollary 8.** *Let $f_1, \ldots, f_m$ be monotone step functions with ranges contained in $[0, B]$. If every f_i is p_i-uniform and pseudo-concave for some $p_i \in [1, 2]$, then we can compute a monotone step function that approximates $\min\{f_1 \oplus \cdots \oplus f_m, B\}$ with factor $1 + O(\varepsilon)$ in $\tilde{O}(\varepsilon^{-1}Bm)$ time.*

Proof. Assuming ε^{-1} is an integer, adjust every p_i down to the nearest multiple of ε, and adjust f_i accordingly. This introduces a $1 + \varepsilon$ overall error factor. Then use Lemma 6 with $\delta = \varepsilon, \Delta = \{\varepsilon\}$ to compute the desired function in $\tilde{O}(Bm\varepsilon^{-1})$ time. ◀

4 Extending Chan's number-theoretic construction

As mentioned in Section 1.3, the main results of this section are two approximation algorithms in $\tilde{O}(n + \varepsilon^{-2}B^{1/3})$ and $\tilde{O}(\varepsilon^{-3/2}n^{3/4} + \varepsilon^{-2})$ time respectively (the latter time bound assumes $n = O(1/\varepsilon)$). These results rely on Lemma 6.

4.1 Number-theoretic construction

To avoid checking boundary conditions, from now on we assume $\varepsilon > 0$ is sufficiently small.

Our algorithm extends Chan's technique by using a multi-level structure defined as follows.

▶ **Definition 9.** *For fixed parameters $\delta_1, \delta_2, \ldots, \delta_d$ satisfying condition*

$$\varepsilon \leq \delta_1,\ \delta_i \leq \delta_{i+1}/2,\ \delta_d \leq 1/8 \tag{1}$$

and a finite real number set $\Delta_1 \subset [\delta_1, 8\delta_1]$, a set tower *$(\Delta_1, \Delta_2, \ldots, \Delta_d)$ generated by Δ_1 is defined by recurrence*[3]

$$\Delta_{i+1} := [\delta_{i+1}, 8\delta_{i+1}] \cap \bigcup_{k \in \mathbb{Z}} k\Delta_i, \quad i = 1, 2, \ldots, d-1. \tag{2}$$

We refer to Δ_1 as the base set *and Δ_d as the* top set *of this set tower. We also say that the base set Δ_1 generates the top set Δ_d.*

If Δ_d^* is the top set generated by a singleton base set $\Delta_1^* = \{x\}$, then for every $y \in \Delta_d^*$ we say x generates *y.*

We have the following simple facts about set towers.

▶ **Proposition 10.** *If x generates y then $x \in \Delta_1$ implies $y \in \Delta_d$. Conversely, for every $y \in \Delta_d$, there exists $x \in \Delta_1$ which generates y, and for every $1 \leq i \leq d$ there exists $z \in \Delta_i$ such that both y/z and z/x are integers.*

---

[2] The SMAWK algorithm [1] finds all row-minima in an $n \times n$ matrix A satisfying the *Monge property* $A[i, j] + A[i + 1, j + 1] \leq A[i, j + 1] + A[i + 1, j]$ using only $O(n)$ queries.

[3] For a number k and a set A of numbers, $kA := \{ka : a \in A\}$.

▶ **Proposition 11.** *For any $1 \le i \le d$, $|\Delta_i| \le 8^{i-1}(\delta_i/\delta_1)|\Delta_1|$, and we can compute Δ_i in $\tilde{O}(8^{i-1}(\delta_i/\delta_1)|\Delta_1|)$ time given Δ_1 as input.*

Proof. For $2 \le i \le d$, we have

$$|\Delta_i| = \Big|[\delta_i, 8\delta_i] \cap \bigcup_{k\in\mathbb{Z}} k\Delta_{i-1}\Big| \le \sum_{x\in\Delta_{i-1}} 8\delta_i/x \le |\Delta_{i-1}|8\delta_i/\delta_{i-1}.$$

The proof of size upper bounds follows by induction. Elements of Δ_i can be generated straightforwardly within the time bound. ◀

▶ **Lemma 12.** *Let $T_1, T_2, \ldots, T_d$ be positive real numbers satisfying $T_1 \ge 2$ and $T_{i+1} \ge 2T_i$. There exist at least $T_d/(\log T_d)^{O(d)}$ integers t satisfying the following condition: t can be written as a product of integers $t = n_1 n_2 \cdots n_d$, such that $n_1 n_2 \cdots n_i \in (T_i/2, T_i]$ for every $1 \le i \le d$.*

The proof of Lemma 12 is deferred to Appendix A. Lemma 12 helps us prove the following fact, which is a multi-level extension of [3, Lemma 6].

▶ **Lemma 13.** *For any parameters $\delta_1, \ldots, \delta_d$ satisfying condition (1), there exists a base set Δ_1 of size $\frac{\delta_1}{\varepsilon} \cdot (\log \varepsilon^{-1})^{O(d)}$, such that every $p \in [1, 2]$ can be approximated by a Δ_d-multiple with additive error $O(\varepsilon)$, where Δ_d is the top set generated by Δ_1.*

This base set Δ_1 can be constructed in $\tilde{O}(\varepsilon^{-1}\delta_1^{-1})$ time deterministically.

Proof. Let $P = \{1, 1+\varepsilon, 1+2\varepsilon, \ldots, 1+\lfloor\frac{1}{\varepsilon}\rfloor\varepsilon\}$. It suffices to approximate every value $p \in P$ with additive error ε using Δ_d-multiples. For any $p \in P$ and $y \in \Delta_d \subset [\delta_d, 8\delta_d]$, p is approximated with additive error ε by a multiple of y if and only if $y \in \bigcup_{j\in\mathbb{Z}} \left[\frac{p-\varepsilon}{j}, \frac{p}{j}\right]$.

Our constructed base set Δ_1 will satisfy $\Delta_1 \subset [\delta_1, 4\delta_1]$. Suppose integers $k_1, k_2, \ldots, k_{d-1}$ satisfy

$$k_1 k_2 \cdots k_{i-1} \in [\delta_i/\delta_1, 2\delta_i/\delta_1], \quad \text{for every } 2 \le i \le d. \tag{3}$$

Then by Definition 9, for any $x \in \Delta_1 \subset [\delta_1, 4\delta_1]$, we have $xk_1k_2\cdots k_{i-1} \in \Delta_i$ for every $2 \le i \le d$.

For any integer j satisfying

$$k_1 k_2 \cdots k_{d-1} j \in [p/(4\delta_1), p/(2\delta_1)], \tag{4}$$

the interval $[\frac{p-\varepsilon}{k_1k_2\cdots k_{d-1}j}, \frac{p}{k_1k_2\cdots k_{d-1}j}]$ is contained in $[\delta_1, 4\delta_1]$.

We say an integer K is *good* for p, if K can be expressed as a product of integers $k_1k_2\cdots k_{d-1}j$ satisfying conditions (3) and (4). For such K, any $x \in [\frac{p-\varepsilon}{K}, \frac{p}{K}] \cap \Delta_1$ generates an element $y = xk_1k_2\cdots k_{d-1} \in \Delta_d \cap [\frac{p-\varepsilon}{j}, \frac{p}{j}]$ such that p can be approximated by a multiple of y with additive error ε.

By Lemma 12, the number of good integers K for p is at least

$$\frac{p/(4\delta_1)}{\big(\log(p/(4\delta_1))\big)^{O(d)}} = \Omega\Big(\frac{\delta_1^{-1}}{(\log\varepsilon^{-1})^{O(d)}}\Big),$$

and at most $p/(2\delta_1) = O(\delta_1^{-1})$, by (4). Using conditions (3) and (4) we can compute all these K's by simple dynamic programming. We denote the union of their associated intervals by

$$I_p := \bigcup_{K \text{ good for } p} \left[\frac{p-\varepsilon}{K}, \frac{p}{K}\right] \subset [\delta_1, 4\delta_1]. \tag{5}$$

Note that these intervals are disjoint since $p/(K+1) \le (p-\varepsilon)/K$, so the total length of I_p can be lower-bounded as

$$\lambda(I_p) \ge \frac{\delta_1^{-1}}{(\log \varepsilon^{-1})^{O(d)}} \cdot \frac{p-(p-\varepsilon)}{\max K} \ge \frac{\varepsilon}{(\log \varepsilon^{-1})^{O(d)}}. \tag{6}$$

We have seen that p is approximated by a Δ_d-multiple with additive error ε as long as $\Delta_1 \cap I_p \neq \emptyset$. We compute I_p for every $p \in P$, and use the standard greedy algorithm (for Hitting Set problem) to construct a base set $\Delta_1 \subset [\delta_1, 4\delta_1]$ which intersects with every I_p: in each round we find a point $x \in [\delta_1, 4\delta_1]$ that hits the most I_p's, include x into Δ_1, and remove the I_p's that are hit by x. In each round the number of remaining I_p's decreases by

$$s := \frac{\min_p \lambda(I_p)}{4\delta_1 - \delta_1} \ge \frac{\varepsilon/\delta_1}{(\log \varepsilon^{-1})^{O(d)}},$$

so the solution size $|\Delta_1|$ is upper-bounded by

$$1 + \log_{1/(1-s)} |P| = O\left(\frac{\log |P|}{s}\right) = \frac{\delta_1}{\varepsilon} (\log \varepsilon^{-1})^{O(d)}.$$

To implement this greedy algorithm, we use standard data structures (for example, segment trees) that support finding x which hits the most intervals, reporting an interval hit by x, removing an interval, all in logarithmic time per operation. The number of operations is bounded by the total number of small intervals, so the running time is at most $\tilde{O}(|P| \cdot \frac{p}{2\delta_1}) = \tilde{O}(\delta_1^{-1}\varepsilon^{-1})$. ◀

The following lemma evenly partitions the base set Δ_1 into r subsets $\Delta_1^{(1)}, \ldots, \Delta_1^{(r)}$, and partitions the profit values $P = \{p_1, \ldots, p_m\}$ into $P^{(1)} \cup \cdots \cup P^{(r)}$, so that $P^{(j)}$ can be approximated by $\Delta_d^{(j)}$-multiples. An additional requirement is that $P^{(1)}, \ldots, P^{(r)}$ should have size $O(|P|/r)$ each.

▶ **Lemma 14.** *Let $\delta_1, \ldots, \delta_d$ be parameters satisfying condition (1). Let $P = \{p_1, \ldots, p_m\} \subset [1,2]$ with $m = O(\varepsilon^{-1})$. Given a positive integer parameter $r = O(\min\{\frac{\delta_1}{\varepsilon}, m\})$, there exist r base sets $\Delta_1^{(1)}, \Delta_1^{(2)}, \ldots, \Delta_1^{(r)}$ each of size $\frac{\delta_1}{\varepsilon r} \cdot (\log \varepsilon^{-1})^{O(d)}$, and a partition of $P = P^{(1)} \cup P^{(2)} \cup \cdots \cup P^{(r)}$ each of size $O(m/r)$, such that for every $1 \le j \le r$, every $p \in P^{(j)}$ can be approximated by a $\Delta_d^{(j)}$-multiple with additive error $O(\varepsilon)$, where $\Delta_d^{(j)}$ is the top set generated by $\Delta_1^{(j)}$.*

These r base sets and the partition of P can be computed by a deterministic algorithm in $\tilde{O}(\varepsilon^{-2}/r)$ time .

Proof. First construct the base set Δ_1 of size $\frac{\delta_1}{\varepsilon}(\log \varepsilon^{-1})^{O(d)}$ from Lemma 13 in $\tilde{O}(\delta_1^{-1}\varepsilon^{-1}) = \tilde{O}(\varepsilon^{-2}/r)$ time, and compute the top set Δ_d that it generates. By Proposition 11, $|\Delta_d| \le 8^{d-1}\frac{\delta_d}{\delta_1}|\Delta_1| \le \frac{\delta_d}{\varepsilon}(\log \varepsilon^{-1})^{O(d)}$. Generate and sort all Δ_d-multiples in interval $[1,2]$. For every $p \in P$, use binary search to find the Δ_d-multiple $ky \le p\,(y \in \Delta_d)$ closest to p, and then add p to the set Q_x, where $x \in \Delta_1$ is an element that generates y. (Q_x is initialized as empty for every $x \in \Delta_1$.) Then remove every x with $Q_x = \emptyset$ from Δ_1, so that $|\Delta_1| \le m$, while every $p \in P$ can still be approximated with $O(\varepsilon)$ additive error by a Δ_d-multiple.

Let $D := \max\{r, |\Delta_1|\}$ and let $s := \lceil m/D \rceil$. For every $x \in \Delta_1$ we divide Q_x evenly into small subsets each having size at most s. The total number of these small subsets is

$$\sum_{x \in \Delta_1} \lceil |Q_x|/s \rceil \le |\Delta_1| + \sum_{x \in \Delta_1} |Q_x|/s = |\Delta_1| + m/s \le 2D.$$

We merge these small subsets into r groups, each having at most $\lceil 2D/r \rceil$ small subsets. Then, define set $P^{(j)}$ as the union of small subsets from the j-th group, and let base set $\Delta_1^{(j)}$ contain $x \in \Delta_1$ if any of these small subsets comes from Q_x. So $|\Delta_1^{(j)}| \le \lceil 2D/r \rceil = \frac{\delta_1}{\varepsilon r}(\log \varepsilon^{-1})^{O(d)}$, and $|P^{(j)}| \le s \cdot \lceil 2D/r \rceil = O(m/D) \cdot O(D/r) = O(m/r)$. ◀

4.2 Approximation using set towers

We first prove a slightly improved version of Corollary 8. The only purpose of this lemma is to get rid of the $(\log \varepsilon^{-1})^{O(\log\log \varepsilon^{-1})}$ factor in the final running time.

▶ **Lemma 15.** *Let $f_1, \ldots, f_m$ be monotone step functions with ranges contained in $[0, B]$ for some $1 \le B \le O(\varepsilon^{-1})$. If every f_i is p_i-uniform and pseudo-concave for some $p_i \in [1, 2]$, then we can compute a monotone step function that approximates $\min\{f_1 \oplus \cdots \oplus f_m, B\}$ with factor $1 + O(\varepsilon)$ in $\tilde{O}(\varepsilon^{-1}(Bm + \varepsilon^{-1})/B^{0.01})$ time.*

Proof. Using Lemma 13 with parameters $d = 1, \delta_1 = \varepsilon B^{0.01}$, we get $\Delta \subset [\delta_1, 8\delta_1]$ with size $|\Delta| \le \tilde{O}(\delta_1/\varepsilon) = \tilde{O}(B^{0.01})$, in $\tilde{O}(\varepsilon^{-2}/B^{0.01})$ time. Adjust every p_i down to the nearest Δ-multiple, and adjust f_i accordingly. This introduces at most $1+O(\varepsilon)$ error factor. Then use Lemma 6 to compute a monotone step function f_H that approximates $\min\{f_1 \oplus \cdots \oplus f_m, B\}$ with additive error $e = O(|\Delta|\delta_1) = \tilde{O}(\varepsilon B^{0.02})$, in $\tilde{O}(B^{0.99} m\varepsilon^{-1})$ time.

Let $B_L := e/\varepsilon$, and use Corollary 8 to compute a monotone step function f_L that approximates $\min\{f_1 \oplus \cdots \oplus f_m, B_L\}$ with factor $1+O(\varepsilon)$ in only $\tilde{O}(B_L m\varepsilon^{-1}) = \tilde{O}(B^{0.02} m\varepsilon^{-1})$ time.

Since f_H approximates $\min\{f_1 \oplus \cdots \oplus f_m, B\}$ with additive error εB_L, $\max\{f_L, f_H\}$ is a $1 + O(\varepsilon)$ approximation of $\min\{f_1 \oplus \cdots \oplus f_m, B\}$. ◀

Now we can approximate the profit function $\min\{B, \bigoplus_{p_k \in P^{(j)}} f_k\}$ for each group $P^{(j)}$, using the multi-level approach described in Section 1.3.

▶ **Lemma 16.** *Let $f_1, \ldots, f_m$ be given monotone step functions with ranges contained in $[0, B]$, and every f_k is p_k-uniform and pseudo-concave for some $p_k \in [1, 2]$. Assume $m = O(\varepsilon^{-1})$, $\Omega(\varepsilon^{-0.01}) \le B \le O(\varepsilon^{-1})$. Let r be a given positive integer parameter with $r = O(m), r = o(B)$.*

We can set $d = O(\log\log \varepsilon^{-1})$ and choose d parameters $\delta_1, \ldots, \delta_d$ satisfying condition (1), such that the following statement holds:

Let $P^{(1)} \cup \cdots \cup P^{(r)}$ be the partition of set $P = \{p_1, \ldots, p_m\}$ returned by the algorithm in Lemma 14 with the above parameters. Then for any $1 \le j \le r$, using the base set $\Delta_1^{(j)}$ from Lemma 14, we can compute a monotone step function that approximates $\min\{B, \bigoplus_{p_k \in P^{(j)}} f_k\}$ with factor $1 + O(\varepsilon)$, in $(\varepsilon^{-2}/r^{0.01} + m\varepsilon^{-1}B^{1/2}/r^{3/2})(\log \varepsilon^{-1})^{O(d)}$ time.

Proof. We can assume $B \ge 4r$, and define d to be the unique positive integer such that

$$2^{2^{d-1}} \le \frac{\sqrt{B}}{\sqrt{r}} < 2^{2^d} = 4^{2^{d-1}}.$$

Then $d = O(\log\log \frac{\sqrt{B}}{\sqrt{r}}) = O(\log\log \varepsilon^{-1})$. Pick $\alpha \in [2, 4)$ such that

$$\alpha^{2^{d-1}} = \frac{\sqrt{B}}{\sqrt{r}}. \tag{7}$$

Define

$$\delta_i := \varepsilon\sqrt{Br} \Big/ \alpha^{2^{d-i}},\ 0 \le i \le d. \tag{8}$$

Then

$$\delta_d = \frac{\varepsilon\sqrt{Br}}{\alpha},\ \delta_1 = \varepsilon r \tag{9}$$

Note that $\delta_d = \varepsilon\sqrt{B}\cdot O(\sqrt{r}) = \varepsilon\sqrt{B}\cdot o(\sqrt{B}) = \varepsilon\cdot o(B) = o(1)$. Hence the parameters $\delta_1,\ldots,\delta_d$ satisfy condition (1) for sufficiently small ε.

The base set $\Delta_1^{(j)}$ from Lemma 14 has size $\frac{\delta_1}{\varepsilon r}(\log\varepsilon^{-1})^{O(d)}$. We compute the generated set tower $\Delta_1^{(j)},\Delta_2^{(j)},\ldots,\Delta_d^{(j)}$. By Proposition 11, $|\Delta_i^{(j)}| \le \frac{\delta_i}{\varepsilon r}(\log\varepsilon^{-1})^{O(d)}$. Let

$$t := \max\left\{\alpha,\ \max_j |\Delta_i^{(j)}| \Big/ \frac{\delta_i}{\varepsilon r}\right\} = (\log\varepsilon^{-1})^{O(d)} \tag{10}$$

and define

$$B_i := Bt\Big/\alpha^{2^{d-i}},\ 0\le i\le d. \tag{11}$$

Then $B \le B_d \le B\cdot(\log\varepsilon^{-1})^{O(d)}$, and it's easy to verify that

$$|\Delta_i^{(j)}|\cdot\delta_i \le B_{i-1}\varepsilon,\ \ (1\le i\le d). \tag{12}$$

For every $1\le i\le d$, adjust every $p_k\in P^{(j)}$ down to the nearest $\Delta_i^{(j)}$-multiple and adjust f_k accordingly, which introduces a $1+O(\varepsilon)$ error factor. Then use Lemma 6 to obtain a monotone step function g_i which approximates $\min\{\bigoplus_{p_k\in P^{(j)}} f_k, B_i\}$ with additive error $O(|\Delta_i^{(j)}|\delta_i) = O(\varepsilon B_{i-1})$ in $\tilde{O}(|P^{(j)}|B_i/\delta_i)$ time.

Then we use Lemma 15 to obtain a monotone step function g_0 which approximates $\min\{\bigoplus_{p_k\in P^{(j)}} f_k, B_0\}$ with $1+O(\varepsilon)$ factor, in $\tilde{O}(\varepsilon^{-1}(|P^{(j)}|B_0+\varepsilon^{-1})B_0^{-0.01})$ time. Notice that $B_0 = rt$.

Finally, $\max\{g_0,g_1,g_2,\ldots,g_d\}$ is a $1+O(\varepsilon)$ approximation of $\min\{\bigoplus_{p_k\in P^{(j)}} f_k, B_d\}$, where $B_d\ge B$. Overall running time is

$$\begin{aligned}
&\tilde{O}(\varepsilon^{-1}(|P^{(j)}|B_0+\varepsilon^{-1})B_0^{-0.01}) + \sum_{1\le j\le d}\tilde{O}(|P^{(j)}|B_j/\delta_j)\\
&= \tilde{O}\big(\varepsilon^{-1}(\frac{m}{r}\cdot(rt)+\varepsilon^{-1})(rt)^{-0.01}\big) + d\cdot\tilde{O}\big(\frac{m}{r}B_d/\delta_d\big)\\
&= (\varepsilon^{-2}/r^{0.01} + m\varepsilon^{-1}B^{1/2}/r^{3/2})(\log\varepsilon^{-1})^{O(d)}.
\end{aligned}$$

◀

Now we merge the results from all r groups, and obtain an approximation of the final result $\min\{f_1\oplus\cdots\oplus f_m, B\}$.

▶ **Lemma 17.** *Let $f_1,\ldots,f_m$ be given monotone step functions with ranges contained in $[0,B]$, and every f_k is p_k-uniform and pseudo-concave for some $p_k\in[1,2]$. Assume $m = O(1/\varepsilon), \Omega(\varepsilon^{-0.01}) \le B \le O(\varepsilon^{-1})$. We can approximate $\min\{f_1\oplus\cdots\oplus f_m, B\}$ with factor $1+O(\varepsilon)$ in $O(\varepsilon^{-2}B^{1/3}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time.*

Proof. Assume $m\ge\varepsilon^{-1}$, by adding zero functions which do not change the answer.

Let $r = o(B)$ be a positive integer parameter to be determined later.

Using Lemma 14 and Lemma 16, we can get a partition of $\{p_1,\ldots,p_m\} = P^{(1)}\cup\cdots\cup P^{(r)}$ and then get an $1+O(\varepsilon)$ approximation of $\min\{\bigoplus_{p_k\in P^{(j)}} f_k, B\}$ for every $1\le j\le r$, in $r\cdot(\varepsilon^{-2}/r^{0.01} + m\varepsilon^{-1}B^{1/2}/r^{3/2})(\log\varepsilon^{-1})^{O(d)} = (r^{0.99}+\sqrt{B/r})\varepsilon^{-2}(\log\varepsilon^{-1})^{O(\log\log\varepsilon^{-1})}$ overall time.

Then we use Lemma 3 to merge all these r functions in $\tilde{O}((\frac{1}{\varepsilon})^2 r/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time.

Setting $r = B^{1/3}2^{c\sqrt{\log(1/\varepsilon)}}$, where $c > 0$ is some small enough constant, the total complexity is

$$O(\varepsilon^{-2}B^{1/3}/2^{\Omega(\sqrt{\log(1/\varepsilon)})}).$$ ◀

▶ **Lemma 18.** *Let I be a set of m items with $p_i \in [1,2]$ for every $i \in I$, where $\Omega(\varepsilon^{-2/3}) \le m \le O(\varepsilon^{-1})$. One can approximate f_I with factor $1+O(\varepsilon)$ in $O(\varepsilon^{-3/2}m^{3/4}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time.*

Proof. Let $f_1, \ldots, f_m$ denote the profit functions of the m items.

Let $r = o(m^{1/2})$ be a positive integer parameter to be determined later. Obtain a partition of $\{p_1, \ldots, p_m\} = P^{(1)} \cup \cdots \cup P^{(r)}$ using Lemma 14. Let $B := \max_i \sum_{p \in P^{(i)}} p \le 2\max_i |P^{(i)}| = \Theta(m/r)$. Then $r = o(B)$. Use Lemma 16 to get an $1+O(\varepsilon)$ approximation of $\bigoplus_{p_k \in P^{(j)}} f_k = \min\{\bigoplus_{p_k \in P^{(j)}} f_k, B\}$ for every $1 \le j \le r$, in $r \cdot (\varepsilon^{-2}/r^{0.01} + m\varepsilon^{-1}B^{1/2}/r^{3/2})(\log \varepsilon^{-1})^{O(d)} = (\varepsilon^{-2}r^{0.99} + m^{3/2}\varepsilon^{-1}/r)(\log \varepsilon^{-1})^{O(\log\log \varepsilon^{-1})}$ overall time.

Then we use Lemma 3 to merge all these r functions in $\tilde{O}((\frac{1}{\varepsilon})^2 r/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time.

Setting $r = m^{3/4}\varepsilon^{1/2}2^{c\sqrt{\log(1/\varepsilon)}}$, where $c > 0$ is some small enough constant, the total complexity is

$$O(\varepsilon^{-3/2}m^{3/4}/2^{\Omega(\sqrt{\log(1/\varepsilon)})}).$$ ◀

▶ **Corollary 19** (restated Theorem 2). *For $n = O(\frac{1}{\varepsilon})$, there is a deterministic $(1+\varepsilon)$-approximation algorithm for 0-1 knapsack in $O\Big((n^{3/4}(\frac{1}{\varepsilon})^{3/2} + (\frac{1}{\varepsilon})^2)/2^{\Omega(\sqrt{\log(1/\varepsilon)})}\Big)$ time.*

Proof. Divide the items into $O(\log \frac{n}{\varepsilon})$ groups, each containing items with $p_i \in [2^j, 2^{j+1}]$ for some j. Use Lemma 18 to solve each group, and merge them using Lemma 3. ◀

5 Main algorithm

5.1 A greedy lemma

Our improved algorithm uses the following lemma, which gives an upper bound on the total profit of cheap items (with low p_i/w_i) in an optimal knapsack solution.

▶ **Lemma 20.** *Let H, L be two subsets of items with $p_i \in [1,2]$. Let $W = \sum_{h\in H} w_h$ and $q = \min_{h\in H} \frac{p_h}{w_h}$. Suppose $\max_{l\in L} \frac{p_l}{w_l} \le q(1-\alpha)$ for some $0 < \alpha < 1$. Let $f = f_H \oplus f_L, \tilde{f} = f_H \oplus \min\{\frac{2}{\alpha}, f_L\}$. Then for every $x \le W$, $f(x) = \tilde{f}(x)$.*

Proof. By greedy, $f(W) = \sum_{h\in H} p_h = \tilde{f}(W)$ clearly holds. Now consider $0 \le x < W$. Suppose $f_L(x') + f_H(x-x')$ achieves its maximum value at $x' = w_L$, i.e., $f(x) = f_L(w_L) + f_H(x-w_L)$. It suffices to prove $f_L(w_L) \le \frac{2}{\alpha}$.

Let $J \subseteq H$ be a subset of items with total weight $w_J \le x - w_L$ and total profit achieving optimal value $f_H(x-w_L)$. Let $K \subseteq H\backslash J$ be a subset of items with total weight w_K, such that $w_K \le w_L$, and $w_K + w_i > w_L$ for every remaining item $i \in H\backslash(J \cup K)$. Such K can be constructed by a simple greedy algorithm.

Since $w_J + w_K \le (x - w_L) + w_L < W = \sum_{h\in H} w_h$, the remaining set $H\backslash(J\cup K)$ contains at least one item h_0. Hence, $w_L - w_K < w_{h_0} = p_{h_0}/\frac{p_{h_0}}{w_{h_0}} \le 2/q$, and equivalently $qw_K > qw_L - 2$.

Since $J \cup K$ is a subset of H with total weight bounded by x, we have $f_H(x) \geq \sum_{k\in K} p_k + \sum_{j\in J} p_j$, and thus $f_H(x) - f_H(x-w_L) = f_H(x) - \sum_{j\in J} p_j \geq \sum_{k\in K} p_k \geq qw_K > qw_L - 2$.

Hence $qw_L - 2 < f_H(x) - f_H(x-w_L) \leq f(x) - f_H(x-w_L) = f_L(w_L) \leq q(1-\alpha)w_L$, which shows that $q\alpha w_L \leq 2$. So $f_L(w_L) \leq q(1-\alpha)w_L \leq qw_L \leq 2/\alpha$, which concludes the proof. ◀

5.2 FPTAS for Subset Sum

We will use the efficient FPTAS for the subset sum problem by Kellerer et al. [9] as a subroutine in our algorithm.

▶ **Lemma 21** ([9], implicit). *Let I be a set of n items and W be a number. We can obtain a list S of $O(\frac{1}{\varepsilon})$ numbers in $O(n + (\frac{1}{\varepsilon})^2 \log \frac{1}{\varepsilon})$ time, such that for every $s \leq W$ that is the subset sum $s = \sum_{j\in J} w_j$ of some subset $J \subseteq I$, there exists $s' \in S$ with $s - \varepsilon W \leq s' \leq s$.*

▶ Remark 22. This result wasn't explicitly stated in [9], but can be easily seen from their analysis of the correctness of the FPTAS.

▶ **Corollary 23.** *Let I be a set of n items with $p_i \in [1,2]$ and $p_i = w_i$ for every item $i \in I$. We can approximate f_I with factor $1 + O(\varepsilon)$ in $O(n \log n + \varepsilon^{-2} \log \frac{1}{\varepsilon} \log n)$ time.*

Proof. Notice that approximating s with additive error εW implies approximation factor $1 + O(\varepsilon)$ for $W/2 \leq s \leq W$. So we simply apply Lemma 21 with $W = 2^j$ for $0 \leq j \leq 1 + \log n$, and merge all obtained lists. ◀

5.3 Improved algorithm

▶ **Lemma 24.** *Let I be a set of n items with $p_i \in [1,2]$ for every $i \in I$. We can approximate f_I with factor $1 + O(\varepsilon)$ in $O(n \log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^{9/4} / 2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time.*

Proof. Let $B = \lceil \varepsilon^{-1} \rceil$ and assume $n \geq B$ (if $n < B$, we can simply apply Lemma 18). By Lemma 5, we can approximate f_I with additive error $O(\varepsilon B)$ in $O(n \log \frac{1}{\varepsilon})$ time, so we only need to approximate $\min\{f_I, B\}$ with factor $1 + O(\varepsilon)$.

We sort the items by their unit profits p_i/w_i. Let set H contain the top B items with the highest unit profits. Define $q = \min_{h\in H} \frac{p_h}{w_h}$, and let M be the set of remaining items i with $q(1-\alpha) \leq \frac{p_i}{w_i} \leq q$, where parameter $0 < \alpha < 1$ is to be determined later. Let set L contain the remaining items not included in H or M.

Using Lemma 18, we can compute $\tilde{f}_H$ which approximates f_H with factor $1 + O(\varepsilon)$ in time $O(B^{3/4}\varepsilon^{-3/2}/2^{\Omega(\sqrt{\log(1/\varepsilon)})}) = O(\varepsilon^{-9/4}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$.

Since $\max_{l\in L} \frac{p_l}{w_l} < q(1-\alpha)$, Lemma 20 states that $f_H \oplus f_L$ and $f_H \oplus \min\{2/\alpha, f_L\}$ agree when $x \leq W_H = \sum_{h\in H} w_h$. Since $(f_H \oplus f_L)(W_H) = \sum_{h\in H} p_h \geq B$, this implies $\min\{B, f_H \oplus f_L\} = \min\{B, f_H \oplus \min\{2/\alpha, f_L\}\}$. For every item $l \in L$, we round down p_l to a power of $1+\varepsilon$, so that there are only $\log_{1+\varepsilon} 2 = O(\varepsilon^{-1})$ distinct values. This only multiplies the approximation factor by $1 + \varepsilon$. Then we use Lemma 17 to compute an approximation of $\min\{2/\alpha, f_L\}$ with factor $1 + O(\varepsilon)$ in $\tilde{O}(\varepsilon^{-2}(2/\alpha)^{1/3}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time. We merge it with $\tilde{f}_H$ and obtain an approximation of $\min\{f_H \oplus f_L, B\}$ with factor $1 + O(\varepsilon)$.

For every $m \in M$, we round down p_m so that the unit profit p_m/w_m becomes a power of $1 + \varepsilon$. After rounding, the approximation factor is only multiplied by $1 + \varepsilon$, and there are at most $\log_{1+\varepsilon} \frac{q}{q(1-\alpha)} = O(\alpha/\varepsilon)$ distinct unit profits in M. Let M_q denote the set of items in M with unit profit q. For each q, we use Lemma 23 to obtain a $1 + \varepsilon$ approximation of the function f_{M_q} in $O(|M_q| + \varepsilon^{-2})$ time. Then we use Lemma 3 to merge these functions and obtain a $1 + \varepsilon$ approximation of f_M. The total time is $O(|M| \log n) + \tilde{O}(\alpha\varepsilon^{-3})$.

Finally we merge the functions and get an approximation of $\min\{B, f_L \oplus f_H \oplus f_M\}$ with factor $1 + O(\varepsilon)$. The total time is $O(n \log \frac{1}{\varepsilon}) + \tilde{O}(\alpha\varepsilon^{-3} + \varepsilon^{-2}(2/\alpha)^{1/3}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$, which is $O(n \log \frac{1}{\varepsilon} + \varepsilon^{-9/4}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ if we choose $\alpha = \varepsilon^{3/4}/2^{c\sqrt{\log(1/\varepsilon)}}$ for a sufficiently small constant c. ◀

▶ **Corollary 25** (restated Theorem 1). *There is a deterministic $(1+\varepsilon)$-approximation algorithm for 0-1 knapsack with running time $O(n \log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^{9/4}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$.*

Proof. Divide the items into $O(\log \frac{n}{\varepsilon})$ groups, each containing items with $p_i \in [2^j, 2^{j+1}]$ for some j. Use Lemma 24 to solve each group, and merge them using Lemma 3. ◀

References

1. Alok Aggarwal, Maria M. Klawe, Shlomo Moran, Peter Shor, and Robert Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2(1):195–208, November 1987. doi:10.1007/BF01840359.
2. David Bremner, Timothy M. Chan, Erik D. Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, Mihai Pătraşcu, and Perouz Taslakian. Necklaces, Convolutions, and X+Y. *Algorithmica*, 69(2):294–314, June 2014. doi:10.1007/s00453-012-9734-3.
3. Timothy M. Chan. Approximation Schemes for 0-1 Knapsack. In *Proceedings of the 1st Symposium on Simplicity in Algorithms (SOSA)*, pages 5:1–5:12, 2018. doi:10.4230/OASIcs.SOSA.2018.5.
4. Timothy M. Chan and Ryan Williams. Deterministic APSP, Orthogonal Vectors, and More: Quickly Derandomizing Razborov-Smolensky. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1246–1255, 2016. doi:10.1137/1.9781611974331.ch87.
5. Marek Cygan, Marcin Mucha, Karol Węgrzycki, and Michał Włodarczyk. On Problems Equivalent to (Min,+)-Convolution. *ACM Trans. Algorithms*, 15(1):14:1–14:25, January 2019. doi:10.1145/3293465.
6. Oscar H. Ibarra and Chul E. Kim. Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems. *Journal of the ACM (JACM)*, 22(4):463–468, October 1975. doi:10.1145/321906.321909.
7. Klaus Jansen and Stefan E.J. Kraft. A faster FPTAS for the Unbounded Knapsack Problem. *European Journal of Combinatorics*, 68:148–174, 2018. doi:10.1016/j.ejc.2017.07.016.
8. Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer US, 1972. doi:10.1007/978-1-4684-2001-2_9.
9. Hans Kellerer, Renata Mansini, Ulrich Pferschy, and Maria Grazia Speranza. An efficient fully polynomial approximation scheme for the Subset-Sum Problem. *Journal of Computer and System Sciences*, 66(2):349–370, 2003. doi:10.1016/S0022-0000(03)00006-0.
10. Hans Kellerer and Ulrich Pferschy. A New Fully Polynomial Time Approximation Scheme for the Knapsack Problem. *Journal of Combinatorial Optimization*, 3(1):59–71, July 1999. doi:10.1023/A:1009813105532.
11. Hans Kellerer and Ulrich Pferschy. Improved Dynamic Programming in Connection with an FPTAS for the Knapsack Problem. *Journal of Combinatorial Optimization*, 8(1):5–11, March 2004. doi:10.1023/B:JOCO.0000021934.29833.6b.
12. Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. On the Fine-Grained Complexity of One-Dimensional Dynamic Programming. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 21:1–21:15, 2017. doi:10.4230/LIPIcs.ICALP.2017.21.
13. Eugene L. Lawler. Fast Approximation Algorithms for Knapsack Problems. *Mathematics of Operations Research*, 4(4):339–356, 1979. doi:10.1287/moor.4.4.339.

14 Marcin Mucha, Karol Węgrzycki, and Michał Włodarczyk. Subquadratic Approximation Scheme for Partition. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 70–88, 2019. doi:10.1137/1.9781611975482.5.

15 Donguk Rhee. Faster fully polynomial approximation schemes for knapsack problems. Master's thesis, Massachusetts Institute of Technology, 2015. URL: http://hdl.handle.net/1721.1/98564.

16 Ryan Williams. Faster All-pairs Shortest Paths via Circuit Complexity. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 664–673, 2014. doi:10.1145/2591796.2591811.

A Proof of Lemma 12

▶ **Theorem 26** (Reminder of Lemma 12). *Let $T_1, T_2, \ldots, T_d$ be positive real numbers satisfying $T_1 \geq 2$ and $T_{i+1} \geq 2T_i$. There exist at least $T_d/(\log T_d)^{O(d)}$ integers t satisfying the following condition: t can be written as a product of integers $t = n_1 n_2 \cdots n_d$, such that $n_1 n_2 \cdots n_i \in (T_i/2, T_i]$ for every $1 \leq i \leq d$.*

Proof. For every $1 \leq k \leq d$, we say an ordered k-tuple $(p_1, p_2, \ldots, p_k)$ is *valid* if every p_i is prime, and $p_1 p_2 \cdots p_i \in (T_i/2, T_i]$ for every $1 \leq i \leq k$. Then the product $t = p_1 p_2 \cdots p_d$ of any valid d-tuple $(p_1, \ldots, p_d)$ satisfies our condition. For any integer t, there are at most $d!$ different valid d-tuples with product t (which could be obtained by permuting t's prime factors). Let N_k denote the number of valid k-tuples. Then it suffices to show $N_d/(d!) \geq T_d/(\log T_d)^{O(d)}$.

By the prime number theorem and Bertrand-Chebyshev theorem, there exists a positive constant C such that

$$\pi(x) - \pi(x/2) \geq x/(C \log x), \text{ for all } x \geq 2,$$

where $\pi(x)$ denotes the number of primes less than or equal to x. We will prove $N_k \geq T_k/(C \log T_k)^k$ for all $1 \leq k \leq d$ by induction.

First note that this statement is trivial for $k = 1$. For $k \geq 2$, a valid k-tuple $(p_1, \ldots, p_k)$ can be obtained by appending any prime $p_k \in \big(T_k/(2P), T_k/P\big]$ to any valid $(k-1)$-tuple $(p_1, \ldots, p_{k-1})$ with product $P = p_1 \cdots p_{k-1} \leq T_{k-1}$. The number of such primes p_k is

$$\pi(T_k/P) - \pi\big(T_k/(2P)\big) \geq \frac{T_k/P}{C \log(T_k/P)} \geq \frac{T_k/T_{k-1}}{C \log T_k}.$$

Summing over all valid $(k-1)$-tuples, we have

$$N_k \geq N_{k-1} \cdot \frac{T_k/T_{k-1}}{C \log T_k} \geq \frac{T_{k-1}}{(C \log T_{k-1})^{k-1}} \cdot \frac{T_k/T_{k-1}}{C \log T_k} \geq \frac{T_k}{(C \log T_k)^k}.$$

Hence, $N_d \geq T_d/(C \log T_d)^d$ by induction. Observe that $T_d \geq 2^d$ and we have

$$\frac{N_d}{d!} \geq \frac{T_d}{(Cd \log T_d)^d} \geq \frac{T_d}{(C \log^2 T_d)^d} \geq \frac{T_d}{(\log T_d)^{O(d)}},$$

which finishes the proof. ◀

Testing the Complexity of a Valued CSP Language

Vladimir Kolmogorov
Institute of Science and Technology Austria, Klosterneuburg, Austria
vnk@ist.ac.at

Abstract

A *Valued Constraint Satisfaction Problem* (VCSP) provides a common framework that can express a wide range of discrete optimization problems. A VCSP instance is given by a finite set of variables, a finite domain of labels, and an objective function to be minimized. This function is represented as a sum of terms where each term depends on a subset of the variables. To obtain different classes of optimization problems, one can restrict all terms to come from a fixed set Γ of cost functions, called a *language*.

Recent breakthrough results have established a complete complexity classification of such classes with respect to language Γ: if all cost functions in Γ satisfy a certain algebraic condition then all Γ-instances can be solved in polynomial time, otherwise the problem is NP-hard. Unfortunately, testing this condition for a given language Γ is known to be NP-hard. We thus study exponential algorithms for this *meta-problem*. We show that the tractability condition of a finite-valued language Γ can be tested in $O(\sqrt[3]{3}^{|D|} \cdot poly(\texttt{size}(\Gamma)))$ time, where D is the domain of Γ and $poly(\cdot)$ is some fixed polynomial. We also obtain a matching lower bound under the *Strong Exponential Time Hypothesis* (SETH). More precisely, we prove that for any constant $\delta < 1$ there is no $O(\sqrt[3]{3}^{\delta|D|})$ algorithm, assuming that SETH holds.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Valued Constraint Satisfaction Problems, Exponential time algorithms, Exponential Time Hypothesis

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.77

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1803.02289`.

Funding *Vladimir Kolmogorov*: supported by the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement no 616160.

1 Introduction

Minimizing functions of discrete variables represented as a sum of low-order terms is a ubiquitous problem occurring in many real-world applications. Understanding complexity of different classes of such optimization problems is thus an important task. In a prominent *VCSP framework* (which stands for *Valued Constraint Satisfaction Problem*) a class is parameterized by a set Γ of cost functions of the form $f : D^n \to \mathbb{Q} \cup \{\infty\}$ that are allowed to appear as terms in the objective. Set Γ is usually called a *language*.

Different types of languages give rise to many interesting classes. A widely studied type is *crisp* languages Γ, in which all functions f are $\{0, \infty\}$-valued. They correspond to *Constraint Satisfaction Problems (CSPs)*, whose goal is to decide whether a given instance has a feasible solution. Feder and Vardi conjectured in [11] that there exists a dichotomy for CSPs, i.e. every crisp language Γ is either tractable or NP-hard. This conjecture was refined by Bulatov, Krokhin and Jeavons [7], who proposed a specific algebraic condition

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 77; pp. 77:1–77:12


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

that should separate tractable languages from NP-hard ones. The conjecture was verified for many special cases [27, 5, 3, 2, 8], and was finally proved in full generality by Bulatov [6] and Zhuk [31].

At the opposite end of the VCSP spectrum are the finite-valued CSPs, in which functions do not take infinite values. In such VCSPs, the feasibility aspect is trivial, and one has to deal only with the optimization issue. One polynomial-time algorithm that solves tractable finite-valued CSPs is based on the so-called basic linear programming (BLP) relaxation, and its applicability (also for the general-valued case) was fully characterized by Kolmogorov, Thapper and Živný [22]. The complexity of finite-valued CSPs was completely classified by Thapper and Živný [29], where it is shown that all finite-valued CSPs not solvable by BLP are NP-hard.

A dichotomy is also known to hold for general-valued CSPs, i.e. when cost functions in Γ are allowed to take arbitrary values in $\mathbb{Q} \cup \{\infty\}$. First, Kozik and Ochremiak showed [23] that languages that do not satisfy a certain algebraic condition are NP-hard. Kolmogorov, Krokhin and Rolínek then proved [21] that all other languages are tractable, assuming the (now established) dichotomy for crisp languages conjectured in [7].

In this paper languages Γ that satisfy the condition in [23] are called *solvable*. Since optimization problems encountered in practice often come without any guarantees, it is natural to ask what is the complexity of checking solvability of a given language Γ. We envisage that an efficient algorithm for this problem could help in theoretical investigations, and could also facilitate designing optimization approaches for tackling specific tasks.

Checking solvability of a given language is known as a *meta-problem* or a *meta-question* in the literature. Note that it can be solved in polynomial time for languages on a fixed domain D (since the solvability condition can be expressed by a linear program with $O(|D|^{|D|^m})$ variables and polynomial number of constraints, where $m = 2$ if the language is finite-valued and $m = 4$ otherwise). This naive solution, however, becomes very inefficient if D is a part of the input (which is what we assume in this paper).

The meta-problem above was studied by Thapper and Živný for finite-valued languages [29], and by Chen and Larose for crisp languages [10]. In both cases it was shown to be NP-complete. We therefore focus on exponential-time algorithms. We obtain the following results for the problem of checking solvability of a given finite-valued language Γ:

- An algorithm with complexity $O(\sqrt[3]{3}^{|D|} \cdot \mathit{poly}(\mathtt{size}(\Gamma)))$, where D is the domain of Γ and $\mathit{poly}(\cdot)$ is some fixed polynomial.
- Assuming the *Strong Exponential Time Hypothesis* (SETH), we prove that for any constant $\delta < 1$ the problem cannot be solved in $O(\sqrt[3]{3}^{\delta|D|} \cdot \mathit{poly}(\mathtt{size}(\Gamma)))$ time.

We also present a few weaker results for general-valued languages (see Section 3).

Other related work. There is a vast literature devoted to exponential-time algorithms for various problems, both on the algorithmic side and on the hardness side. Hardness results usually assume one of the following two hypotheses [15, 16, 9].

▶ **Conjecture 1** (Exponential Time Hypothesis (ETH)). *Deciding satisfiability of a* 3*-CNF-SAT formula on* n *variables cannot be solved in* $O(2^{o(n)})$ *time.*

▶ **Conjecture 2** (Strong Exponential Time Hypothesis (SETH)). *For any* $\delta < 1$ *there exists integer* k *such that deciding satisfiability of a* k*-CNF-SAT formula on* n *variables cannot be solved in* $O(2^{\delta n})$ *time.*

Below we discuss some results specific to CSPs. Let (k, d)-CSP be the class of CSP problems on a d-element domain where each constraint involves at most k variables. The number of variables in an instance will be denoted as n. A trivial exhaustive search for

a (k,d)-CSP instance runs in $O^*(d^n)$ time, where notation $O^*(\cdot)$ hides factors polynomial in the size of the input. For $(2,d)$-CSP instances the complexity can be improved to $O^*((d-1)^n)$ [26]. Some important subclasses of $(2,d)$-CSP can even be solved in $O^*(2^{O(n)})$ time. For example, [25] and [4] developed respectively $O(2.45^n)$ and $O^*(2^n)$ algorithms for solving the d-coloring problem. On the negative side, ETH is known to have the following implications:

- The $(2,d)$-CSP problem cannot be solved in $d^{o(n)} = 2^{o(n \log d)}$ time [30].
- The Graph Homomorphism problem cannot be solved in $2^{o\left(\frac{n \log d}{\log \log d}\right)}$ time [12]. (This problem can be viewed as a special case of $(2,d)$-CSP, in which a single binary relation is applied to different pairs of variables).

Recently, exponential-time algorithms for crisp NP-hard languages have been studied using algebraic techniques [17, 18, 24]. For example, [18] showed that the following conditions are equivalent, assuming the (now proved) algebraic CSP dichotomy conjecture: (a) ETH fails; (b) there exists a finite crisp NP-hard language Γ that can be solved in subexponential time (i.e. all Γ-instances on n variables can be solved in $O(2^{o(n)})$ time); (c) all finite crisp NP-hard languages Γ can be solved in subexponential time.

The rest of the paper is organized as follows: Section 2 gives a background on the VCSP framework, and Section 3 presents our results. All proofs are given in the full version of this paper [20].

2 Background

We denote $\overline{\mathbb{Q}} = \mathbb{Q} \cup \{\infty\}$, where ∞ is the positive infinity. A function of the form $f : D^n \to \overline{\mathbb{Q}}$ will be called a *cost function over D of arity n*. We will always assume that the set D is finite. The *effective domain* of f is the set $\mathtt{dom}\, f = \{x \mid f(x) < \infty\}$. Note that $\mathtt{dom}\, f$ can be viewed both as an n-ary relation over D and as a function $D^n \to \{0, \infty\}$. We assume that f is represented as a list of pairs $\{(x, f(x)) \ : \ x \in \mathtt{dom}\, f\}$. Accordingly, we define $\mathtt{size}(f) = \sum_{x \in \mathtt{dom}\, f} [n \log |D| + \mathtt{size}(f(x))]$, where the size of a rational number p/q (for integers p, q) is $\log(|p|+1) + \log|q|$.

▶ **Definition 1.** *A valued constraint satisfaction language Γ over domain D is a set of cost functions $f : D^n \to \overline{\mathbb{Q}}$, where the arity n depends on f and may be different for different functions in Γ. The domain of Γ will be denoted as D_Γ. For a finite Γ we define $\mathtt{size}(\Gamma) = |D| + \sum_{f \in \Gamma} \mathtt{size}(f)$.*

A language Γ is called *finite-valued* if all functions $f \in \Gamma$ take finite (rational) values. It is called *crisp* if all functions $f \in \Gamma$ take only values in $\{0, \infty\}$. We denote $\mathrm{Feas}(\Gamma) = \{\mathtt{dom}\, f \mid f \in \Gamma\}$ to be the crisp language obtained from Γ in a natural way. Throughout the paper, for a subset $A \subseteq D$ we use u_A to denote the unary function $D \to \{0, \infty\}$ with $\arg\min u_A = A$. (Domain D should always be clear from the context). For a label $a \in D$ we also write $u_a = u_{\{a\}}$ for brevity.

▶ **Definition 2.** *An instance $\mathcal{I}$ of the valued constraint satisfaction problem (VCSP) is a function $D^V \to \overline{\mathbb{Q}}$ given by*

$$f_{\mathcal{I}}(x) \ = \ \sum_{t \in T} f_t(x_{v(t,1)}, \dots, x_{v(t,n_t)}) \tag{1}$$

It is specified by a finite set of variables V, finite set of terms T, cost functions $f_t : D^{n_t} \to \overline{\mathbb{Q}}$ of arity n_t and indices $v(t,k) \in V$ for $t \in T, k = 1, \dots, n_t$. A solution to $\mathcal{I}$ is a labeling $x \in D^V$ with the minimum total value. The size of $\mathcal{I}$ is defined as $\mathtt{size}(\mathcal{I}) = |V| + |D| + \sum_{t \in T} \mathtt{size}(f_t)$.

The instance $\mathcal{I}$ is called a Γ-instance if all terms f_t belong to Γ.

The set of all Γ-instances will be denoted as $\mathrm{VCSP}(\Gamma)$. A finite language Γ is called *tractable* if all instances $\mathcal{I} \in \mathrm{VCSP}(\Gamma)$ can be solved in polynomial time, and it is *NP-hard* if the corresponding optimization problem is NP-hard. A long sequence of works culminating with recent breakthrough papers [6, 31] has established that every finite language Γ is either tractable or NP-hard.

2.1 Polymorphisms and cores

Let $\mathcal{O}_D^{(m)}$ denote the set of all operations $g : D^m \to D$ and let $\mathcal{O}_D = \bigcup_{m \geq 1} \mathcal{O}_D^{(m)}$. When D is clear from the context, we will sometimes write simply $\mathcal{O}^{(m)}$ and $\mathcal{O}$.

Any language Γ defined on D can be associated with a set of operations on D, known as the polymorphisms of Γ, which allow one to combine (often in a useful way) several feasible assignments into a new one.

▶ **Definition 3.** *An operation $g \in \mathcal{O}_D^{(m)}$ is a* polymorphism *of a cost function $f : D^n \to \overline{\mathbb{Q}}$ if, for any $x^1, x^2, \ldots, x^m \in \mathtt{dom}\, f$, we have that $g(x^1, x^2, \ldots, x^m) \in \mathtt{dom}\, f$ where g is applied component-wise.*

For any valued constraint language Γ over a set D, we denote by $\mathrm{Pol}^{(m)}(\Gamma)$ the set of all operations on $\mathcal{O}_D^{(m)}$ which are polymorphisms of every $f \in \Gamma$. We also let $\mathrm{Pol}(\Gamma) = \bigcup_{m \geq 1} \mathrm{Pol}^{(m)}(\Gamma)$.

Clearly, if g is a polymorphism of a cost function f, then g is also a polymorphism of $\mathtt{dom}\, f$. For $\{0, \infty\}$-valued functions, which naturally correspond to relations, the notion of a polymorphism defined above coincides with the standard notion of a polymorphism for relations. Note that the projections (aka dictators), i.e. operations of the form $e_n^i(x_1, \ldots, x_n) = x_i$, are polymorphisms of all valued constraint languages. Polymorphisms play the key role in the algebraic approach to the CSP, but, for VCSPs, more general constructs are necessary, which we now define.

▶ **Definition 4.** *An m-ary* fractional operation *ω on D is a probability distribution on $\mathcal{O}_D^{(m)}$. The support of ω is defined as $\mathrm{supp}(\omega) = \{g \in \mathcal{O}_D^{(m)} \mid \omega(g) > 0\}$.*

For an operation $g \in \mathcal{O}^{(m)}$ we will denote χ_g to be characteristic vector of g, i.e. the fractional operation with $\chi_g(g) = 1$ and $\chi_g(h) = 0$ for $h \neq g$.

▶ **Definition 5.** *A m-ary fractional operation ω on D is said to be a* fractional polymorphism *of a cost function $f : D^n \to \overline{\mathbb{Q}}$ if, for any $x^1, x^2, \ldots, x^m \in \mathtt{dom}\, f$, we have*

$$\sum_{g \in \mathrm{supp}(\omega)} \omega(g) f(g(x^1, \ldots, x^m)) \leq \frac{1}{m}(f(x^1) + \ldots + f(x^m)). \tag{2}$$

For a constraint language Γ, $\mathrm{fPol}^{(m)}(\Gamma)$ will denote the set of all m-ary fractional operations that are fractional polymorphisms of each function in Γ. Also, let $\mathrm{fPol}(\Gamma) = \bigcup_{m \geq 1} \mathrm{fPol}^{(m)}(\Gamma)$, $\mathrm{supp}^{(m)}(\Gamma) = \bigcup_{\omega \in \mathrm{fPol}^{(m)}(\Gamma)} \mathrm{supp}(\omega)$ and $\mathrm{supp}(\Gamma) = \bigcup_{m \geq 1} \mathrm{supp}^{(m)}(\Gamma)$.

(It is easy to check that $\mathrm{supp}(\Gamma) \subseteq \mathrm{Pol}(\Gamma)$, and $\mathrm{supp}(\Gamma) = \mathrm{Pol}(\Gamma)$ if Γ is crisp).

Next, we will need the notion of *cores*.

▶ **Definition 6.** *Language Γ on domain D is called a* core *if all operations $g \in \mathrm{supp}^{(1)}(\Gamma)$ are bijections. Subset $B \subseteq D$ is called a* core of Γ *if $B = g(D)$ for some operation $g \in \mathrm{supp}^{(1)}(\Gamma)$ and the language $\Gamma[B]$ is a core, where $\Gamma[B]$ is the language on domain B obtained by restricting each function $f : D^n \to \overline{\mathbb{Q}}$ in Γ to B^n.*

The following facts are folklore knowledge. We do not know an explicit reference (at least in the case of general-valued languages), so we prove them in [20] for completeness.

▶ **Lemma 7.** *Let B be a subset of $D = D_\Gamma$ such that $B = g(D)$ for some $g \in \mathrm{supp}^{(1)}(\Gamma)$.*
(a) *Set B is a core of Γ if and only if $|B| = \texttt{core-size}(\Gamma) \stackrel{\text{def}}{=} \min\{|g(D)| : g \in \mathrm{supp}^{(1)}(\Gamma)\}$.*
(b) *There exists vector $\omega \in \mathrm{fPol}^{(1)}(\Gamma)$ such that $g(D) \subseteq B$ for all $g \in \mathrm{supp}(\omega)$. Furthermore, if B is a core of Γ then such ω can be chosen so that $g(a) = a$ for all $g \in \mathrm{supp}(\omega)$ and $a \in B$.*
(c) *Let $\mathcal{I}$ be a Γ-instance on variables V. Then $\min_{x \in B^V} f_{\mathcal{I}}(x) = \min_{x \in D^V} f_{\mathcal{I}}(x)$.*
For a language Γ we denote $\mathcal{B}_\Gamma^{\texttt{core}}$ to be the set of subsets $B \subseteq D$ which are cores of Γ, and $\mathcal{O}_\Gamma^{\texttt{core}}$ to be set of operations $g \in \mathrm{supp}^{(1)}(\Gamma)$ such that $g(D) \in \mathcal{B}_\Gamma^{\texttt{core}}$ (or equivalently such that $|g(D)| = \texttt{core-size}(\Gamma)$).

2.2 Dichotomy theorem

Several types of operations play a special role in the algebraic approach to (V)CSP.

▶ **Definition 8.** *An operation $g \in \mathcal{O}_D^{(m)}$ is called*
- idempotent *if $g(x, \dots, x) = x$ for all $x \in D$;*
- cyclic *if $m \geq 2$ and $g(x_1, x_2, \dots, x_m) = g(x_2, \dots, x_m, x_1)$ for all $x_1, \dots, x_m \in D$;*
- symmetric *if $m \geq 2$ and $g(x_1, x_2, \dots, x_m) = g(x_{\pi(1)}, x_{\pi(2)}, \dots, x_{\pi(m)})$ for all $x_1, \dots, x_m \in D$, and any permutation π on $[m]$;*
- Siggers *if $m = 4$ and $g(r, a, r, e) = g(a, r, e, a)$ for all $a, e, r \in D$.*

A fractional operation ω is said to be idempotent/cyclic/symmetric if all operations in $\mathrm{supp}(\omega)$ have the corresponding property.

Note, the Siggers operation is traditionally defined in the literature as an **idempotent** operation g satisfying $g(r, a, r, e) = g(a, r, e, a)$. Here we follow the terminology in [1] that does not require idempotency. (In [10] such an operation was called *quasi-Siggers*).

We can now formulate the dichotomy theorem.

▶ **Theorem 9.** *Let Γ be a language. If the core of Γ admits a cyclic fractional polymorphism then Γ is tractable [21, 6, 31]. Otherwise Γ is NP-hard [23].*

We will call languages Γ satisfying the condition of Theorem 9 *solvable*. The following equivalent characterizations of solvability are either known or can be easily be derived from previous work [28, 19, 22, 23] (see [20]):

▶ **Lemma 10.** *Let Γ be a language and $g \in \mathrm{supp}^{(1)}(\Gamma)$. The following conditions are equivalent:*
(a) *Γ is solvable.*
(b) *Γ admits a cyclic fractional polymorphism of some arity $m \geq 2$.*
(c) *$\mathrm{supp}(\Gamma)$ contains a Siggers operation.*
(d) *$\Gamma \cup \{u_a \mid a \in B\}$ is solvable for any core B of Γ.*
(e) *$\Gamma[g(D_\Gamma)]$ is solvable.*
Furthermore, a finite-valued language Γ is solvable if and only if it admits a symmetric fractional polymorphism of arity 2.

Note that checking solvability of a given language Γ is a decidable problem. Indeed, condition (c) can be tested by solving a linear program with $|\mathcal{O}_D^{(4)}| = |D|^{|D|^4}$ variables and $O(poly(\texttt{size}(\Gamma)))$ constraints, where we maximize the total weight of Siggers operations subject to linear constraints expressing that $\omega \in \mathbb{R}^{\mathcal{O}_D^{(4)}}$ is a fractional polymorphism of Γ of arity 4.

2.3 Basic LP relaxation

Symmetric operations are known to be closely related to LP-based algorithms for CSP-related problems. One algorithm in particular has been known to solve many VCSPs to optimality. This algorithm is based on the so-called *basic LP relaxation*, or BLP, defined as follows.

Let $\mathbb{M}_n = \{\mu \geq 0 \mid \sum_{x \in D^n} \mu(x) = 1\}$ be the set of probability distributions over labelings in D^n. We also denote $\Delta = \mathbb{M}_1$; thus, Δ is the standard $(|D|-1)$-dimensional simplex. The corners of Δ can be identified with elements in D. For a distribution $\mu \in \mathbb{M}_n$ and a variable $v \in \{1, \dots, n\}$, let $\mu_{[v]} \in \Delta$ be the marginal probability of distribution μ for v:

$$\mu_{[v]}(a) \;=\; \sum_{x \in D^n : x_v = a} \mu(x) \qquad \forall a \in D.$$

Given a VCSP instance $\mathcal{I}$ in the form (1), we define the value $\mathrm{BLP}(\mathcal{I})$ as follows:

$$\begin{aligned} \mathrm{BLP}(\mathcal{I}) \;=\; & \min_{\mu,\alpha} \sum_{t \in T} \sum_{x \in \mathtt{dom}\, f_t} \mu_t(x) f_t(x) && (3)\\ \text{s.t.}\;\; (\mu_t)_{[k]} &= \alpha_{v(t,k)} && \forall t \in T, k \in \{1, \dots, n_t\}\\ \mu_t &\in \mathbb{M}_{n_t} && \forall t \in T\\ \mu_t(x) &= 0 && \forall t \in T, x \notin \mathtt{dom}\, f_t\\ \alpha_v &\in \Delta && \forall v \in V \end{aligned}$$

If there are no feasible solutions then $\mathrm{BLP}(\mathcal{I}) = \infty$. The objective function and all constraints in this system are linear, therefore this is a linear program. Its size is polynomial in $\mathtt{size}(\mathcal{I})$, so $\mathrm{BLP}(\mathcal{I})$ can be found in time polynomial in $\mathtt{size}(\mathcal{I})$.

We say that BLP *solves* $\mathcal{I}$ if $\mathrm{BLP}(\mathcal{I}) = \min_{x \in D^n} f_{\mathcal{I}}(x)$, and BLP solves $\mathrm{VCSP}(\Gamma)$ if it solves all instances $\mathcal{I}$ of $\mathrm{VCSP}(\Gamma)$. The following results are known.

▶ **Theorem 11** ([22]). *(a) BLP solves* $\mathrm{VCSP}(\Gamma)$ *if and only if* Γ *admits a symmetric fractional polymorphism of every arity* $m \geq 2$*. (b) If* Γ *is finite-valued then BLP solves* $\mathrm{VCSP}(\Gamma)$ *if and only if* Γ *admits a symmetric fractional polymorphism of arity* 2 *(i.e. if it is solvable).*

BLP relaxation also plays a key role for general-valued languages, as the following result shows. Recall that u_A for a subset $A \subseteq D$ is the unary function $D \to \{0, \infty\}$ with $\mathtt{dom}\, u_A = A$.

▶ **Definition 12.** *Consider instance* $\mathcal{I}$ *with the set of variables* V *and domain* D*. For node* $v \in V$ *denote* $D_v = \{a \in D \mid \exists x \in D^V \text{ s.t. } f_{\mathcal{I}}(x) < \infty, x_v = a\}$*. We define* $\mathrm{Feas}(\mathcal{I})$ *and* $\mathcal{I} + \mathrm{Feas}(\mathcal{I})$ *to be the instances with variables* V *and the following objective functions:*

$$f_{\mathrm{Feas}(\mathcal{I})}(x) = \sum_{v \in V} u_{D_v}(x_v) \qquad\qquad f_{\mathcal{I}+\mathrm{Feas}(\mathcal{I})}(x) = f_{\mathcal{I}}(x) + f_{\mathrm{Feas}(\mathcal{I})}(x)$$

It is easy to see that $f_{\mathcal{I}}(x) = f_{\mathcal{I}+\mathrm{Feas}(\mathcal{I})}(x)$ for any $x \in D^V$. However, the BLP relaxations of these two instances may differ.

▶ **Theorem 13** ([21]). *If* Γ *is solvable and* $\mathcal{I}$ *is a* Γ*-instance then BLP solves* $\mathcal{I} + \mathrm{Feas}(\mathcal{I})$*.*

If Γ is solvable and we know a core B of Γ, then an optimal solution for every Γ-instance can be found by using the standard self-reducibility method, in which we repeatedly add unary terms of the form $u_a(x_v)$ to the instance for different $v \in V$ and $a \in B$ and check whether this changes the optimal value of the BLP relaxation. A formal description of the method is given below. (Notations $\mathcal{I}[B]$ and $\mathcal{I} + u_a(x_v)$ should be self-explanatory; in particular, the former is the instance obtained from $\mathcal{I}$ by restricting each term to domain B).

Algorithm 1: `LP-Probe`$(\mathcal{I}, B)$. Input: instance $\mathcal{I}$ with variables V and domain D, set $B \subseteq D$. Output: either a labeling $x^* \in \arg\min_{x \in D^V} f_{\mathcal{I}}(x)$ with $x^* \in B^V$ or a flag in $\{\varnothing, \texttt{FAIL}\}$.

1 compute value $LP^* = BLP(\mathcal{I}+\text{Feas}(\mathcal{I}))$, then update $\mathcal{I} \leftarrow \mathcal{I}[B]$ (or return $\varnothing$ if $LP^* = \infty$)
2 **for** *each variable* $v \in V$ *in some order* **do**
3 **for** *each label* $a \in B$ *in some order* **do**
4 let $\mathcal{I}' = \mathcal{I} + u_a(x_v)$, and compute $LP' = BLP(\mathcal{I}' + \text{Feas}(\mathcal{I}'))$
5 if $LP' = LP^*$ then update $\mathcal{I} \leftarrow \mathcal{I}'$ and go to line 2 (i.e. proceed with the next variable v)
6 return `FAIL`
7 return labeling $x^* \in B^V$ where x^*_v equals the label a for which term $u_a(x_v)$ has been added

▶ **Lemma 14.**

(a) *If* `LP-Probe`$(\mathcal{I}, B)$ *returns a labeling* x^* *then* $x^* \in \arg\min_{x \in D^V} f_{\mathcal{I}}(x)$.

(b) *If* `LP-Probe`$(\mathcal{I}, B)$ *returns* $\varnothing$ *then instance* $\mathcal{I}$ *is infeasible.*

(c) *Suppose that* $\mathcal{I}$ *is a* Γ*-instance where* Γ *is solvable and* $B \in \mathcal{B}_\Gamma^{core}$. *Then* `LP-Probe`$(\mathcal{I}, B) \neq$ `FAIL`.

Proof. Part (a) holds by construction, and part (b) can be derived from the following two facts (which hold under the preconditions of part (b)):

- $\min_{x \in B^V} f_{\mathcal{I}}(x) = \min_{x \in D^V} f_{\mathcal{I}}(x)$ by Lemma 7(c).
- BLP solves all instances to which it is applied during the algorithm. Indeed, by Lemma 10 the language $\Gamma' = \Gamma[B] \cup \{u_a \mid a \in B\}$ is solvable. The initial instance is a Γ-instance, and all instances in line 4 are Γ'-instances. The claim now follows from Theorem 13. ◀

2.4 Meta-questions and uniform algorithms

In the light of the previous discussion, it is natural to ask the following questions about a given language Γ: (i) Is Γ solvable? (ii) Is Γ a core? (iii) What is a core of Γ? Such questions are usually called *meta-questions* or *meta-problems* in the literature. For finite-valued languages their computational complexity has been studied in [29].

▶ **Theorem 15** ([29]). *Problems (i) and (ii) for* $\{0,1\}$*-valued languages are NP-complete and co-NP-complete, respectively.*

▶ **Theorem 16** ([29]). *There is a polynomial-time algorithm that, given a core finite-valued language* Γ*, either finds a binary idempotent symmetric fractional polymorphism* ω *of* Γ *with* $|\text{supp}(\omega)| = O(poly(\texttt{size}(\Gamma)))$*, or asserts that none exists.*

For crisp languages the following hardness results are known.

▶ **Theorem 17** ([14]). *Deciding whether a given crisp language* Γ *with a single binary relation is a core is a co-NP-complete problem. (Equivalently, testing whether a given directed graph has a non-bijective homomorphism onto itself is an NP-complete problem).*

▶ **Theorem 18** ([10]). *Deciding whether a given crisp language* Γ *with binary relations is solvable is an NP-complete problem.*

It is still an open question whether an analogue of Theorem 16 holds for crisp languages, i.e. whether solvability of a given core crisp language Γ can be tested in polynomial time. However, it is known [10] that the answer would be positive assuming the existence of a certain *uniform* polynomial-time algorithm for CSPs.

▶ **Definition 19.** *Let $\mathcal{F}$ be a class of languages. A uniform polynomial-time algorithm for $\mathcal{F}$ is a polynomial-time algorithm that, for each input $(\Gamma, \mathcal{I})$ with $\Gamma \in \mathcal{F}$ and $\mathcal{I} \in \mathrm{VCSP}(\Gamma)$, computes $\min_x f_{\mathcal{I}}(x)$.*

▶ **Theorem 20** ([10])**.** *Suppose that there exists a uniform polynomial-time algorithm for the class of solvable core crisp languages. Then there exists a polynomial-time algorithm that decides whether a given core crisp language is solvable (or equivalently admits a Siggers polymorphism).*

Currently it is not known whether a uniform polynomial-time algorithm for core crisp languages exists. (Algorithms in [6, 31] assume that needed polymorphisms of the language are part of the input; furthermore, the worst-case bound on the runtime is exponential in $|D|$).

We remark that [10] considered a wider range of meta-questions for crisp languages. In particular, they studied the complexity of deciding whether a given Γ admits polymorphism $g \in O_D^{(m)}$ satisfying a given *strong linear Maltsev condition* specified by a set of linear identities. Examples of such identities are $g(x, \dots, x) \approx x$ (meaning that g is idempotent), $g(x_1, x_2, \dots, x_m) \approx g(x_2, \dots, x_m, x_1)$ (meaning that g is cyclic), and $g(r, a, r, e) \approx g(a, r, e, a)$ (meaning that g is Siggers). We refer to [10] for further details.

3 Our results

In this section the domain of language Γ is always denoted as D, and its size as $d = |D|$.

Our algorithms will construct Γ-instances $\mathcal{I}$ on $n = d^m$ variables (where $m \le 4$) with $\mathtt{size}(\mathcal{I}) = O(poly(\mathtt{size}(\Gamma)))$ for some fixed polynomial. We denote $T_{n,\Gamma}$ to be the running time of a procedure that computes $\mathrm{Feas}(\mathcal{I})$ for such $\mathcal{I}$'s. Also, let $T^*_{n,\Gamma}$ be the combined running times of computing $\mathrm{Feas}(\mathcal{J})$ for instances $\mathcal{J}$ during a call to $\mathtt{LP\text{-}Probe}(\mathcal{I}, B)$ for such $\mathcal{I}$ and some subset B. Note, if Γ is finite-valued then computing $\mathrm{Feas}(\mathcal{I})$ is a trivial problem, so $T_{n,\Gamma}$ and $T^*_{n,\Gamma}$ would be polynomial in $n + \mathtt{size}(\Gamma)$.

Conditional cores. First, we consider the problem of computing a core $B \in \mathcal{B}_\Gamma^{\mathtt{core}}$ of a given language Γ. A naive solution is to solve a linear program with $|\mathcal{O}^{(1)}| = d^d$ variables. We will present an alternative technique that runs more efficiently (in the case of finite-valued languages) but is allowed to output an incorrect result if Γ is not solvable. It will be convenient to introduce the following terminology: language Γ is a *conditional core* if either Γ is a core or Γ is not solvable. Similarly, set B is a *conditional core of* Γ if either $B \in \mathcal{B}_\Gamma^{\mathtt{core}}$ or Γ is not solvable. Note, $B = \varnothing$ is a conditional core of Γ if and only if Γ is not solvable.

To compute a conditional core of Γ, we will use the following approach. Consider a pair (Γ, σ) where σ is a string of size $O(poly(\Gamma))$ that specifies set $\mathcal{B}_\sigma$ of candidate cores of Γ. Formally, $\mathcal{B}_\sigma = \{B_1, \dots, B_N\}$ where $\varnothing \ne B_i \subseteq D$ for each $i \in [N]$. We assume that elements of $\mathcal{B}_\sigma$ can be efficiently enumerated, i.e. there exists a polynomial-time procedure for computing B_1 from σ and B_{i+1} from (σ, B_i). If $\mathcal{B}$ is a set of subsets $B \subseteq D$, we will denote

$$\begin{aligned} \mathcal{O}[\mathcal{B}] &= \{g \in \mathcal{O}^{(1)} \mid g(D) = B \text{ for some } B \in \mathcal{B}\} \\ \widehat{\mathcal{O}}[\mathcal{B}] &= \{g \in \mathcal{O}^{(1)} \mid g(D) \subseteq B \text{ for some } B \in \mathcal{B}\} \end{aligned}$$

▶ **Theorem 21.** *There exists an algorithm that for a given input* (Γ, σ) *does one of the following:*

(a) *Produces a fractional polymorphism* $\omega \in \mathrm{fPol}^{(1)}(\Gamma)$ *with* $\mathrm{supp}(\omega) \subseteq \widehat{\mathcal{O}}[\mathcal{B}_\sigma]$ *and* $|\mathrm{supp}(\omega)| \le 1 + \sum_{f\in\Gamma} |\mathtt{dom}\, f|$.

(b) *Asserts that there exists no vector* $\omega \in \mathrm{fPol}^{(1)}(\Gamma)$ *with* $\mathrm{supp}(\omega) \subseteq \widehat{\mathcal{O}}[\mathcal{B}_\sigma]$.

(c) *Asserts that one of the following holds: (i)* Γ *is not solvable; (ii)* $\mathcal{B}_\sigma \cap \mathcal{B}_\Gamma^{core} = \varnothing$.

It runs in $(|\mathcal{B}_\sigma| + O(poly(\mathtt{size}(\Gamma)))) \cdot (T^*_{d,\Gamma} + O(poly(\mathtt{size}(\Gamma))))$ *time and uses* $O(poly(\mathtt{size}(\Gamma)))$ *space.*

The algorithm in the theorem above is based on the ellipsoid method [13], which tests feasibility of a polytope using a polynomial number of calls to the separation oracle. In our case this oracle is implemented via one or more calls to $\mathtt{LP\text{-}Probe}(\mathcal{I}, B)$ for appropriate $\mathcal{I}$ and B.

One possibility would be to use Theorem 21 with the set $\mathcal{B}_\sigma = \{B \subseteq D \mid B \neq \varnothing, B \neq D\}$. If the algorithm returns result (a) then we can take operation $g \in \mathrm{supp}(\omega)$ and call the algorithm recursively for the language $\Gamma[g(D)]$ on a smaller domain. If we get result (b) or (c) then one can show that Γ is a conditional core, so we can stop. For finite-valued languages this approach would run in $O(2^d \cdot poly(\mathtt{size}(\Gamma)))$ time. We will pursue an alternative approach with an improved complexity $O(\sqrt[3]{3}^{\,d} \cdot poly(\mathtt{size}(\Gamma)))$.

This approach will use partitions $\Pi = \{D_1, \dots, D_k\}$ of domain D. For such Π we denote

$$\begin{aligned} \mathcal{O}_\Pi &= \{g \in \mathcal{O}_D^{(1)} \,:\, g(a) = g(b) \qquad \forall a, b \in A \in \Pi\} \\ \Pi^\perp &= \{B \subseteq D \,:\, |B \cap A| = 1 \qquad \forall A \in \Pi\} \end{aligned}$$

We say that Π is a *partition of* Γ if the set $\mathcal{O}_\Pi \cap \mathrm{supp}(\Gamma)$ is non-empty. In particular, the partition $\Pi = \{\{a\} \mid a \in D\}$ of D into singletons is a partition of Γ, since $\mathrm{supp}(\Gamma)$ contains the identity mapping $D \to D$. We say that Π is a *maximal partition of* Γ if Π is a partition of Γ and no coarser partition $\Pi' \succ \Pi$ (i.e. Π' with $\mathcal{O}_{\Pi'} \subset \mathcal{O}_\Pi$) has this property. Clearly, for any Γ there exists at least one Π which is a maximal partition of Γ. By analogy with cores, we say that Π is a *conditional (maximal) partition of* Γ if either Π is a (maximal) partition of Γ or Γ is not solvable.

In the results below Π is always assumed to be a partition of D.

▶ **Lemma 22.**

(a) *If* Π *is a maximal partition of* Γ *then* $\mathcal{B}_\Gamma^{core} \subseteq \Pi^\perp$ *and* $\mathcal{O}_\Gamma^{core} = \mathcal{O}[\Pi^\perp] \cap \mathrm{supp}(\Gamma)$.

(b) *If* Π *is a partition of* D *then* $|\Pi^\perp| \le \sqrt[3]{3}^{\,d}$.

▶ **Theorem 23.** *There exists an algorithm with runtime* $T_{d,\Gamma} + T_{|\Pi|,\Gamma} + O(poly(\mathtt{size}(\Gamma)))$ *that for a given input* (Γ, Π) *does one of the following:*

(a) *Asserts that* Π *is a conditional partition of* Γ.

(b) *Asserts that* Π *is not a partition of* Γ.

As before, the algorithm in Theorem 23 is based on the ellipsoid method. However, now we cannot use procedure $\mathtt{LP\text{-}Probe}(\mathcal{I}, B)$ to implement the separation oracle, since a candidate core B is not available. Instead, we solve the BLP relaxation of instance $\mathcal{I}$ and derive a separating hyperplane from a (fractional) optimal solution of the relaxation.

▶ **Corollary 24.** (1) *A conditional maximal partition* Π *of* Γ *can be computed in* $O(d^2) \cdot T_{d,\Gamma} + O(poly(\mathtt{size}(\Gamma)))$ *time.* (2) *Once such* Π *is found, a conditional core* B *of* Γ *can be computed using* $(|\Pi^\perp| + O(poly(\mathtt{size}(\Gamma)))) \cdot (T^*_{d,\Gamma} + O(poly(\mathtt{size}(\Gamma))))$ *time and* $O(poly(\mathtt{size}(\Gamma)))$ *space. If* $B \neq \varnothing$ *then the algorithm also produces a fractional polymorphism* $\omega \in \mathrm{fPol}(\Gamma)$ *such that* $\mathrm{supp}(\omega) \subseteq \mathcal{O}[\Pi^\perp]$, $|\mathrm{supp}(\omega)| \le 1 + \sum_{f\in\Gamma} |\mathtt{dom}\, f|$ *and* $\mathrm{supp}(\omega)$ *contains an operation* g *with* $g(D) = B$.

In part (1) we use a greedy search that starts with $\Pi = \{\{a\} \mid a \in D\}$ and then repeatedly calls the algorithm in Theorem 23 for coarser partitions Π. In part (2) we call the algorithm from Theorem 21 with $\sigma = \Pi$ and $\mathcal{B}_\sigma = \Pi^\perp$. For further details we refer to [20].

Testing solvability of a conditional core. Once we have found a conditional core B of Γ, we need to test whether language $\Gamma[B]$ is solvable. This problem is known to be solvable in polynomial-time for finite-valued languages [29], see Theorem 16. Their result can be extended as follows.

▶ **Theorem 25.** *There exists an algorithm that for a given language Γ does one of the following:*

(a) *Produces an idempotent fractional polymorphism $\omega \in \mathrm{fPol}(\Gamma)$ certifying solvability of Γ:*
- *ω has arity $m = 2$ and is symmetric, if Γ is finite-valued;*
- *ω has arity $m = 4$ and contains a Siggers operation in the support, if Γ is not finite-valued. Furthermore, in each case vector ω satisfies $|\operatorname{supp}(\omega)| \le 1 + \sum_{f\in\Gamma} \binom{|\mathtt{dom}\, f|}{m}$.*

(b) *Asserts that one of the following holds: (i) Γ is not solvable; (ii) Γ is not a core.*

Its runtime is $O(poly(\mathtt{size}(\Gamma)))$ if Γ is finite-valued, and $O(T^*_{d^4,\Gamma} \cdot poly(\mathtt{size}(\Gamma)))$ otherwise.

Combining procedures in Corollary 24 and the algorithm in Theorem 25 yields our main algorithmic result.

▶ **Corollary 26.** *Solvability of a given finite-valued language Γ can be tested in $O(\sqrt[3]{3}^{\,d} \cdot poly(\mathtt{size}(\Gamma)))$ time. If the answer is positive, the algorithm also returns a fractional polymorphism $\omega_1 \in \mathrm{fPol}^{(1)}(\Gamma)$ with $\operatorname{supp}(\omega_1) \subseteq \mathcal{O}_\Gamma^{\mathtt{core}}$ and a symmetric idempotent fractional polymorphism $\omega_2 \in \mathrm{fPol}^{(2)}(\Gamma[B])$ where $B = g(D)$ for some $g \in \operatorname{supp}(\omega_1)$; furthermore, $|\operatorname{supp}(\omega_m)| \le 1 + \sum_{f\in\Gamma} \binom{|\mathtt{dom}\, f|}{m}$ for $m \in \{1, 2\}$.*

Hardness results. Let us fix a constant $L \in \{1, \infty\}$. As Theorems 15, 17 and 18 state, testing whether Γ is (i) solvable and (ii) is a core are both NP-hard problems for $\{0, L\}$-valued languages. We now present additional hardness results under the Exponential Time Hypothesis (ETH) and the Strong Exponential Time Hypothesis (SETH) (see Conjectures 1 and 2). Note that for Theorem 27 we simply reuse the reductions from [10]. We say that a family of languages $\mathcal{F}$ is *k-bounded* if each $\Gamma \in \mathcal{F}$ satisfies $\mathtt{size}(\Gamma) = O(poly(d))$ for some fixed polynomial, and $\mathtt{arity}(f) \le k$ for all $f \in \Gamma$.

▶ **Theorem 27.** *Suppose that ETH holds. Then there exists a 2-bounded family $\mathcal{F}$ of $\{0, L\}$-valued languages such that the following problems cannot be solved in $O(2^{o(d)})$ time:*

(a) *Deciding whether language $\Gamma \in \mathcal{F}$ is solvable.*

(b) *Deciding whether language $\Gamma \in \mathcal{F}$ is a core.*

▶ **Theorem 28.** *Suppose that SETH holds. Then for any $\delta < 1$ there exists an $O(1)$-bounded family $\mathcal{F}$ of $\{0, L\}$-valued languages such that the following problems cannot be solved in $O(\sqrt[3]{3}^{\,\delta d})$ time:*

(a) *Deciding whether language $\Gamma \in \mathcal{F}$ is solvable (assuming the existence of a uniform polynomial-time algorithm for core crisp languages, in the case when $L = \infty$).*

(b) *Deciding whether language $\Gamma \in \mathcal{F}$ satisfies $\mathtt{core\text{-}size}(\Gamma) \le d/3$.*

References

1. L. Barto, A. Krokhin, and R. Willard. Polymorphisms, and how to use them. In A. Krokhin and S. Živný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*. Dagstuhl Follow-Ups series, Volume 7, 2017.
2. Libor Barto. The dichotomy for conservative constraint satisfaction problems revisited. In *Proceedings of the 26th IEEE Symposium on Logic in Computer Science (LICS'11)*, pages 301–310. IEEE Computer Society, 2011. doi:10.1109/LICS.2011.25.
3. Libor Barto, Marcin Kozik, and Todd Niven. The CSP dichotomy holds for digraphs with no sources and no sinks (a positive answer to a conjecture of Bang-Jensen and Hell). *SIAM Journal on Computing*, 38(5):1782–1802, 2009. doi:10.1137/070708093.
4. A. Björklund, T. Husfeldt, and M. Koivisto. Set partitioning via inclusion–exclusion. *SIAM J. Computing*, 39(2):546–563, 2009.
5. Andrei Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *Journal of the ACM*, 53(1):66–120, 2006. doi:10.1145/1120582.1120584.
6. Andrei Bulatov. A dichotomy theorem for nonuniform CSPs. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS'17)*. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.37.
7. Andrei Bulatov, Andrei Krokhin, and Peter Jeavons. Classifying the Complexity of Constraints using Finite Algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005. doi:10.1137/S0097539700376676.
8. Andrei A. Bulatov. Complexity of conservative constraint satisfaction problems. *ACM Transactions on Computational Logic*, 12(4), 2011. Article 24. doi:10.1145/1970398.1970400.
9. Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In *Proceedings of the 4th International Workshop on Parameterized and Exact Computation (IWPEC)*, volume 5917 of *LNCS*, pages 75–85, 2009.
10. Hubie Chen and Benoit Larose. Asking the Metaquestions in Constraint Tractability. *ACM Transactions on Computation Theory (TOCT)*, 9(3), October 2017. doi:10.1145/3134757.
11. Tomás Feder and Moshe Y. Vardi. The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study through Datalog and Group Theory. *SIAM Journal on Computing*, 28(1):57–104, 1998. doi:10.1137/S0097539794266766.
12. Fedor V. Fomin, Alexander Golovnev, Alexander S. Kulikov, and Ivan Mihajlin. Lower Bounds for the Graph Homomorphism Problem. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 9134 of *LNCS*. Springer, 2015.
13. Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*. Springer, 1988.
14. P. Hell and J. Nešetřil. The core of a graph. *Discrete Mathematics*, 109(1-3):117–126, 1992.
15. Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.
16. Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.
17. P. Jonsson, V. Lagerkvist, G. Nordh, , and B. Zanuttini. Strong partial clones and the time complexity of SAT problems. *Journal of Computer and System Sciences*, 84:52–78, 2017.
18. Peter Jonsson, Victor Lagerkvist, and Biman Roy. Time Complexity of Constraint Satisfaction via Universal Algebra. In *Proceedings of the 42nd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, 2017.
19. Keith Kearnes, Petar Marković, and Ralph McKenzie. Optimal strong Mal'cev conditions for omitting type 1 in locally finite varieties. *Algebra Universalis*, 72(1):91–100, 2014.
20. Vladimir Kolmogorov. Testing the complexity of a valued CSP language. arXiv, 2019. arXiv:1803.02289v3.
21. Vladimir Kolmogorov, Andrei Krokhin, and Michal Rolínek. The Complexity of General-Valued CSPs. *SIAM Journal on Computing (SICOMP)*, 46(3):1087–1110, 2017.

22 Vladimir Kolmogorov, Johan Thapper, and Stanislav Živný. The power of linear programming for general-valued CSPs. *SIAM Journal on Computing*, 44(1):1—-36, 2015.

23 Marcin Kozik and Joanna Ochremiak. Algebraic Properties of Valued Constraint Satisfaction Problem. arXiv, 2015. Extended abstract published in ICALP'15. arXiv:1403.0476.

24 Victor Lagerkvist and Magnus Wahlström. Which NP-Hard SAT and CSP Problems Admit Exponentially Improved Algorithms? arXiv, 2018. arXiv:1801.09488.

25 E.L. Lawler. A note on the complexity of the chromatic number problem. *Inf. Process. Lett.*, 5(3):66–67, 1976.

26 Igor Razgon. Complexity Analysis of Heuristic CSP Search Algorithms. In *International Workshop on Constraint Solving and Constraint Logic Programming (CSCLP)*, volume 3978 of *LNCS*, pages 88–99, 2005.

27 Thomas J. Schaefer. The Complexity of Satisfiability Problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC'78)*, pages 216–226. ACM, 1978. doi:10.1145/800133.804350.

28 Mark H. Siggers. A strong Mal'cev condition for locally finite varieties omitting the unary type. *Algebra Universalis*, 64(1-2):15–20, 2010.

29 Johan Thapper and Stanislav Živný. The complexity of finite-valued CSPs. *Journal of the ACM (JACM)*, 63(4), 2016.

30 Patrick Traxler. The time complexity of constraint satisfaction. In *Proceedings of the 3rd International Workshop on Parameterized and Exact Computation (IWPEC)*, pages 190–201, 2008.

31 Dmitriy Zhuk. A Proof of CSP Dichotomy Conjecture. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS'17)*. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.38.

Towards Optimal Depth Reductions for Syntactically Multilinear Circuits

Mrinal Kumar
University of Toronto, Canada
https://mrinalkr.bitbucket.io/
mrinalkumar08@gmail.com

Rafael Oliveira
University of Toronto, Canada
http://www.cs.utoronto.ca/~rafael/
rafael@cs.toronto.edu

Ramprasad Saptharishi
Tata Institute of Fundamental Research
https://www.tcs.tifr.res.in/~ramprasad/
ramprasad@tifr.res.in

Abstract

We show that any n-variate polynomial computable by a syntactically multilinear circuit of size $\text{poly}(n)$ can be computed by a depth-4 syntactically multilinear ($\Sigma\Pi\Sigma\Pi$) circuit of size at most $\exp\left(O\left(\sqrt{n \log n}\right)\right)$. For degree $d = \omega(n/\log n)$, this improves upon the upper bound of $\exp\left(O(\sqrt{d} \log n)\right)$ obtained by Tavenas [14] for general circuits, and is known to be asymptotically optimal in the exponent when $d < n^{\varepsilon}$ for a small enough constant ε. Our upper bound matches the lower bound of $\exp\left(\Omega\left(\sqrt{n \log n}\right)\right)$ proved by Raz and Yehudayoff [12], and thus cannot be improved further in the exponent. Our results hold over all fields and also generalize to circuits of small individual degree.

More generally, we show that an n-variate polynomial computable by a syntactically multilinear circuit of size $\text{poly}(n)$ can be computed by a syntactically multilinear circuit of product-depth Δ of size at most $\exp\left(O\left(\Delta \cdot (n/\log n)^{1/\Delta} \cdot \log n\right)\right)$. It follows from the lower bounds of Raz and Yehudayoff [12] that in general, for constant Δ, the exponent in this upper bound is tight and cannot be improved to $o\left((n/\log n)^{1/\Delta} \cdot \log n\right)$.

2012 ACM Subject Classification Theory of computation → Circuit complexity

Keywords and phrases arithmetic circuits, multilinear circuits, depth reduction, lower bounds

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.78

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at https://arxiv.org/abs/1902.07063.

Funding *Mrinal Kumar*: A part of this work was done during the postdoctoral stay at Harvard, during the lower bounds semester at Simons Institute for the Theory of Computing, Berkeley and while visiting TIFR, Mumbai.
Rafael Oliveira: Part of this work was done while visiting the Simons Institute for the Theory of Computing.
Ramprasad Saptharishi: Research supported by Ramanujan Fellowship of DST.

Acknowledgements We are extremely thankful to Ben Rossman, who pointed us towards this question, and for many stimulating discussions at various stages of this work. We also thank Shubhangi Saraf, Amir Shpilka and Ben Lee Volk for many helpful conversations. Mrinal also thanks Prahladh Harsha for accommodating him in his apartment for a part of the visit to TIFR, where a part of this paper was written.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi; Article No. 78; pp. 78:1–78:15

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

1 Introduction

An algebraic circuit over a field $\mathbb{F}$ and variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is a directed acyclic graph whose internal vertices (called gates) are labeled as either $+$ (sum) or $\times$ (product), and leaves (vertices of indegree zero) are labeled by the variables in $\mathbf{x}$ or constants from $\mathbb{F}$. The gates of outdegree zero in a circuit are called its output gates. Algebraic circuits give a natural and succinct representation for multivariate polynomials; analogous to the way Boolean circuits give a succinct representation of Boolean functions. We refer the reader to the excellent survey of Shpilka and Yehudayoff [13] for an introduction to the area of algebraic circuit complexity. One of the main protagonists in the results in this paper will be the class of syntactically multilinear circuits which we now define.

▶ **Definition 1** (Syntactically Multilinear Circuits)**.** *An algebraic circuit C is said to be syntactically multilinear if at every product gate v in C with inputs $u_1, u_2, \ldots, u_t$, the set of variables in the sub-circuits rooted at u_i are pairwise disjoint from each other.*

The size of an algebraic circuit is the number of edges in it, and its depth is the length of the longest path from an output gate to a leaf. Intuitively, the size of a circuit is an indicator of the time complexity of computing the polynomial, and its depth indicates how fast the polynomial can be computed in parallel.

We now introduce a sequence of fundamental structural results for algebraic circuits, that are collectively called depth reductions; this is the main focus of this paper.

Depth Reductions

In a beautiful, surprising and influential work, Valiant et al. [15] showed that every polynomial family which is efficiently computable by an algebraic circuit is also efficiently computable in parallel. Formally, they showed the following theorem.

▶ **Theorem 2** ([15])**.** *There is an absolute constant $c \in \mathbb{N}$ such that the following is true. If P be an n-variate homogeneous polynomial of degree d over any field $\mathbb{F}$ which can be computed by an algebraic circuit C of size s, then P can be computed by an algebraic circuit C' (of unbounded fan-in) of depth $c \log d$ and size $(snd)^c$.*

In particular, the theorem says that every polynomial family of polynomially bounded (in n) degree that is computable by a circuit of size $\mathrm{poly}(n)$ and arbitrary depth, is also efficiently computable by a circuit of size $\mathrm{poly}(\log n)$ and depth $O(\log n)$.

In a remarkable extension of Theorem 2, Agrawal and Vinay [1] showed that one can parallelize algebraic circuits even more (reducing the depth to a constant), at the cost of a larger (a non-trivial subexponential factor) blow up in the circuit size. The version of their theorem stated below is due to Tavenas [14], who optimized the parameters further.

▶ **Theorem 3** ([1, 8, 14])**.** *There is an absolute constant $c \in \mathbb{N}$ such that the following is true. If P is an n-variate homogeneous polynomial of degree d over any field $\mathbb{F}$ which can be computed by an algebraic circuit C of size s, then P can be computed by a homogeneous $\Sigma\Pi\Sigma\Pi$ algebraic circuit C' of size $(snd)^{c\sqrt{d}}$.*

Here, a $\Sigma\Pi\Sigma\Pi$ circuit is an algebraic circuit with four layers of alternating sum and product gates with the top layer being a sum layer. Throughout this paper, when we say a depth-4 circuit, we mean a $\Sigma\Pi\Sigma\Pi$ circuit.

We note that while Theorem 3 as stated above reduces a homogeneous circuit of arbitrary depth to a homogeneous circuit of depth-4, but it easily follows from the proof that the depth reduction preserves syntactic restrictions. That is, if we start with a syntactically multilinear

and homogeneous circuit, the resulting depth-4 circuit is also syntactically multilinear and homogeneous. This statement will be of particular interest as we study depth reductions for syntactically multilinear circuits in this paper.

On the optimality of reductions to depth-4

An immediate consequence of Theorem 2 and Theorem 3 is that strong enough lower bounds for algebraic circuits of bounded depth imply superpolynomial lower bounds for general algebraic circuits. Thus, the questions of proving lower bounds for bounded depth circuits, and that of understanding if the parameters in Theorem 3 can be improved further seem to be of fundamental interest. In the last few years, we have had significant progress on both these fronts. Following a long line of work starting with a work of Kayal [7] and Gupta et al. [5], we now know extremely good lower bounds for homogeneous depth-4 circuits.

▶ **Theorem 4** (Kumar and Saraf [9]). *There exists a polynomial family $\{f_n\}$, where f_n is a homogeneous n-variate polynomial of degree $d = n^{\varepsilon}$, for an absolute constant $\varepsilon > 0$, such that f_n is computable by an algebraic circuit of size* $\mathrm{poly}(n)$*, but any homogeneous depth-4 circuit computing f_n has size $n^{\Omega(\sqrt{d})}$.*

Moreover, the family $\{f_n\}$ is computable by a syntactically multilinear circuit of polynomial size.

If we allow the hard polynomial to be explicit but not necessarily have small circuits, then upper bound on the degree d in the above theorem can be increased to as large as $n^{1-\varepsilon}$ for any constant $\varepsilon > 0$.[1] Thus, in general, the exponent in the upper bound on the size of the depth-4 circuit obtained in Theorem 3 cannot be improved asymptotically. In fact, the theorem shows that we cannot even expect such an improvement for syntactically multilinear circuits in the setting when the degree d is sufficiently smaller than the number of variables n. A natural question here is to understand if Theorem 3 is also asymptotically tight in the exponent when the degree is larger. The following result of Raz and Yehudayoff goes a long way towards answering this question.

▶ **Theorem 5** ([12]). *There is a family of multilinear polynomials $\{f_n\}$ such that, for every n, the polynomial f_n is an n-variate degree $d = \Theta(n)$ polynomial that can be computed by a syntactically multilinear circuit of size* $\mathrm{poly}(n)$*, but any multilinear circuit of depth-4 computing f_n has size $n^{\Omega\left(\sqrt{n/\log n}\right)}$.*

More generally, for any constant Δ, any syntactically multilinear circuit of product-depth[2] *Δ computing f_n must have size $n^{\Omega\left((n/\log n)^{1/\Delta}\right)}$.*

For depth-4 circuits (or $\Delta = 2$), asimilar result was proved by Hegde and Saha [6] for the more general[3] class of circuits called multi-k-ic circuits, where the *formal degree* of any variable in the circuit is bounded by a parameter k (formally defined in Definition 17).

▶ **Theorem 6** ([6]). *There is an explicit family $\{f_n\}$ of n-variate multilinear polynomials of degree $d = \Theta(n)$ such that, for every $k \leq (n \log n)^{0.9}$, any multi-k-ic circuit of depth-4 computing f_n has size at least $n^{\Omega\left(\sqrt{n/(k \log n)}\right)}$.*

[1] Though this is not explicitly mentioned in these results, the proofs can be extended to this regime of parameters.
[2] Also referred to as a syntactically multilinear $(\Sigma\Pi)^{\Delta}$ circuit.
[3] A multilinear circuit is a multi-k-ic circuit for $k = 1$.

Thus, Theorem 5 and Theorem 6 shows that the exponent $\sqrt{d}$ in the exponent in Theorem 3 cannot be replaced by $o\left(\sqrt{n/\log n}\right)$. Thus, in the regime when $d = \Theta(n)$, there is a gap of $\sqrt{\log n}$ between the known lower bounds and what is potentially achievable via depth reduction. Raz and Yehudayoff [12] also observe that using their techniques, the lower bound cannot be improved to $n^{\omega(\sqrt{n/\log n})}$. Our main motivation for this work was to bridge this gap. In the light of Theorem 4, we believed the upper bound of $n^{O(\sqrt{d})}$ in Theorem 3 to be right bound for multilinear circuits for all d, and had hoped to improve the lower bound in Theorem 5 to $n^{\Omega(\sqrt{n})}$.

However, as we discuss next, the correct exponent for depth reduction to depth-4 in the high degree regime turns out to be $\sqrt{n/\log n}$. In addition to being surprising, this also offers a potentially viable approach to the question of proving superpolynomial lower bounds for syntactically multilinear circuits by extending Theorem 4 to the high degree regime. We now state our results and discuss the connections to multilinear circuit lower bounds.

1.1 Results

We start by stating our main theorems.

▶ **Theorem 7.** *Let C be a multi-k-ic circuit of size s computing a polynomial in n variables. Then, there is a multi-k-ic $\Sigma\Pi\Sigma\Pi$ circuit C' of size $s^{O\left(\sqrt{\frac{kn}{\log s}}\right)}$ computing the same polynomial.*

The ideas in the proof of Theorem 7 generalize to give the following statement about reduction to depth Δ circuits for any constant Δ.

▶ **Theorem 8.** *Let C be a multi-k-ic circuit of size s computing a polynomial in n variables. Then, there is a multi-k-ic $(\Sigma\Pi)^{\Delta}$ circuit C' computing the same polynomial whose size is at most*

$$s^{O\left(\Delta\cdot(nk/\log s)^{1/\Delta}\right)}.$$

Thus, for $s = \text{poly}(n)$, $k = o(\log s)$ and $n \geq d \geq \omega\left(\frac{kn}{\log s}\right)$, the exponents in the upper bounds in Theorem 7 are asymptotically better than that in Theorem 3. An immediate consequence of Theorem 7 is the following corollary.

▶ **Corollary 9.** *Let $\{f_n\}$ be an explicit family of multilinear polynomials, such that f_n is an n variate polynomial of degree $d = \omega(n/\log n)$, and any multilinear $\Sigma\Pi\Sigma\Pi$ circuit computing f_n has size at least $n^{\Omega(\sqrt{d})}$. Then, $\{f_n\}$ requires superpolynomial size syntactically multilinear circuits.*

The corollary is of interest since by Theorem 4, we know $n^{\Omega(\sqrt{d})}$ lower bounds for homogeneous multilinear $\Sigma\Pi\Sigma\Pi$ circuits, when $d = n^{\varepsilon}$. Thus extending these bounds so that they hold for higher degree polynomials will imply superpolynomial lower bounds for multilinear circuits. The current best lower bound known for multilinear circuits is a nearly quadratic lower bound in a recent work of Alon et al. [3]. The standard technique for proving lower bounds for multilinear models is via the rank of the *partial derivative matrix* under a random partition of variables (due to Raz [10]). This has been useful in almost all of the known lower bounds for multilinear models, such as super polynomial lower bounds for multilinear formulas [10], exponential lower bounds for constant depth multilinear circuits [12] as well as the currently known superlinear and nearly quadratic lower bounds for multilinear circuits [11, 3]. However, this technique is too weak to yield even super-cubic lower bounds for syntactically multilinear circuits. Thus, currently we do not even have potential approaches to proving superpolynomial

lower bounds for multilinear circuits. In the light of this, it certainly seems worth exploring if the partial derivative based methods used in the proof of Theorem 4 can be extended to work for multilinear polynomials whose degree $d = \omega(n/\log n)$ is high. As far as we understand, there does not seem to be strong evidence one way or the other about this.

For multi-k-ic circuits, we do not even know superpolynomial lower bounds for formulas or even constant depth formulas. Based on the discussion above, Theorem 7 does seem to offer a potentially viable approach to prove these lower bounds.

Finally, we note again that the upper bound on the size of the depth-4 circuit obtained in Theorem 7 cannot be further improved asymptotically in the exponent as Theorem 5 shows.

1.2 Proof Overview

We focus on giving an outline of the proof of Theorem 7 for the multilinear case (or $k = 1$). The proof follows the strategy of the proof of Theorem 3 with some key differences, which we point out as we go along. There are two main steps and we now give a sketch of both of them.

Balancing a syntactically multilinear circuit

For this step, the key notion is that of a *balanced* circuit. We say that a circuit C is balanced with respect to a potential function $\Phi : C \to \mathbb{N}$ (e.g. degree, number of variables), if the fan-in of every product g in C is a constant, and $\Phi(g) \geq 2\Phi(h)$ for every child h of g. In the proof of Theorem 3, the authors essentially use the results of Valiant et al. [15] to balance a homogeneous circuit with the potential function Φ being the formal degree of a gate. For our proof, we show that a syntactically multilinear circuit can in fact be balanced with the potential function being the number of variables in the sub-circuit rooted at a gate. Our proof of this part involves the machinery of *gate quotients* and *frontier decompositions* developed by Valiant et al. in their original proof, although there are some crucial differences which require some non-trivial (albeit simple) insights.

One such challenge stems from the fact that in a homogeneous circuit, the formal degree of any two children of a product gate is the same and equal to the formal degree of the parent, whereas the children might depend on very different (even completely disjoint) sets of variables. To get around this, our notion of *frontier* is different from that of Valiant et al [15]. In [15], frontier is defined with respect to vertices, whereas we define frontier with respect to edges. As a consequence, our frontier decomposition statements are slightly different from those in [15], although they continue to have a natural semantic meaning. This is detailed in Section 5.

Reduction to depth-4 from a balanced circuit

In the second part of our proof, we show that any balanced syntactically multilinear circuit of size s computing a polynomial in n variables can be depth reduced to a syntactically multilinear depth-4 circuit of size $s^{O(\sqrt{n/\log n})}$. The proof is along the lines of the proof of the analogous statement in the homogeneous (non-multilinear) setting by Chillara et al. [4]. The high level idea of the proof is the following : in a balanced circuit C, the polynomial computed at any gate g can be written as a sum of product of *terms*, where the product fan-in is a constant, the sum fan-in is upper bounded by the size of the circuit, and the number of variables in any of the terms is at most half of the number of variables in g. Moreover, each of the terms is a polynomial computed by a gate in C, so this decomposition can be recursively applied. We apply this decomposition repeatedly till every term in the sum of products expression of the output depends on at most t variables. We argue that the

sum fan-in of this sum of products expression is at most $s^{O(n/t)}$. Now, we expand each of the terms (which is a multilinear polynomial) as a sum of multilinear monomials in t variables. Thus, the total size of the $\Sigma\Pi\Sigma\Pi$ circuit obtained is $2^t \cdot s^{O(n/t)}$ which is $s^{O(\sqrt{n/\log s})}$ for $t = \sqrt{n \log s}$.

In the proof of the analogous statement for homogeneous non-multilinear circuits, at the end of the repeated applications of the decomposition, each of the terms is of degree at most t. Thus, a sum of product expansion of each such term has size $\binom{n}{t}$, and so the total size of the $\Sigma\Pi\Sigma\Pi$ circuit obtained is $n^t \cdot s^{O(n/t)}$, which for $s = \text{poly}(n)$ is minimized for $t = \sqrt{n}$ and equals $s^{O(\sqrt{n})}$. This explains the gain in the size obtained by Theorem 7.

2 Preliminaries

In this section, we describe the notion of parse-trees and gate quotients which are crucial to our proof and set up some of the machinery we need for the proof.

2.1 Parse-trees and quotients

▶ **Definition 10** (Parse-trees). *Let C be an algebraic circuit. For any $u_0 \in C$, a* parse-tree T rooted at u_0 *is a subcircuit of C that satifies the following properties:*

- *the node $u_0 \in T$,*
- *if $u \in T$ is a multiplication gate of C with $u = v_1 \times v_2$, then v_1, v_2 are also in T,*
- *if $u \in T$ is an addition gate of C with $u = v_1 + v_2$, then* exactly one *of v_1 or v_2 is in T.*

Any such sub-circuit computes just a monomial, and this shall be called the value the parse-tree. Although the parse-tree defined above need not be a tree, it shall unfolded to a tree.

If T is a parse-tree rooted at u, and v is a node that appears on its right-most path, then the tree T' obtained by replacing v only on the right-most path by a leaf labelled 1 is said to be a v-snipped parse-tree rooted at u.

▶ **Definition 11** (Var operator). *For any nodes $u \in C$, we denote by* $\text{Var}(u)$ *the vector $(d_1, \ldots, d_n) \in \mathbb{N}^n_{\geq 0}$ where d_i is the maximum x_i-degree over all parse-trees rooted at u.*

Similarly, for any pair of nodes $u, v \in C$, we denote by $\text{Var}(u : v)$ *the vector $(d_1, \ldots, d_n)$ where d_i is the maximum x_i-degree over all v-snipped parse-tree rooted at u.*

We shall also define $|(d_1, \ldots, d_n)| = \sum d_i$.

For a syntactically multilinear circuit C, note that $|\text{Var}(g)|$ for any gate $g \in C$ is precisely the number of distinct variables in the sub-circuit rooted at g.

▶ Remark 12. Throughout this discussion, we will assume that the circuit is *right heavy*. This means that for every multiplication gate, $w = w_L \times w_R$, $\text{Var}(w_R) \geq \text{Var}(w_L)$. Note that this is without loss of generality, since *left* and *right* are merely labels that we can assign arbitrarily to the children of every gate in the circuit.

▶ **Definition 13** (Gate Quotient). *For every two gates u, v in C, the* gate quotient *of u with respect to v, denoted by $[u : v]$ is defined inductively as follows.*

- *If $u = v$, then $[u : v] = 1$.*
- *If $u = u_1 + u_2$, then $[u : v] = [u_1 : v] + [u_2 : v]$.*
- *If $u = u_L \times u_R$, then $[u : v] = [u_L][u_R : v]$.*
- *If v does not appear in the subcircuit rooted at u, then $[u : v] = 0$.*

▶ **Lemma 14.** *Let $u, v \in C$. Then, the polynomial $[u]$ is the sum of values of all parse-trees rooted at u. Furthermore, the polynomial $[u : v]$ is the sum of the values of all the v-snipped parse-trees T that are rooted at u.*

The above lemma is almost folklore and a proof of it can be seen in the work of Allender et al. [2].

2.2 Syntactic restrictions on parse-trees

We remark that throughout this paper, by degree, we mean the syntactic or formal degree, which could be much larger than the actual or semantic degree. The following observation records some basic properties of the Var operator.

▶ **Observation 15.** *Let C be any algebraic circuit. Then,*

- $\mathrm{Var}(u)$ *is monotonically non-increasing as u moves towards the leaves. That is, if u is an ancestor of v, then ever coordinate of* $\mathrm{Var}(u)$ *is at least as large as the corresponding coordinate in* $\mathrm{Var}(v)$.
  Similarly, for any fixed v, the vector $\mathrm{Var}(u : v)$ *is monotonically non-increasing as u moves towards the leaves.*
- *For any multiplication gate $u = u_1 \times u_2$, we have* $\mathrm{Var}(u) = \mathrm{Var}(u_1) + \mathrm{Var}(u_2)$. *Similarly for any v, we have* $\mathrm{Var}(u : v) = \mathrm{Var}(u_1) + \mathrm{Var}(u_2 : v)$.
- *For any addition gate $u = u_1 + u_2$, we have* $\mathrm{Var}(u) = \max(\mathrm{Var}(u_1), \mathrm{Var}(u_2))$, *the coordinate-wise max of the two vectors. Similarly for any v,* $\mathrm{Var}(u : v) = \max(\mathrm{Var}(u_1 : v), \mathrm{Var}(u_2 : v))$.

Proof. The proofs immediately follow from the definitions. ◀

For two vectors $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{N}^n_{\geq 0}$, we shall say $\mathbf{v}_1 \preceq \mathbf{v}_2$ if each coordinate of $\mathbf{v}_1$ is at most the corresponding coordinate in $\mathbf{v}_2$.

▶ **Observation 16.** *Suppose $u \in C$ and w is a node in C such that there is some parse-tree rooted at u with w appearing on its rightmost path. Then,*

$$\mathrm{Var}(u : w) + \mathrm{Var}(w) \preceq \mathrm{Var}(u).$$

Similarly, suppose w is a node in C such that there is some v-parse-tree rooted at u with w appearing on its rightmost path. Then,

$$\mathrm{Var}(u : w) + \mathrm{Var}(w : v) \preceq \mathrm{Var}(u : v).$$

Proof. The proof is straightforward; we just give the proof of the second equation. Fix a coordinate i. If $d_i = (\mathrm{Var}(u : w))_i$ then there is some w-snipped parse-tree T_i rooted at u whose x_i-degree equals d_i. Similarly if $e_i = (\mathrm{Var}(w : v))_i$, then there is some v-snipped parse-tree T'_i rooted at w whose x_i-degree is e_i. Clearly the *gluing* of T_i and T'_i obtained by replacing the snipped vertex w in T_i with the tree T'_i is a v-snipped parse-tree rooted at u with x_i-degree $d_i + e_i$. Therefore $d_i + e_i \leq (\mathrm{Var}(u : v))_i$ and the claim follows. ◀

▶ **Definition 17** (Syntactically multilinear and multi-k-ic circuits). *A circuit C is said to be* syntactically multilinear *if* $\mathrm{Var}(u) \in \{0, 1\}^n$ *for all $u \in C$.*

A circuit C is said to be syntactically multi-k-ic *if* $\mathrm{Var}(u) \in \{0, 1, \ldots, k\}^n$ *for all $u \in C$.*

3 Frontier edges and quotient

▶ **Definition 18** (Frontier edges). *For a circuit C, an edge between two gates g_1, g_2 (where g_1 is the parent) is said to be an* m-frontier edge *(for a parameter m) if*

$$|\mathrm{Var}(g_1)| \geq m \text{ and } |\mathrm{Var}(g_2)| < m.$$

We use $\mathcal{F}_m^{\times}$ to denote the set of all m-frontier edges (g_1, g_2) where g_1 is a multiplication gate and g_2 is its right child. *We use $\mathcal{F}_m^{+}$ to denote those where g_1 is an addition gate.*

Furthermore, if $v \in C$ is a fixed gate, we shall say that (g_1, g_2) is an m-frontier edge with respect to v *if*

$$|\mathrm{Var}(g_1 : v)| \geq m \text{ and } |\mathrm{Var}(g_2 : v)| < m.$$

We will use $\mathcal{F}_{m,v}^{\times}$ to denote the set of all edges (g_1, g_2) that are m-frontier edges with respect to v where g_1 is a multiplication gate (and g_2 is its right child), and $\mathcal{F}_{m,v}^{+}$ to denote those where g_1 is an addition gate (and g_2 is any child).

4 Decomposition via gate quotients

In this section, we prove the following lemma, which is the key technical observation needed for our proofs.

▶ **Lemma 19.** *Let u, v be gates in an algebraic circuit C with $|\mathrm{Var}(u)| \geq m$ and $|\mathrm{Var}(v)| < m$. Then,*

$$[u] = \sum_{(w,z)\in\mathcal{F}_m^{\times}} [u : w] \cdot [w_L] \cdot [z] \; + \sum_{(w,z)\in\mathcal{F}_m^{+}} [u : w] \cdot [z] \tag{1}$$

$$[u : v] = \sum_{(w,z)\in\mathcal{F}_{m,v}^{\times}} [u : w] \cdot [w_L] \cdot [z : v] \; + \sum_{(w,z)\in\mathcal{F}_{m,v}^{+}} [u : w] \cdot [z : v] \tag{2}$$

We shall give an informal sketch using the concept of parse-trees. A complete formal proof of the lemma can be found in Appendix A. For any u, v, we have that $[u : v]$ is the sum of all v-snipped parse-trees rooted at u. For any parse-tree, since $|\mathrm{Var}(u)| \geq m$ and $|\mathrm{Var}(v)| < m$ and $\mathrm{Var}(\cdot)$ is a monotonically non-increasing function as we move towards the leaves, there must be a unique edge $(w, z) \in \mathcal{F}_{m,v}^{\times} \cup \mathcal{F}_{m,v}^{+}$ on its right-most path such that $|\mathrm{Var}(w)| \geq m$ and $|\mathrm{Var}(z)| < m$.

If $(w, z) \in \mathcal{F}_{m,v}^{\times}$, then $w = w_L \times z$ is a multiplication gate. Therefore, the sum of the values of all v-snipped parse-trees with w (and hence the edge (w, z)) on its rightmost path is exactly $[u : w][w : v] = [u : w][w_L][z : v]$.

If $(w, z) \in \mathcal{F}_{m,v}^{+}$, then $w = w_1 + z$ is an addition gate. Then, $[u : w] \cdot [w : v]$ is the sum of all v-snipped parse-trees with w on its rightmost path and $[u : w][w : v] = [u : w][w_1 : v] + [u : w][z : v]$. Each v-snipped parse-tree with w on its rightmost path either has (w, w_1) on the rightmost path or (w, z). The term $[u : w][w_1 : v]$ is precisely the sum of the values of such[4] parse-trees with (w, w_1) on its rightmost path, and $[u : w][z : v]$ is precisely the sum of the values of those parse-trees with (w, z) on its rightmost path.

[4] v-snipped parse-trees rooted at u that have w on its rightmost path

Since the rightmost path of any v-snipped parse-tree rooted at u has a *unique* edge $(w,z) \in \mathcal{F}^{\times}_{m,v} \cup \mathcal{F}^{+}_{m,v}$, summing over all such potential edges gives

$$[u:v] = \sum_{(w,z)\in\mathcal{F}^{\times}_{m,v}} [u:w]\cdot[w_L]\cdot[z:v] \;+\; \sum_{(w,z)\in\mathcal{F}^{+}_{m,v}} [u:w]\cdot[z:v].$$

5 Balancing syntactically multilinear circuits

In this section, we prove the following theorem.

▶ **Theorem 20.** *Suppose C is an algebraic circuit of size s. Then, there is a circuit C' of size* poly(s) *computing the same polynomial with the following structural properties.*

- *all addition gates in C' have fan-in $O(s^4)$,*
- *all multiplication gates in C' have fan-in at most 5,*
- *for any multiplication gate $g \in C'$, any child h of g satisfies $|\mathrm{Var}(h)| \le |\mathrm{Var}(g)|/2$.*

Furthermore, if C is syntactically multi-k-ic, then so is C'.

Proof. Without loss of generality, we may assume that the circuit is *right-heavy* in the sense that for every multiplication gate $u = u_1 \times u_2$ we have $|\mathrm{Var}(u_2)| \ge |\mathrm{Var}(u_1)|$. We shall build a new circuit C' that computes all $[u:v]$'s and $[u]$'s for gates $u, v \in C$ using the equations in Lemma 19.

We shall assume inductively that we have already computed all $[w]$'s with $|\mathrm{Var}(w)| < t$ and also all $[w,v]$ with $|\mathrm{Var}(w,v)| < t$. Suppose $u \in C$ such that $|\mathrm{Var}(u)| = t$. Using (1) from Lemma 19 with $m = t/2$ we have

$$[u] = \sum_{(w,z)\in\mathcal{F}^{\times}_{m}} [u:w]\cdot[w_L]\cdot[z] \;+\; \sum_{(w,z)\in\mathcal{F}^{+}_{m}} [u:w]\cdot[z].$$

By Observation 16, $|\mathrm{Var}(w)| \ge t/2$ implies that $|\mathrm{Var}(u:w)| \le t/2$. Furthermore, $|\mathrm{Var}(z)| \le t/2$ by the choice of the frontier edge and $|\mathrm{Var}(w_L)| \le t/2$ since C is right-heavy. This allows us to compute all nodes of the form $[u]$ with $|\mathrm{Var}(u)| \le t$.

If $u, v \in C$ such that $|\mathrm{Var}(u:v)| = t$. Using (2) from Lemma 19 with $m = t/2$, we have

$$[u:v] = \sum_{(w,z)\in\mathcal{F}^{\times}_{m,v}} [u:w]\cdot[w_L]\cdot[z:v] \;+\; \sum_{(w,z)\in\mathcal{F}^{+}_{m,v}} [u:w]\cdot[z:v].$$

We can restrict the edges in the RHS to only those edges (w,z) that is present in at least one v-snipped parse-tree rooted at u (if not, this edge's contribution to the RHS is zero). Therefore by Observation 16, $\mathrm{Var}(w:v) + \mathrm{Var}(u:w) \preceq \mathrm{Var}(u:v)$ and therefore we have $|\mathrm{Var}(u:w)| \le t/2$. Furthermore, by the choice of the frontier, we also have $|\mathrm{Var}(z:v)| \le t/2$. The non-trivial case is $\mathrm{Var}(w_L)$ which could in principle be large but again $\mathrm{Var}(w_L) \preceq \mathrm{Var}(w:v) \preceq \mathrm{Var}(u:v)$ as any parse-tree rooted w_L is a sub-tree of a v-snipped tree rooted at u. Since we have already computed all gates $[w]$ with $\mathrm{Var}(w) \le t$, we can write

$$\begin{aligned}
[u:v] &= \sum_{(w,z)\in\mathcal{F}^{\times}_{m,v}} [u:w]\cdot[w_L]\cdot[z:v] \;+\; \sum_{(w,z)\in\mathcal{F}^{+}_{m,v}} [u:w]\cdot[z:v] \\
&= \sum_{(w,z)\in\mathcal{F}^{\times}_{m,v}} [u:w]\cdot\left(\sum_{(p,q)\in\mathcal{F}^{\times}_{m_w}} [w_L:p]\cdot[p_L]\cdot[q] \;+\; \sum_{(p,q)\in\mathcal{F}^{+}_{m_w}} [w_L:p]\cdot[q]\right)\cdot[z:v]
\end{aligned}$$

$$+ \sum_{(w,z)\in\mathcal{F}^{+}_{m,v}} [u:w]\cdot[z:v],$$

where $m_w = \mathrm{Var}(w_L)/2$.

The required structural properties of C' are readily seen from the above construction. ◀

6 Reduction to depth four from balanced circuits

We now show how to reduce a balanced circuit to a depth-4 circuit. This would complete the proof of our main theorem. We shall use the notation $\Sigma\Pi\left(\Sigma\Pi\right)_t$ to refer to $\Sigma\Pi\Sigma\Pi$ circuits computing polynmomials of the form

$$F = \sum_i \prod_j Q_{ij},$$

with $|\mathrm{Var}(Q_{ij})| \le t$.

The proof of this part follows the outline of a similar argument in Chillara et al. [4] of reducing to depth-4 from a balanced circuit. However, there are some differences: our potential is $|\mathrm{Var}(\,\cdot\,)|$ and not the degree (as is usually the case). Since this potential function also falls as we go from a sum (+) gate to its children, we need one more simple observation in our argument to bound the number of steps in the recursion in the proof. We now provide the details.

▶ **Lemma 21.** *Let C be a multi-k-ic circuit of size s such that every multiplication gate g in C has fan-in at most 5 and for every child h of g in C, $\mathrm{Var}(h) \le \mathrm{Var}(g)/2$.*

Then, for any positive integer $0 \le t \le kn$, there is an equivalent multi-k-ic $\Sigma\Pi\left(\Sigma\Pi\right)_t$ circuit C' that computes the same polynomial, with the following properties:

- *the top fan-in of C' is at most $s^{O(kn/t)}$,*
- *the size of C' is at most $2^t \cdot s^{O(kn/t)}$,*
- *each of the (+)-gates closer to the leaves compute polynomials that are computed by gates in C.*

Proof. Since C is balanced, with product fan-in at most 5, every gate g in C can be written as

$$g = \sum_{i=1}^{s} \prod_{j=1}^{5} g_{i,j}\,, \tag{3}$$

where each $g_{i,j}$ is also computed by a gate in the circuit C, $|\,\mathrm{Var}(g_{i,j})| \le |\,\mathrm{Var}(g)|/2$. With this notation, (3) applied on the root of C says that C, which is a syntactically multi-k-ic circuit, can be trivially written as a $\Sigma\Pi\left(\Sigma\Pi\right)_{kn/2}$. A natural idea would be to apply (3) on the $g_{i,j}$'s until we get a $\Sigma\Pi\left(\Sigma\Pi\right)_t$ circuit. All that is needed is to bound the number of summands (or the top fan-in of the resulting $\Sigma\Pi\left(\Sigma\Pi\right)_t$ circuit) at the end of this process. Observe that for every $i \in \{1,2,\ldots,s\}$, we could have that $\left|\mathrm{Var}(\prod_{j=1}^{5} g_{i,j})\right|$ is *much* smaller than $|\mathrm{Var}(g)|$ itself. To handle this, we shall pretend that

We will view the process as a tree in the natural way. The root of the tree corresponds to the root of the circuit, and all other nodes in the tree correspond to products of addition gates in C. The children of a node in the tree correspond to the summands in the sum of product representation of that node obtained by expanding one of its factors according to (3). The leaves of this tree are products of addition gates $\prod g'_i$ such that $|\mathrm{Var}(g'_i)| \le t$ for

each factor g_i'. The tree has a branching factor of at most s, hence it suffices to get a bound on the depth of the tree to get a bound on the number of leaves which would be the top fan-in of the $\Sigma\Pi(\Sigma\Pi)_t$ representation.

Let $g\prod_\ell w_\ell$ be an internal node in the tree with $|\mathrm{Var}(g)| > t$. After applying (3) on g, we get

$$g\left(\prod_\ell w_\ell\right) = \sum_{i=1}^{s}\left(\prod_{j=1}^{5} g_{i,j}\cdot\prod_\ell w_\ell\right).$$

We now consider two cases.

- $\left|\mathbf{Var}\left(\prod_{j=1}^{5} g_{i,j}\right)\right| < 3t/4$: In this case, $\left|\mathrm{Var}\left(\prod_{j=1}^{5} g_{i,j}\cdot\prod_\ell w_\ell\right)\right| \le |\mathrm{Var}\left(g\cdot\prod_\ell w_\ell\right)| - t/4$.
- $\left|\mathbf{Var}\left(\prod_{j=1}^{5} g_{i,j}\right)\right| \ge 3t/4$: Since $\mathrm{Var}(g) \succeq \mathrm{Var}(g_{i,1}\cdots g_{i,5}) = \mathrm{Var}(g_{i,1}) + \cdots + \mathrm{Var}(g_{i,5})$ and $|\mathrm{Var}(g_{i,j})| \le t/2$, it follows that the number of factors h in $\prod_{j=1}^{5} g_{i,j}\cdot\prod_\ell w_\ell$ with $|\mathrm{Var}(h)| \ge t/16$ is at least one more than the number of such factors in $g\cdot\prod_\ell w_\ell$. This is because besides the factor $g_{i,j}$ with largest $|\mathrm{Var}(g_{i,j})|$, the other four factors together must contribute at least $(3t/4) - (t/2) = (t/4)$ to $|\mathrm{Var}(g_{i,1}\cdots g_{i,5})|$ and hence at least one of them must have $|\mathrm{Var}(g_{i,k})| \ge t/16$.

Thus, in any edge of the tree, either $|\mathrm{Var}(\,\cdot\,)|$ decreases by $t/4$ or the number of factors with $|\mathrm{Var}(\,\cdot\,)| \ge t/16$ increases by one. The root node g_0 has $|\mathrm{Var}(g_0)| \le kn$. Hence, the depth of the tree is bounded by $(16+4)(kn/t) = O(nk/t)$. Therefore, C can be computed by a syntactically multi-k-ic $\Sigma\Pi(\Sigma\Pi)_t$ circuit of top fan-in at most $s^{O(nk/t)}$.

To get the bound on the overall size of the $\Sigma\Pi(\Sigma\Pi)_t$ circuit, we need to bound the sparsity of the polynomials computed by bottom two layers. Note that if $\mathrm{Var}(f) = (d_1,\ldots,d_n)$, then f can have at most $\prod(1+d_i)$ monomials. Since $2^x \ge 1+x$ for all positive integers x, it follows that $|\mathrm{Var}(f)| \le t$ implies that f has at most 2^t monomials. Therefore, the total size of the $\Sigma\Pi(\Sigma\Pi)_t$ circuit is $2^t\cdot s^{O(kn/t)} = 2^{O\left(t+\frac{kn\log s}{t}\right)}$. ◀

From Theorem 20 and setting $t = \sqrt{kn\log s}$ in Lemma 21, we get Theorem 7 restated below.

▶ **Theorem 7.** *Let C be a multi-k-ic circuit of size s computing a polynomial in n variables. Then, there is a multi-k-ic $\Sigma\Pi\Sigma\Pi$ circuit C' of size $s^{O\left(\sqrt{\frac{kn}{\log s}}\right)}$ computing the same polynomial.*

6.1 Reduction to higher depths

We now prove Theorem 8 which shows that similar savings can be obtained in depth reductions to larger depth.

▶ **Theorem 8.** *Let C be a multi-k-ic circuit of size s computing a polynomial in n variables. Then, there is a multi-k-ic $(\Sigma\Pi)^\Delta$ circuit C' computing the same polynomial whose size is at most*

$$s^{O\left(\Delta\cdot(nk/\log s)^{1/\Delta}\right)}.$$

Proof of Theorem 8. We shall assume, without loss of generality, that the circuit C is balanced (by applying Theorem 20 if necessary). The proof follows via repeated applications of Lemma 21.

Applying Lemma 21 with $t = nk/(nk/\log s)^{1/\Delta}$, we obtain a $\Sigma\Pi\,(\Sigma\Pi)_t$ circuit C' of the form

$$C' = \sum_{i=1}^{s'} \prod_j g_{ij},$$

with $s' = s^{O\left((kn/\log s)^{1/\Delta}\right)}$ and $|\mathrm{Var}(g_{ij})| \leq t$ for all i, j. Furthermore, since each g_{ij} being a polynomial computed by a gate in C, they are computable by multi-k-ic circuits of size at most s. By induction, each g_{ij} has a multi-k-ic $(\Sigma\Pi)^{\Delta-1}$ circuit of size at most

$$s^{O\left((\Delta-1)\cdot(t/\log s)^{1/(\Delta-1)}\right)} = s^{O\left((\Delta-1)\cdot(nk/\log s)^{1/\Delta}\right)}.$$

Replacing each g_{ij} by this circuit, we obtain a $(\Sigma\Pi)^{\Delta}$ circuit of size at most

$$s' \cdot s^{O\left((\Delta-1)\cdot(nk/\log s)^{1/\Delta}\right)} = s^{O\left(\Delta\cdot(kn/\log s)^{1/\Delta}\right)}.$$ ◀

7 Open problems

The most interesting question that comes out of this work is to prove a lower bound of $n^{\omega(\sqrt{n/\log n})}$ for syntactically multilinear circuits of depth-4 for an explicit polynomial. A natural and first approach to this could be to understand if the shifted partials based methods can prove a lower bound of $n^{\Omega(\sqrt{d})}$ for homogeneous depth-4 circuits for a polynomial family with degree $d = \omega(n/\log n)$.

Another question of interest would be to understand the *correct* exponent for the depth reduction results to depth-4 (and also to higher depth) for various regimes of the degree d. From [9], we know that for $d = O(n^{\varepsilon})$ for a small enough constant ε, $\sqrt{d}$ is the correct exponent, whereas for d being nearly n, the results in this paper and those of Raz and Yehudayoff [12] show that the correct exponent is $\sqrt{n/\log n}$. But we do not understand this phenomenon for other values of d.

References

1. Manindra Agrawal and V. Vinay. Arithmetic Circuits: A Chasm at Depth Four. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008)*, pages 67–75, 2008. doi:10.1109/FOCS.2008.32.
2. Eric Allender, Jia Jiao, Meena Mahajan, and V. Vinay. Non-Commutative Arithmetic Circuits: Depth Reduction and Size Lower Bounds. *Theoretical Computer Science*, 209(1-2):47–86, 1998. doi:10.1016/S0304-3975(97)00227-2.
3. Noga Alon, Mrinal Kumar, and Ben Lee Volk. Unbalancing Sets and an Almost Quadratic Lower Bound for Syntactically Multilinear Arithmetic Circuits. In *CCC 2018*, pages 1–16, 2018. arXiv:1708.02037. doi:10.4230/LIPIcs.CCC.2018.11.
4. Suryajith Chillara, Mrinal Kumar, Ramprasad Saptharishi, and V. Vinay. The Chasm at Depth Four, and Tensor Rank : Old results, new insights. *CoRR*, 2016. arXiv:1606.04200.
5. Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Approaching the Chasm at Depth Four. *Journal of the ACM*, 61(6):33:1–33:16, 2014. Preliminary version in the *28th Annual IEEE Conference on Computational Complexity (CCC 2013)*. doi:10.1145/2629541.
6. Sumant Hegde and Chandan Saha. Improved Lower Bound for Multi-r-ic Depth Four Circuits as a Function of the Number of Input Variables. *Proceedings of Indian National Science Academy*, 83(4):907–922, 2017. doi:10.16943/ptinsa/2017/49224.

7 Neeraj Kayal. An exponential lower bound for the sum of powers of bounded degree polynomials. In *Electronic Colloquium on Computational Complexity (ECCC)TR12-081*, 2012. URL: http://eccc.hpi-web.de/report/2012/081/.

8 Pascal Koiran. Arithmetic Circuits: The Chasm at Depth Four Gets Wider. *Theoretical Computer Science*, 448:56–65, 2012. doi:10.1016/j.tcs.2012.03.041.

9 Mrinal Kumar and Shubhangi Saraf. On the power of homogeneous depth 4 arithmetic circuits. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2014)*, pages 364–373, 2014. doi:10.1109/FOCS.2014.46.

10 Ran Raz. Multi-Linear Formulas for Permanent and Determinant are of Super-Polynomial Size. *J. ACM*, 56(2):8:1–8:17, 2009. Preliminary version in the *36th Annual ACM Symposium on Theory of Computing (STOC 2004)*. doi:10.1145/1502793.1502797.

11 Ran Raz, Amir Shpilka, and Amir Yehudayoff. A lower bound for the size of syntactically multilinear arithmetic circuits. *SIAM J. Comput.*, 38(4):1624–1647, 2008. Preliminary version in the *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007)*. doi:10.1137/070707932.

12 Ran Raz and Amir Yehudayoff. Lower Bounds and Separations for Constant Depth Multilinear Circuits. *Computational Complexity*, 18(2):171–207, 2009. Preliminary version in the *23rd Annual IEEE Conference on Computational Complexity (CCC 2008)*. doi:10.1007/s00037-009-0270-8.

13 Amir Shpilka and Amir Yehudayoff. Arithmetic Circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5:207–388, March 2010. doi:10.1561/0400000039.

14 Sébastien Tavenas. Improved bounds for reduction to depth 4 and depth 3. *Inf. Comput.*, 240:2–11, 2015. Preliminary version in the *38th Internationl Symposium on the Mathematical Foundations of Computer Science (MFCS 2013)*. doi:10.1016/j.ic.2014.09.004.

15 Leslie G. Valiant, Sven Skyum, S. Berkowitz, and Charles Rackoff. Fast Parallel Computation of Polynomials Using Few Processors. *SIAM J. Comput.*, 12(4):641–644, 1983. Preliminary version in the *6th Internationl Symposium on the Mathematical Foundations of Computer Science (MFCS 1981)*. doi:10.1137/0212043.

A Proof of Lemma 19

Proof of Lemma 19. The proof shall proceed by induction on the height of u (leaves are at height 0). We shall present the proof of (2); the proof of (1) is analogous.

Case 1: $u = u_L \times u_R$. For any w, we have that $[u : w] = 1$ if $u = w$, and $[u : w] = [u_1] \cdot [u_2 : w]$ whenever $u \neq w$. In particular, since $|\mathrm{Var}(v)| < m \leq |\mathrm{Var}(u)|$ the LHS is $[u : v] = [u_L] \cdot [u_R : v]$.

If $|\mathrm{Var}(u_R)| \geq m$, then for any $(w, z) \in \mathbb{F}^+_{m,v}$ or $\mathcal{F}^\times_{m,v}$ we have $w \neq u$. Inducting on u_R,

$$\begin{aligned}
\mathrm{LHS} &= [u_L] \cdot [u_R : v] \\
&= [u_L] \cdot \left(\sum_{(w,z) \in \mathcal{F}^\times_{m,v}} [u_R : w] \cdot [w_L] \cdot [z : v] \; + \sum_{(w,z) \in \mathcal{F}^+_{m,v}} [u_R : w] \cdot [z : v] \right) \\
&= \sum_{(w,z) \in \mathcal{F}^\times_{m,v}} [u_L] \cdot [u_R : w] \cdot [w_L] \cdot [z : v] \; + \sum_{(w,z) \in \mathcal{F}^+_{m,v}} [u_L] \cdot [u_R : w] \cdot [z : v] \\
&= \sum_{(w,z) \in \mathcal{F}^\times_{m,v}} [u : w] \cdot [w_L] \cdot [z : v] \; + \sum_{(w,z) \in \mathcal{F}^+_{m,v}} [u : w] \cdot [z : v] \; = \; \mathrm{RHS}.
\end{aligned}$$

On the other hand, if $|\mathrm{Var}(u_R)| < m$ then $[u : w] = 0$ for any $w \neq u$ with $|\mathrm{Var}(w)| \geq m$. Hence,

$$\begin{aligned}\text{RHS} &= \sum_{(w,z)\in\mathcal{F}^{\times}_{m,v}} [u:w]\cdot[w_L]\cdot[z:v] \;+\; \sum_{(w,z)\in\mathcal{F}^{+}_{m,v}} [u:w]\cdot[z:v] \\ &= [u:u]\cdot[u_L]\cdot[u_R:v] = [u:v] = \text{LHS}.\end{aligned}$$

Case 2: $u = u_1 + u_2$. For any w, we have that $[u : w] = 1$ if $u = w$, and $[u : w] = [u_1 : w] + [u_2 : w]$ whenever $u \neq w$. In particular, since $|\mathrm{Var}(v)| < m \leq |\mathrm{Var}(u)|$ the LHS is $[u : v] = [u_1 : v] + [u_2 : v]$.
Since u is a + gate, $(u, u_j) \notin \mathcal{F}^{\times}_{m,v}$ for any j. If $|\mathrm{Var}(u_j)| < m$ for some j, then the edge $(u, u_j) \in \mathcal{F}^{+}_{m,v}$. Hence,

$$\begin{aligned}\text{RHS} &= \sum_{(w,z)\in\mathcal{F}^{\times}_{m,v}} [u:w]\cdot[w_L]\cdot[z:v] \;+\; \sum_{(w,z)\in\mathcal{F}^{+}_{m,v}} [u:w]\cdot[z:v] \\ &=: \qquad\qquad T_1 \qquad\qquad + \qquad\qquad T_2\end{aligned}$$

In T_1, since every $(w, z) \in \mathcal{F}^{\times}_{m,v}$ has $w \neq u$ we have

$$\begin{aligned}T_1 &:= \sum_{(w,z)\in\mathcal{F}^{\times}_{m,v}} \left(\sum_i [u_i:w]\right)\cdot[w_L]\cdot[z:v] \\ &= \sum_{(w,z)\in\mathcal{F}^{\times}_{m,v}} \left(\sum_{i:|\mathrm{Var}(u_i)|\geq m} [u_i:w]\right)\cdot[w_L]\cdot[z:v] \qquad (\text{since } [u_j:w]=0 \text{ if } |\mathrm{Var}(u_j)|<m) \\ &= \sum_{i:|\mathrm{Var}(u_i)|\geq m} \;\sum_{(w,z)\in\mathcal{F}^{\times}_{m,v}} [u_i:w]\cdot[w_L]\cdot[z:v].\end{aligned}$$

As for the other term, it can be written as

$$\begin{aligned}T_2 &:= \sum_{(w,z)\in\mathcal{F}^{+}_{m,v}} [u:w]\cdot[z:v] \\ &= \sum_{\substack{(w,z)\in\mathcal{F}^{+}_{m,v}\\ w\neq u}} [u:w]\cdot[z:v] \;+\; \sum_{j:|\mathrm{Var}(u_j)|<m} [u:u]\cdot[u_j:v] \\ &= \sum_{\substack{(w,z)\in\mathcal{F}^{+}_{m,v}\\ w\neq u}} \left(\sum_i [u_i:w]\right)\cdot[z:v] \;+\; \sum_{j:|\mathrm{Var}(u_j)|<m} [u_j:v] \\ &= \sum_{\substack{(w,z)\in\mathcal{F}^{+}_{m,v}\\ w\neq u}} \left(\sum_{i:|\mathrm{Var}(u_i)|\geq m} [u_i:w]\right)\cdot[z:v] \;+\; \sum_{j:|\mathrm{Var}(u_j)|<m} [u_j:v] \\ &= \sum_{i:|\mathrm{Var}(u_i)|\geq m} \;\sum_{(w,z)\in\mathcal{F}^{+}_{m,v}} [u_i:w]\cdot[z:v] \;+\; \sum_{j:|\mathrm{Var}(u_j)|<m} [u_j:v].\end{aligned}$$

The last equality holds because $[u_i : u] = 0$. Putting it together,

$$\begin{aligned}\text{RHS} &= T_1 + T_2 \\ &= \sum_{i:|\mathrm{Var}(u_i)|\geq m} \left(\sum_{(w,z)\in\mathcal{F}^{\times}_{m,v}} [u_i:w]\cdot[w_L]\cdot[z:v] + \sum_{(w,z)\in\mathcal{F}^{+}_{m,v}} [u_i:w]\cdot[z:v]\right)\end{aligned}$$

$$+ \sum_{j:|\mathrm{Var}(u_j)|<m} [u_j : v]$$

$$= \sum_{i:|\mathrm{Var}(u_i)|\geq m} [u_i : v] + \sum_{j:|\mathrm{Var}(u_j)|<m} [u_j : v] \qquad \text{(induction)}$$

$$= [u : v] = \mathrm{LHS}. \qquad \blacktriangleleft$$

Sum-Of-Squares Bounds via Boolean Function Analysis

Adam Kurpisz
ETH Zürich, Department of Mathematics, Rämistrasse 101, 8092 Zürich, Switzerland
adam.kurpisz@ifor.math.ethz.ch

Abstract

We introduce a method for proving bounds on the SoS rank based on Boolean Function Analysis and Approximation Theory. We apply our technique to improve upon existing results, thus making progress towards answering several open questions.

We consider two questions by Laurent. First, finding what is the SoS rank of the linear representation of the set with no integral points. We prove that the SoS rank is between $\lceil \frac{n}{2} \rceil$ and $\lceil \frac{n}{2} + \sqrt{n \log 2n} \rceil$. Second, proving the bounds on the SoS rank for the instance of the Min Knapsack problem. We show that the SoS rank is at least $\Omega(\sqrt{n})$ and at most $\lceil \frac{n+4\lceil \sqrt{n} \rceil}{2} \rceil$. Finally, we consider the question by Bienstock regarding the instance of the Set Cover problem. For this problem we prove the SoS rank lower bound of $\Omega(\sqrt{n})$.

2012 ACM Subject Classification Theory of computation → Semidefinite programming; Theory of computation → Convex optimization

Keywords and phrases SoS certificate, SoS rank, hypercube optimization, semidefinite programming

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.79

Category Track A: Algorithms, Complexity and Games

Funding Supported by SNSF project PZ00P2_174117.

Acknowledgements I would like to express my gratitude to Markus Schweighofer for fruitful discussions.

1 Introduction

A *boolean hypercube optimization problem* is a (constrained) polynomial optimization problem in n-variables, where the set of feasible points is restricted to a subset of an n-dimensional hypercube. This family of problems lies at the heart of theoretical computer science. However, solving general optimization problem over boolean hypercube is NP-hard, since the class contains, e.g., the Independent Set problem.

One of the most successful approaches for constructing theoretically efficient algorithms for this family of problems is the *Sum of Squares (SoS)* algorithm [23, 41, 43, 51]. For a wide variety of combinatorial optimization problems it provides the best available algorithms [1, 20, 5, 24, 36]. Recently, SoS has also been applied to problems in ROBUST ESTIMATION [26], DICTIONARY LEARNING [3, 49] and TENSOR COMPLETION AND DECOMPOSITION [4, 25, 45]. Other applications can be found in [5, 7, 12, 13, 17, 18, 24, 38, 39, 46]; see also the surveys [14, 33, 35].

On the other hand it is known that the SoS algorithm admits certain weaknesses. For example, a $\Omega(n)$ degree SoS certificate is needed to detect a simple integrality argument for some instance of the KNAPSACK problem as shoved by Grigoriev in [21], see also [22, 28, 34]. Further weaknesses of SoS for KNAPSACK problems were proved in [11, 31]. Some lower bounds on the effectiveness of the SoS have been shown for CSP problems [27, 52] and for

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 79; pp. 79:1–79:15

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

the planted clique problem [2, 40]. Finally, degree $\Omega(\sqrt{n})$ SoS was proved to have problems scheduling unit size jobs on a single machine to minimize the number of late jobs, see [30]. The problem is solvable in polynomial time using the Moore-Hodgson algorithm.

Proving SoS lower bounds is not an easy task. In most of the results, the lower bound is proved on the *Lasserre relaxation* side, by explicitly giving a *pseudoexpectation* operator that maps polynomials of certain degree to real numbers, such that the corresponding *moment matrix* for variables and *localizing matrices* for constraints are Positive Semi-definite (PSD). Both finding a suitable pseudoexpectation and proving PSDness might be very difficult.

In this paper we follow the less common approach of working on a *dual* side, the *SoS certificate of nonnegativity*. This approach does not require finding a pseudoexpectation operator and proving PSDness of the matrices, which we find very convenient. We prove lower bounds by exploiting the properties of boolean functions and by using the results from approximation theory, especially the theory of approximating boolean functions, see e.g. [44, 50, 53]. This, in our case, directly implies the existence/non-existence of the SoS certificate. To the best of our knowledge there are not that many SoS lower bounds using this approach, see e.g. [37].

On an intuitive level our approach might be explained in the following way. Imagine one wants to write a degree d SoS certificate for some function f that is nonnegative over a subset of the boolean hypercube. Consider a vertex x of the hypercube, where $f(x) < 0$. Note that in order to successfully write a certificate one has to use the constraint g, that made the point x infeasible, and multiply it by some SoS polynomial s of degree at most $2d$. Moreover, the values $f(x)$ and $g(x)$ give some lower bound on the value of $s(x)$. Ideally, we would like s to take value zero on other vertices of the hypercube, this is the case for degree n certificates. However, the lower the degree d, the more other vertices of the hypercube are "affected" by the value of $s(x)$.

This simple observation is strong enough to provide the best known SoS ranks for the following problems:

For $B \geq 2$. The Empty Integral Hull (EIH) problem is a feasibility problem of the form

$$EIH = \{0,1\}^n \cap \{x \in [0,1]^n \mid \sum_{i \in [n] \setminus I} x_i + \sum_{i \in I} (1 - x_i) \geq \frac{1}{B} \text{ for all } I \subseteq [n]\} \tag{1.1}$$

For $B = 2$, a long list of results for the performance of the lift and project methods for EIH problem is known. In [33], Laurent shows that the Sherali-Adams *rank* is n. They then conjecture the SoS rank of EIH is $n - 1$. Moreover, the rank is also equal to n for the Lovász-Schrijver N_+ operator (with positive semidefiniteness) [19], the Lovász-Schrijver N_+ operator strengthen with Chvátal cuts [15], and the N_+ operator combined with Gomory mixed integer cuts (equivalent to disjunctive cuts) [16]. Recently, in [29], the conjecture was disproved, and it was shown that the SoS rank of EIH is between $\Omega(\sqrt{n})$ and $n - \Omega(n^{1/3})$. In this paper we prove the following result:

▶ **Theorem 1.** *For $B = 2$, the SoS rank of EIH problem is between $\lceil \frac{n}{2} \rceil$ and $\lceil \frac{n}{2} + \sqrt{n \log 2n} \rceil$.*

In this paper we also prove lower and upper bounds on the SoS rank for any $B \geq 2$, see Theorem 12 and Lemma 13.

The second considered problem is the instance of the Min Knapsack (MK) problem. For $P \geq 2$, the problem is defined as:

$$\text{MK:} \quad \min \sum_{i \in [n]} x_i \qquad \text{s.t.} \quad \sum_{i \in [n]} x_i \geq \frac{1}{P} \qquad x \in \{0,1\}^n \tag{1.2}$$

For $P = 2$ the problem was previously considered by Cook and Dash [15]. They proved that the Lovasz-Schrijver hierarchy rank is n. For the Sherali-Adams hierarchy Laurent in [33] proved that the rank is also equal to n and raised the open question to find the rank for the SoS hierarchy. For $n = 2$ they also proved that the SoS rank is 2, but whether or not this happens for general n was left open. In [30] the possibility that the Lasserre/SoS rank is n for $n \geq 3$ is ruled out. Finally, in [28], a SoS rank lower bound of $t = \Omega(\log^{1-\varepsilon} n)$, for $\epsilon > 0$, was presented. In this paper we prove the following result:

▶ **Theorem 2.** *For $P = 2$, the SoS rank of the MK problem is between $\Omega(\sqrt{n})$ and $\lceil \frac{n+4\lceil\sqrt{n}\rceil}{2} \rceil$.*

Moreover, we prove a SoS rank lower bound for any $P \geq 2$ of the value $\Omega(\sqrt{n} + \sqrt{n \log P})$, see Lemma 14.

The third problem we consider is the instance of the Set Cover (SC) problem:

$$\text{SC:} \qquad \min \sum_{i\in[n]} x_i \qquad \text{s.t.} \quad \sum_{i\in[n]\setminus\{j\}} x_i \geq 1 \qquad \forall j \in [n] \qquad (1.3)$$
$$x \in \{0,1\}^n$$

This instance was considered in [8], where an open question was raised asking what is the rank of this polytope, conjecturing that the SoS rank is at least $n/4$, based on numerical experiments. In [29], the conjecture was supported by proving that the rank is at least $\log^{1-\delta}(n)$ for any $\delta > 0$. Finally, the instance can be seen as the Min Knapsack instance with Knapsack Cover inequalities, see [29]. In this paper we prove the following result:

▶ **Theorem 3.** *The SoS rank of the SC problem is at least $\Omega(\sqrt{n})$.*

2 Preliminaries

For any $n \in \mathbb{N}$ we denote $[n] = \{1, \ldots, n\}$. Let $\mathbb{R}[x] = \mathbb{R}[x_1, \ldots, x_n]$ be the ring of *n-variate real polynomials*. Furthermore, let $\mathcal{G}$ be the set of polynomials

$$\mathcal{G} \;:=\; \{g_0 := 1, g_1, \ldots, g_m : \; g_i \in \mathbb{R}[x] \text{ for all } \; i \in [m]\}.$$

For a given set $\mathcal{G}$, the corresponding *semialgebraic set* is defined as

$$\mathcal{G}_+ \;:=\; \{x \in \mathbb{R}^n \mid g(x) \geq 0 \text{ for all } g \in \mathcal{G}\} \subseteq \mathbb{R}^n.$$

Moreover, for any given semialgebraic set $\mathcal{G}_+ \subseteq \mathbb{R}^n$, let $\mathcal{K}(\mathcal{G}_+)$ be the set of *nonnegative polynomials* over the set $\mathcal{G}_+$

$$\mathcal{K}(\mathcal{G}_+) \;:=\; \{f \in \mathbb{R}[x] \mid f(x) \geq 0 \text{ for all } x \in \mathcal{G}_+\}.$$

For a given $f \in \mathbb{R}[x]$ and a set $\mathcal{G}$ we define the corresponding *constrained polynomial optimization problem* (CPOP)

$$f^* \;:=\; \min\{f(x) \mid x \in \mathcal{G}_+\} \;=\; \max\{\lambda \in \mathbb{R} \mid f - \lambda \in \mathcal{K}(\mathcal{G}_+)\}. \qquad (2.1)$$

and the *constrained polynomial feasibility problem* (CPFP), which consists in testing whether the corresponding set $\mathcal{G}_+$ is empty or not, see e.g., [9].

Since the problems CPOP and CPFP are NP-hard in general it is desirable to find a proper subset that is a good inner approximation of $\mathcal{K}(\mathcal{G}_+)$ such that the corresponding program is computationally *tractable*.

SoS method

The *SoS method* approximates the cone $\mathcal{K}(\mathcal{G}_+)$ by using the set of *sum of square polynomials*. Let $\mathbb{SOS} := \{s \mid s = \sum_{i=1}^{k} f_i^2,\ f \in \mathbb{R}[x],\ k \in \mathbb{N}^*\}$ be the set of *(finite) sum of square polynomials (SoS)*. Let

$$\Sigma_{n,d}^{\mathcal{G}} := \left\{\sum_{i=1}^{m} s_i g_i \mid s_i \in \mathbb{SOS},\ g_i \in \mathcal{G},\ deg(s_i) \leq 2d, \text{ for all } i \in [m] \text{ and } deg(s_0) \leq 2\left\lceil \tfrac{2d+deg(\mathcal{G})}{2} \right\rceil\right\},$$

where $deg(\mathcal{G}) = \max\{deg(g) \mid g \in \mathcal{G}\}$. The *degree d SoS certificate* for f being nonnegative over $\mathcal{G}_+$ is $f \in \Sigma_{n,d}^{\mathcal{G}}$. The *degree d SoS program* for CPOP takes the form:

$$f_{\mathbb{SOS}}^d := \max\{\lambda \in \mathbb{R} \mid f - \lambda \in \Sigma_{n,d}^{\mathcal{G}}\}; \tag{2.2}$$

and for CPFP it consists in testing whether or not $-1 \in \Sigma_{n,d}^{\mathcal{G}}$ and, if so, implying that the set $\mathcal{G}_+$ is empty.

SoS method over the boolean hypercube

In this paper we consider optimization over the boolean hypercube $\mathcal{H} := \{0,1\}^n$. We assume that $\mathcal{G}$ is such that $\mathcal{G}_+ \subseteq \mathcal{H}$. We fix the following notation. Let $\mathcal{H}^+$ and $\mathcal{H}^-$ be $\mathcal{G}_+$ and $\mathcal{H} \setminus \mathcal{G}_+$, respectively. For $I \in [n]$, let $x_I \in \mathbb{R}^n$ be the characteristic vector of set I such that $(x_I)_i = 1$ if $i \in I$ and 0 otherwise. Moreover, for any $f \in \mathbb{R}[x]$, let $\mathcal{H}^-(f) := \{x \in \mathcal{H} \mid f(x) < 0\}$ and $\mathcal{H}^+(f) := \{x \in \mathcal{H} \mid f(x) \geq 0\}$.

Throughout the paper we assume that $\mathcal{G}$ is always of the form

$$\mathcal{G} := \{g_0 := 1, g_1, \ldots, g_m, \pm(x_1^2 - x_1), \ldots, \pm(x_n^2 - x_n) :\ g_i \in \mathbb{R}[x] \text{ for all } \ i \in [m]\}.$$

In this case solving a degree d SoS program can be done via Semi-definite Program (SDP) of size $O(\sum_{k=0}^{d} \binom{n}{k})$. Moreover, it is known that degree n SoS program is *exact* meaning that for CPOP we have $f_{\mathbb{SOS}}^n = f^*$ and for CPFP we have $-1 \in \Sigma_{n,n}$ if and only if the set $\mathcal{G}_+$ is empty, see e.g. [6, 32, 33]. A very useful result was recently proved in [48] giving an upper bound on the degree of the SoS certificate for every degree r unconstrained boolean hypercube optimization problem:

▶ **Theorem 4** ([48]). *Every n-variate polynomial of degree r, nonnegative over the unconstrained boolean hypercube has a degree $\lceil \frac{n+r-1}{2} \rceil$ SoS certificate.*

Now we give the following definition.

▶ **Definition 5.** *The SoS* rank *for a CPOP (CPFP) is the smallest degree d such that the degree d SoS program is exact.*

That is why the SoS rank for EIH problem is the smallest degree d such that there exist SoS polynomials s_I, for all $I \subseteq [n]$ of degree at most $2d$ and s_0 of degree at most $2d+2$ such that: $-1 = s_0 + \sum_{I \subseteq [n]} s_I(x) \left(\sum_{i \in [n] \setminus I} x_i + \sum_{j \in I}(1 - x_j) - 1/B\right)$.

Analogously, the SoS rank for the MK problem is the smallest d such that there exist SoS polynomials s_1 of degree at most $2d$ and s_0 of degree at most $2d+2$ such that: $\sum_{i \in [n]} x_i - 1 = s_0 + s_1 \left(\sum_{i \in [n]} x_i - 1/P\right)$.

Finally, the SoS rank for the SC problem is the smallest degree d such that there exist SoS polynomials $s_1, \ldots, s_n$ of degree at most $2d$ and s_0 of degree at most $2d+2$ such that: $\sum_{i \in [n]} x_i - 2 = s_0 + \sum_{j \in [n]} s_j(x) \left(\sum_{j \neq i \in [n]} x_i - 1\right)$.

3 Analysis of boolean functions

In this section we show some properties of boolean functions.

The first one is a very intuitive statement saying that the lower degree SoS boolean function we consider, the less of the total mass of the values is in one particular point.

The second one is the special case of results in [44, 50, 53] giving a lower bound on the degree of a real polynomial that approximates the NOR boolean function in ℓ_∞-norm.

▶ **Lemma 6.** *For every function $f : \{0,1\}^n \to \mathbb{R}$ of degree at most d and every subset $J \subseteq [n]$ the following holds:*

$$\sum_{S\subseteq[n]} f^2(x_S) \geq \frac{2^n}{\sum_{i=0}^{d}\binom{n}{i}} f^2(x_J).$$

Moreover, for every $J \subseteq [n]$ there exists a degree d polynomial such that the above inequality is satisfied with equality.

Proof. We will use some elementary Fourier analysis of boolean functions (see e.g. [42, Ch. 1]). For the sake of following an established notation we analyze the function $h : \{\pm 1\}^n \to \mathbb{R}$ instead of $f : \{0,1\}^n \to \mathbb{R}$ using the bijective transformation $f(x) = h(w)$, for $w = 1 - 2x$. Clearly this transformation preserves the degree of the function.

Let $w_I \in \mathbb{R}^n$ be the characteristic vector of set $I \subseteq [n]$ such that $(w_I)_i = -1$ if $i \in I$ and 1 otherwise. The *Fourier expansion* of the function h takes the following form:

$$h(w) = \sum_{\substack{I\subseteq[n]\\|I|\leq d}} \widehat{h}(I) \prod_{i\in I} w_i,$$

where $\widehat{h}(I)$ it the *Fourier coefficient* of the monomial $\prod_{i\in I} w_i$. Let $g : \{\pm 1\}^n \to \mathbb{R}$ be such that $g := h^2$. Thus $g(w) = \left(\sum_{\substack{I\subseteq[n]\\|I|\leq d}} \widehat{h}(I) \prod_{i\in I} w_i\right)^2$ and its Fourier expansion is of the form:

$$g(w) = \sum_{\substack{I\subseteq[n]\\|I|\leq d}} \left(\widehat{h}(I)\right)^2 + \sum_{\substack{I\neq K\subseteq[n]\\|I|,|K|\leq d}} \widehat{h}(I)\widehat{h}(K) \prod_{i\in I\triangle K} w_i, \tag{3.1}$$

since all monomials squared evaluate to 1 over the $\{\pm 1\}^n$ boolean hypercube.

By [42, Fact 1.12] and Equation 3.1 we know that

$$\widehat{g}(\varnothing) = \mathop{\mathbb{E}}_{w\sim\{\pm1\}^n} g(w) = 2^{-n} \sum_{I\subseteq[n]} g(w_I) = \sum_{\substack{I\subseteq[n]\\|I|\leq d}} \left(\widehat{h}(I)\right)^2 \tag{3.2}$$

Without loss of generality, we assume that the set J in the statement of the Theorem is the empty set. Thus:

$$g(w_\varnothing) = \sum_{\substack{I\subseteq[n]\\|I|\leq d}} \left(\widehat{h}(I)\right)^2 + \sum_{\substack{I\neq K\subseteq[n]\\|I|,|K|\leq d}} \widehat{h}(I)\widehat{h}(K). \tag{3.3}$$

Now, we show that among all boolean functions normalized to have expected value over $\{\pm 1\}^n$ boolean hypercube equal to $\sum_{\substack{I\subseteq[n]\\|I|\leq d}} \left(\widehat{h}(I)\right)^2$ the function $g := h^2$, such that all Fourier

coefficients of h are the same, takes the largest value on the point $w_\varnothing$. Indeed, consider a function g evaluated at point $w_\varnothing$. The RHS of Equation 3.3 satisfies:

$$\sum_{\substack{I,K\subseteq[n]\\|I|,|K|\leq d}} \widehat{h}(I)\widehat{h}(K) \leq \frac{1}{2}\sum_{\substack{I,K\subseteq[n]\\|I|,|K|\leq d}} \left(\widehat{h}^2(I)+\widehat{h}^2(K)\right) = \sum_{i=0}^{d}\binom{n}{i}\sum_{\substack{I\subseteq[n]\\|I|\leq d}}\widehat{h}^2(I)$$

and the equality holds when all Fourier coefficients of h are the same.

Finally, we show that for the function $g = h^2$, such that all Fourier coefficients of h are the same, the Theorem holds. Assume w.l.o.g. that for every $I \subseteq [n]$, $\widehat{h}(I) = 1$. Then we get

$$h^2(w_\varnothing) = g(w_\varnothing) = \left(\sum_{i=0}^{d}\binom{n}{i}\right)^2 \quad \text{and} \quad \sum_{I\subseteq[n]} h^2(w_I) = 2^n \sum_{\substack{I\subseteq[n]\\|I|\leq d}}\left(\widehat{h}(I)\right)^2 = 2^n\sum_{i=0}^{d}\binom{n}{i}.$$

thus the first part of the statement follows.

Note, that, for $J = \varnothing$, the above proof gives an explicit construction of a degree d polynomial that satisfies the inequality in the statement of Lemma 6 with equality. In an analogous way one can construct a function for other sets $J \subseteq [n]$, thus the second part of the claim follows. ◀

Clearly a similar statement holds for functions being a sum of squares of boolean functions.

▶ **Corollary 3.1.** *For every function $s = \sum_i h_i^2$ such that for every i, $h_i : \{0,1\}^n \to \mathbb{R}$ is of degree at most d and every subset $J \subseteq [n]$ the following holds:*

$$\sum_{S\subseteq[n]} s(x_S) \geq \frac{2^n}{\sum_{i=0}^{d}\binom{n}{i}} s(x_J).$$

Now we present the second result in [44, 50, 53]. We start with the following definitions.

A boolean function $f : \{0,1\}^n \to \mathbb{R}$ is *symmetric* if $f(x) = f_i \in \mathbb{R}$ for every $x \in \{0,1\}^n$ such that $|x| = i$. A polynomial $p : \mathbb{R}^n \to \mathbb{R}$ *approximates* a symmetric boolean function f in ℓ_∞-norm within an error c, if $|f(x) - p(x)| \leq c$ for every $x \in \{0,1\}^n$. A symmetric boolean function $f := \{0,1\}^n \to \{0,1\}$ is a *NOR* function if it satisfies $f(0,\ldots,0) = 1$ and $f(x) = 0$ for every other $x \in \{0,1\}^n$.

Now we present the result proved in [44, 50, 53].

▶ **Theorem 7** ([44, 50, 53]). *For every constant $2^{-n} \leq c < 1/2$ the minimum degree of a real polynomial that approximates a NOR boolean function in ℓ_∞-norm within en error c is $\Theta(\sqrt{n} + \sqrt{n\log 1/c})$.*

3.1 Application to the SoS certificates

In this section, we apply Theorem 4, Corollary 3.1 and Theorem 7 to provide lower and upper bounds on the SoS rank for some family of problems. We start with the following definition.

▶ **Definition 8.** *A polynomial f is called a* Single Vertex Cutting *(SVC) constraint if there exists only one $x \in \{0,1\}^n$ such that $f(x) < 0$. A set $\mathcal{G}$ is SVC if all functions $g_1,\ldots,g_m \in \mathcal{G}$ are SVC and every $x \in \{0,1\}^n$ is cut by at most one $g \in \mathcal{G}$. A problem is SVC if its corresponding set $\mathcal{G}$ is SVC.*

Note that all three problems considered in this paper are SVC Problems.

▶ **Theorem 9.** *Consider an SVC system $\mathcal{G}$. For some $f \in \mathbb{R}[x]$, assume that $\mathcal{H}^-(f) = \mathcal{H}^-$, thus $|\mathcal{H}^-(f)| = m$. For every $x_J \in \mathcal{H}^-(f)$ let $g_J \in \mathcal{G}$ be such that $g_J(x_J) < 0$, g_J is unique.*

There is no degree d certificate for f over system $\mathcal{G}$, if for every $c_J \geq 1$, for every J such that $x_J \in \mathcal{H}^-(f)$, the following holds

$$\sum_{x_J \in \mathcal{H}^+(f)} f(x_J) + \sum_{x_J \in \mathcal{H}^-(f)} f(x_J)(1-c_J) < \left(\frac{2^n}{\sum_{k=0}^{d} \binom{n}{k}} - 1 \right) \sum_{x_I \in \mathcal{H}^-(f)} c_I \frac{f(x_I)}{g_I(x_I)} \cdot \min_{\substack{x_J \in \mathcal{H} \\ J \neq I}} g_I(x_J)$$

Proof. Assume, there exists a SoS certificate for f of degree d over $\mathcal{G}$, then

$$f(x) = s_0 + \sum_{x_J \in \mathcal{H}^-} s_J(x) g_J(x), \qquad \text{for every } x \in \{0,1\}^n,$$

for s_J, being SoS of degree at most $2d$, and s_0 being SoS of degree at most $2\left\lceil \frac{2d+deg(\mathcal{G})}{2} \right\rceil$. Consider $x_J \in \mathcal{H}^-(f)$. Since g_J is the only constraint from $\mathcal{G}$ that is negative on this point we get that

$$s_J(x_J) \geq \frac{f(x_J)}{g_J(x_J)}$$

Let $c_J \geq 1$ be such that $s_J(x_J) = c_J \cdot \frac{f(x_J)}{g_J(x_J)}$, we obtain:

$$f(x_J) = \underbrace{s_0(x_J)}_{\geq 0} + \underbrace{s_J(x_J) g_J(x_J)}_{= c_J f(x_J)} + \sum_{\substack{x_I \in \mathcal{H}^- \\ I \neq J}} \underbrace{s_I(x_J) g_I(x_J)}_{\geq 0}.$$

By summing up over all points in $\mathcal{H}^-(f)$ we get:

$$\sum_{x_J \in \mathcal{H}^-(f)} f(x_J) = \underbrace{\sum_{x_J \in \mathcal{H}^-(f)} s_0(x_J)}_{\geq 0} + \underbrace{\sum_{x_J \in \mathcal{H}^-(f)} s_J(x_J) g_J(x_J)}_{= \sum_{x_J \in \mathcal{H}^-(f)} c_J f(x_J)} + \sum_{x_J \in \mathcal{H}^-(f)} \sum_{\substack{x_I \in \mathcal{H}^- \\ I \neq J}} s_I(x_J) g_I(x_J)$$

thus

$$\sum_{x_J \in \mathcal{H}^-(f)} f(x_J)(1 - c_J) = \sum_{x_J \in \mathcal{H}^-(f)} s_0(x_J) + \sum_{x_J \in \mathcal{H}^-(f)} \left(\sum_{\substack{x_I \in \mathcal{H}^- \\ I \neq J}} s_I(x_J) g_I(x_J) \right).$$

Finally, we have

$$\sum_{x_J \in \mathcal{H}^+(f)} f(x_J) + \sum_{x_J \in \mathcal{H}^-(f)} f(x_J)(1 - c_J) = \sum_{x_J \in \mathcal{H}} s_0(x_J) + \sum_{x_J \in \mathcal{H}} \left(\sum_{\substack{x_I \in \mathcal{H}^- \\ I \neq J}} s_I(x_J) g_I(x_J) \right),$$

where

$$\sum_{x_J \in \mathcal{H}} \left(\sum_{\substack{x_I \in \mathcal{H}^- \\ I \neq J}} s_I(x_J) g_I(x_J) \right) \geq \sum_{x_I \in \mathcal{H}^-} \left(\left(\frac{2^n}{\sum_{k=0}^{d} \binom{n}{k}} - 1 \right) s_I(x_I) \left(\min_{\substack{x_J \in \mathcal{H} \\ J \neq I}} g_I(x_J) \right) \right)$$

and the last inequality follows by Corollary 3.1. Since, $\sum_{x_J \in \mathcal{H}} s_0(x_J) \geq 0$ and $s_I(x_I) = c_I \cdot \frac{f(x_I)}{g_I(x_I)}$, the claim follows. ◀

▶ **Theorem 10.** *If every SoS certificate of f over system $\mathcal{G}$ of the form $f = s_0 + \sum_{i=1}^{m} s_i g_i$, is such that the polynomial $s(x) := \sum_{i=1}^{m} s_i(x)$ necessarily has to take at least value $1-c$ for $x = (0,0,\ldots,0)$ and at most value c, for every other $x \in \{0,1\}^n$, for some constant $2^{-n} \leq c < 1/2$, then the degree of the certificate is at least $\Omega\left(\sqrt{n} + \sqrt{n \log 1/c}\right)$.*

Proof. Assume by contradiction that there exists a SoS certificate of degree smaller than $\Omega\left(\sqrt{n} + \sqrt{n \log 1/c}\right)$. Note that s is a real polynomial that approximates the NOR function in ℓ_∞-norm within the constant error c. Since s is of degree smaller than $\Omega\left(\sqrt{n} + \sqrt{n \log 1/c}\right)$ it contradicts Theorem 7. ◀

Finally, we show an argument for an upper bounds on the SoS rank.

▶ **Lemma 11.** *Let f be of degree at most $d+1$ and let g_i for $i \in [m]$ be linear. If there exist SoS polynomials $s_i, \ldots, s_m$ of degree at most d such that*

$$f(x) \geq \sum_{i=1}^{m} s_i(x) g_i(x), \qquad \text{for every } x \in \{0,1\}^n,$$

then there exists a SoS certificate for f of degree $\lceil \frac{n+d}{2} \rceil$.

Proof. Note that $f(x) - \sum_{i=1}^{m} s_i(x) g_i(x) \geq 0$, for every $x \in \{0,1\}^n$. By Theorem 4 and the fact that we consider g_i linear, for $i \in [m]$, we get that there exists a SoS polynomial s_0 of degree at most $\lceil \frac{n+(d+1)-1}{2} \rceil$ such that: $f(x) - \sum_{i=1}^{m} s_i(x) g_i(x) = s_0$. ◀

4 Application to the EIH problem

In this section we show how to use the results presented in Section 3.1 to derive lower and upper bounds on the SoS rank for the Empty Integral Hull problem.

4.1 SoS rank lower bounds

In this section we prove a lower bounds on the SoS rank for the Empty Integral Hull problem.

▶ **Theorem 12.** *The SoS rank for the Empty Integral Hull problem parametrized by constant B, is greater or equal the minimum d, which satisfies:*

$$\frac{B}{B-1} \geq \frac{2^n}{\sum_{k=0}^{d} \binom{n}{k}}.$$

Proof. We directly apply Theorem 9. We want to prove that for $f(x) = -1$ there is no SoS certificate of degree d, for d such that $\frac{B}{B-1} < \frac{2^n}{\sum_{k=0}^{d} \binom{n}{k}}$. Following the notation from Theorem 9 note that for every $g_I \in \mathcal{G}$, the smallest nonnegative value over the hypercube $\{0,1\}^n$ of g_I is $1 - 1/B$. By Theorem 9 there is no degree d SoS certificate for $f = -1$, if for every $c_I \geq 1$, $I \subseteq [n]$, the following is satisfied:

$$(-1) \sum_{I \subseteq [n]} (1 - c_I) < \left(\frac{2^n}{\sum_{k=0}^{d} \binom{n}{k}} - 1 \right) \sum_{I \subseteq [n]} c_I \frac{-1}{-\frac{1}{B}} \cdot \left(1 - \frac{1}{B}\right).$$

Let $c = 1/2^n \sum_{I \subseteq [n]} c_I$. The above is satisfied if

$$\frac{c-1}{c} < \left(\frac{2^n}{\sum_{k=0}^{d} \binom{n}{k}} - 1 \right) (B-1).$$

Since $c \geq 1$, we have $\frac{c-1}{c} < 1$ and the above inequality holds if $\frac{1}{B-1} < \left(\frac{2^n}{\sum_{k=0}^{d} \binom{n}{k}} - 1 \right)$. ◀

We get the following corollary.

▶ **Corollary 4.1.** *The SoS rank for the EIH problem with $B = 2$, is at least $\lceil n/2 \rceil$.*

Proof. By Theorem 12 the SoS rank of the EIH for $B = 2$ is greater or equal to the minimum d that satisfies: $2 \geq \frac{2^n}{\sum_{k=0}^{d} \binom{n}{k}}$. which holds for $d \geq \lceil n/2 \rceil$. ◀

One can apply Theorem 12 to easily reprove the result from [31], saying that for $B = 2^{n+1}$ the SoS rank is at least n.

▶ **Corollary 4.2.** *For $B = 2^{n+1}$ the EIH problem has the SoS rank n.*

4.2 SoS rank upper bounds

In this section we prove an upper bound on the SoS rank for the Empty Integral Hull problem.

▶ **Lemma 13.** *The SoS rank for the Empty Integral Hull problem parametrized by the constant B, is less or equal the minimum d, that satisfies*

$$\frac{Bn}{Bn-1} > \frac{2^n}{\sum_{k=0}^{d} \binom{n}{k}}.$$

Proof. We follow the notation and reasoning from Theorem 9. Let $c_I \geq 1$, for $I \subseteq [n]$ be large enough constants, whose value is specified later.

Our goal is to construct a SoS certificate for $f = -1$ using the SoS polynomials of degree at most $2d$, for d given in the statement of this lemma. In our construction we take $s_0 = 0$ and consider s_J, for $J \subseteq [n]$, to be 2^n polynomials squared, each of which constructed as in Lemma 6, such that for every $I, J \subseteq [n]$ we have $\sum_{K \subseteq [n]} s_I(x_K) g_I(x_K) = \sum_{K \subseteq [n]} s_J(x_K) g_J(x_K)$.

We want to prove that there exists a certificate of the form

$$-1 = f(x_J) = \underbrace{s_J(x_J) g_J(x_J)}_{= c_J f(x_J) = -c_J} + \sum_{\substack{x_I \in \mathcal{H} \\ I \neq J}} \underbrace{s_I(x_J) g_I(x_J)}_{\geq 0}, \qquad \text{for every } J \subseteq [n], \tag{4.1}$$

where, for every $J \subseteq [n]$, by the construction of the polynomial s_J, we know that:

$$\left(\frac{2^n}{\sum_{k=0}^{d} \binom{n}{k}} - 1 \right) s_J(x_J) \left(\underbrace{\min_{\substack{x_I \in \mathcal{H} \\ I \neq J}} g_J(x_I)}_{= 1 - 1/B} \right) \leq \sum_{\substack{x_I \in \mathcal{H} \\ I \neq J}} \underbrace{s_I(x_J) g_I(x_J)}_{\geq 0} \leq \left(\frac{2^n}{\sum_{k=0}^{d} \binom{n}{k}} - 1 \right) s_J(x_J) \left(\underbrace{\max_{\substack{x_I \in \mathcal{H} \\ I \neq J}} g_J(x_I)}_{= n - 1/B} \right).$$

Let

$$\sum_{\substack{x_I \in \mathcal{H} \\ I \neq J}} \underbrace{s_I(x_J) g_I(x_J)}_{\geq 0} = \alpha \left(1 - \frac{1}{B} \right) \left(\frac{2^n}{\sum_{k=0}^{d} \binom{n}{k}} - 1 \right) s_J(x_J), \qquad \text{for every } J \subseteq [n],$$

for some $1 \leq \alpha \leq \frac{n - 1/B}{1 - 1/B}$. Since $s_J(x_J) = c_J \cdot f(x_J)/g_J(x_J) = c_J \cdot \frac{-1}{-1/B} = c_J \cdot B$, thus satisfying Equation 4.1 requires:

$$\frac{c_J - 1}{c_J} = \alpha \left(\frac{2^n}{\sum_{k=0}^{d} \binom{n}{k}} - 1 \right) (B - 1), \qquad \text{for every } J \subseteq [n].$$

Now, put all c_J equal and note that c_J can be chosen arbitrarily, so there exists SoS certificate of degree d if we can satisfy,

$$\alpha\left(\frac{2^n}{\sum_{k=0}^{d}\binom{n}{k}}-1\right)(B-1)<1.$$

which holds if

$$\frac{n-1/B}{1-1/B}(B-1)\left(\frac{2^n}{\sum_{k=0}^{d}\binom{n}{k}}-1\right)=(Bn-1)\left(\frac{2^n}{\sum_{k=0}^{d}\binom{n}{k}}-1\right)<1.$$

Finally, the above is satisfied if $\frac{2^n}{\sum_{k=0}^{d}\binom{n}{k}}<\frac{Bn}{Bn-1}$. ◀

For $B=2$ we get the following corollary.

▶ **Corollary 4.3.** *The SoS rank for the EIH problem for $B=2$ is at most $\lceil \frac{n}{2}+\sqrt{n\log 2n}\ \rceil$.*

Proof. By Lemma 13 we know that the SoS rank is smaller or equal to the smallest d that satisfies

$$\frac{2^n}{\sum_{k=0}^{d}\binom{n}{k}}<\frac{2n}{2n-1}.$$

Since $2^n=\sum_{k=0}^{d}\binom{n}{k}+\sum_{k=d+1}^{n}\binom{n}{k}$ we get the requirement

$$\frac{\sum_{k=d+1}^{n}\binom{n}{k}}{2^n-\sum_{k=d+1}^{n}\binom{n}{k}}<\frac{2n}{2n-1}-1$$

and finally

$$\sum_{k=d+1}^{n}\binom{n}{k}<\frac{2^n}{2n}.$$

By the standard Chernoff bound, see e.g. [10], for $d>n/2$ we know that $\sum_{k=d+1}^{n}\binom{n}{k}=\sum_{k=0}^{n-d-1}\binom{n}{k}\le 2^n\exp\frac{-(2d+2-n)^2}{4n}$. Thus for $d>\frac{1}{2}\left(n-2+2\sqrt{n\log 2n}\right)$ there exists a SoS certificate for EIH problem for $B=2$. ◀

5 Application to the MK problem

In this section we prove SoS rank lower and upper bounds for the MK problem.

5.1 SoS rank lower bound

We start with presenting the lower bound for the MK Problem.

▶ **Lemma 14.** *The SoS rank lower bound for the MK problem for is at least $\Omega(\sqrt{n}+\sqrt{n\log P})$.*

Proof. By contradiction, assume there exists a SoS certificate of degree smaller than $\Omega(\sqrt{n}+\sqrt{n\log P})$ for the function $\sum_{i=1}^{n}x_i-1$, namely

$$\sum_{i=1}^{n}x_i-1=s_0(x)+s_1(x)\left(\sum_{i=1}^{x}x_i-\frac{1}{P}\right),\qquad\text{for every } x\in\{0,1\}^n.$$

for s_0 and s_1 SoS of degree smaller than $\Omega(\sqrt{n}+\sqrt{n\log P})$. Since $s_0(x)\geq 0$, for every $x\in\{0,1\}^n$, we know that

$$s_1(0,\ldots,0)\geq \frac{-1}{-\frac{1}{P}}=P \qquad \text{and} \qquad s_1(x)\leq \frac{\sum_{i=1}^n x_i-1}{\sum_{i=1}^x x_i-\frac{1}{P}}\leq 1 \text{ for every other } x\in\{0,1\}^n.$$

This implies the existence of the function $\overline{s}(x)=s(x)/P$ of degree smaller than $\Omega(\sqrt{n}+\sqrt{n\log P})$ that approximates the NOR function within ℓ_∞-norm within the error $1/P$ which contradicts Theorem 10. ◀

A direct application of Lemma 14 gives the following corollary:

▶ **Corollary 5.1.** *The SoS rank lower bound for MK problem for $P=2$ is at least $\Omega(\sqrt{n})$.*

5.2 SoS rank upper bounds

In this section we prove an upper bound on the SoS rank for the MK problem for $P=2$. We start with proving the following Lemma.

▶ **Lemma 15.** *For every $n\in\mathbb{N}$ there exists a function $f:\{0,1\}^n\to\mathbb{R}$ of degree $2\lceil\sqrt{n}\,\rceil$ such that $f(0,\ldots,0)\geq 2$, $f(x)=0$ for every $x\in\{0,1\}^n$ such that $|x|=1$ and for every other $x\in\{0,1\}^n$ the function takes the value $|f(x)|\leq 1$.*

Proof. Our construction provides a symmetric polynomial thus in the following we construct a univariate polynomial $g:\mathbb{R}\to\mathbb{R}$ such that $f(x_1,\ldots,x_n):=g(\sum_{i=1}^n x_i)$ satisfies the claimed properties. We start with presenting steps for constructing polynomial g.

Step 1: Consider a Chebyshev polynomial $T_d(x):[-1,1]\to[-1,1]$ of degree $d=2\lceil\sqrt{n}\,\rceil$. Prove that the smallest root (left-most one) appears before point $x=-1+1/n$.

Step 2: Shift and spread the polynomial $T_d(x)$ such that the domain is $[0,n]$, namely consider $T_d^{'}(x)=T_d(2x/n-1)$. Note that the smallest zero of $T_d^{'}$ appears before point $x=1/2$.

Step 3: Extend the domain of $T_d^{'}(x)$ to be $[-1/2,n]$ and then prove that $T_d^{'}(-1/2)=T_d(-1-1/n)$ takes the absolute value at least 2.

Step 4: Consider a polynomial $T_d^{''}$ obtained from shifting the polynomial $T_d^{'}$ to the right such that the smallest zero is exactly at point $x=1$. Note that $g:=T_d^{''}$ satisfies all the required properties.

It remains to prove the claims from Step 1 and 3. Let us start with the claim from Step 1. It is known that Chebyshev polynomial of degree d, $T_d(x)$, has zeros at points $\cos\left(\frac{2j-1}{d}\frac{\pi}{2}\right)$ for $j=1,\ldots,d$ see e.g. [47][Equation 1.17], thus smallest zero is at the point $\cos\left(\pi-\frac{\pi}{2d}\right)$. We consider $d=2\lceil\sqrt{n}\,\rceil$, thus it is sufficient to prove that $\cos\left(\pi-\frac{\pi}{4\sqrt{n}}\right)+1-\frac{1}{n}\leq 0$ for every $n\in\mathbb{R}_+$. To do so, consider the function

$$h(z)=\cos\left(\pi-\frac{\pi}{4z}\right)+1-\frac{1}{z^2}$$

and note that $h(z)$ is increasing in z for $z\in\mathbb{R}_+$. Indeed the derivative of $h(z)$ is equal to

$$h^{'}(z)=\frac{2}{z^3}-\frac{\pi}{4z^2}\sin\left(\pi-\frac{\pi}{4z}\right)=\frac{8-\pi z\sin\left(\frac{\pi}{4z}\right)}{4z^3}.$$

Since, for all $z\in\mathbb{R}_+$, $\sin z\leq z$, it is enough to show that $8-\pi z\frac{\pi}{4z}\geq 0$, which is clearly the case. Next we show that $\lim_{z\to\infty}h(z)=0$. Indeed, by the algebraic limit theorem

$$\lim_{z\to\infty}\left(\cos\left(\pi-\frac{\pi}{4z}\right)+1-\frac{1}{z^2}\right)=\lim_{z\to\infty}\cos\left(\pi-\frac{\pi}{4z}\right)-\lim_{z\to\infty}\frac{1}{z^2}+1=0.$$

It remains to prove the claim from Step 3. To do so, we use the characterization of Chebyshev polynomial given in [47, Equation 1.12]:

$$T_d(x) = \frac{1}{2}\left(\left(x - \sqrt{x^2-1}\right)^d + \left(\sqrt{x^2-1} + x\right)^d\right).$$

For $d = 2\lceil\sqrt{n}\rceil$ and $x = -1 - 1/n$ we get:

$$T_{2\lceil\sqrt{n}\rceil}\left(-1-\frac{1}{n}\right) \geq \frac{1}{2}\left(\left(-1-\frac{1}{n}-\sqrt{\left(-1-\frac{1}{n}\right)^2-1}\right)^{2\lceil\sqrt{n}\rceil}\right)$$
$$\geq \frac{1}{2}\left(\left(-1-\frac{\sqrt{2}}{\sqrt{n}}\right)^{2\sqrt{n}}\right)$$

where $\frac{1}{2}\left(\left(-1-\frac{\sqrt{2}}{\sqrt{n}}\right)^{2\sqrt{n}}\right)$ is increasing in n and for $n = 1$ takes value bigger than 2. ◀

Finally, we prove the SoS rank upper bound for the MK problem for $P = 2$.

▶ **Lemma 16.** *The SoS rank for the MK for $P = 2$ is at most $d = \lceil\frac{n+4\lceil\sqrt{n}\rceil}{2}\rceil$.*

Proof. We show that there exist a SoS certificate of degree $d = \lceil\frac{n+4\lceil\sqrt{n}\rceil}{2}\rceil$ for the function $\sum_{i=1}^m x_i - 1$. Consider the polynomial f of degree $2\lceil\sqrt{n}\rceil$ constructed in Lemma 15. Take $s_1 = f^2$ and note that $\sum_{i=1}^n x_i - 1 - 0.8f^2(x)\left(\sum_{i=1}^m x_i - 1/2\right) \geq 0$ for every $x \in \{0,1\}^n$, since $(0.8 \cdot 2)^2 > 2$ and $0.8^2 < 2/3$. Applying Theorem 4 we get that there exists a SoS certificate for $\sum_{i=1}^n -1$ of degree $d = \lceil\frac{n+4\lceil\sqrt{n}\rceil}{2}\rceil$. ◀

6 Application to the SC problem

In the last section we prove the SoS rank lower bound for the SC problem.

▶ **Lemma 17.** *The SoS rank for the SC problem is at least $\Omega(\sqrt{n})$.*

Proof. By contradiction, assume there exists a SoS certificate of degree smaller than $\Omega(\sqrt{n})$ for the function $\sum_{i=1}^n x_i - 2$, namely: $\sum_{i=1}^n x_i - 2 = s_0(x) + \sum_{j=1}^n s_j(x)\left(\sum_{\substack{i=1\\i\neq j}}^n x_i - 1\right)$. We follow the idea in the proof of Lemma 14. Let $s(x) = \sum_{i=1}^n s_i(x)$ and take $k = \lfloor n/3\rfloor$. For every $j \in [n]$ and $x \in \{0,1\}^n$ such that $|x| \geq 3$ we know that $\sum_{\substack{i=1\\i\neq j}}^n x_i - 1 \geq \sum_{i=1}^n x_i - 2$. Thus, note that for every $\ell \in [k]$ and for every $x \in \{0,1\}^n$ such that $|x| = 3\ell$ we have:

$$s(0,\ldots,0) \geq 2 \qquad \text{and} \qquad s(x) \leq 1, \qquad \text{for every } x \in \{0,1\}^n, \text{ s.t. } |x| = 3\ell, \text{ for } \ell \in [k].$$

Now consider a symmetric function $\overline{s}$ over k-dimensional boolean hypercube $\{0,1\}^k$, $\overline{s} := \{0,1\}^k \to \mathbb{R}$ that takes the values:

$$\overline{s}(z) := \binom{n}{3\ell}^{-1} \sum_{\substack{x\in\{0,1\}^n\\|x|=3\ell}} s(x), \qquad \text{for every } z \in \{0,1\}^k,\ |z| = \ell.$$

Clearly, $deg(\overline{s})$ is at most $deg(s)$ and note that it holds:

$$\overline{s}(0,\ldots,0) \geq 2 \qquad \text{and} \qquad \overline{s}(z) \leq 1 \qquad \text{for every other } z \in \{0,1\}^k$$

and the function $\overline{s}(z)/3$ is of degrees smaller than $\Omega(\sqrt{n})$ and approximates the NOR function in ℓ_∞-norm within the error $1/3$, which contradicts Theorem 10. ◀

References

1 S. Arora, S. Rao, and U. V. Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56(2):5:1–5:37, 2009.

2 B. Barak, S. B. Hopkins, J. A. Kelner, P. Kothari, A. Moitra, and A. Potechin. A Nearly Tight Sum-of-Squares Lower Bound for the Planted Clique Problem. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 428–437, 2016.

3 B. Barak, J. A. Kelner, and D. Steurer. Dictionary Learning and Tensor Decomposition via the Sum-of-Squares Method. In *STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 143–151, 2015.

4 B. Barak and A. Moitra. Noisy Tensor Completion via the Sum-of-Squares Hierarchy. In *COLT 2016, New York, USA, June 23-26, 2016*, pages 417–445, 2016.

5 B. Barak, P. Raghavendra, and D. Steurer. Rounding Semidefinite Programming Hierarchies via Global Correlation. In *FOCS*, pages 472–481, 2011.

6 B. Barak and D. Steurer. Proofs, beliefs, and algorithms through the lens of sum-of-squares, 2016. URL: `https://www.sumofsquares.org`.

7 M. Hossein Bateni, M. Charikar, and V. Guruswami. MaxMin allocation via degree lower-bounded arborescences. In *STOC*, pages 543–552, 2009.

8 D. Bienstock and M. Zuckerberg. Subset Algebra Lift Operators for 0-1 Integer Programming. *SIAM Journal on Optimization*, 15(1):63–95, 2004.

9 S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

10 H. Chernof. A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations. *Ann. Math. Statist.*, 23(4):493–507, December 1952.

11 K. K. H. Cheung. Computation of the Lasserre Ranks of Some Polytopes. *Math. Oper. Res.*, 32(1):88–94, 2007.

12 E. Chlamtac. Approximation Algorithms Using Hierarchies of Semidefinite Programming Relaxations. In *FOCS*, pages 691–701, 2007.

13 E. Chlamtac and G. Singh. Improved Approximation Guarantees through Higher Levels of SDP Hierarchies. In *APPROX-RANDOM*, pages 49–62, 2008.

14 E. Chlamtac and M. Tulsiani. *Convex Relaxations and Integrality Gaps*, pages 139–169. Springer US, Boston, MA, 2012.

15 W. Cook and S. Dash. On the matrix-cut rank of polyhedra. *Math. Oper. Res.*, 26(1):19–30, 2001.

16 G. Cornuéjols and Y. Li. On the Rank of Mixed 0, 1 Polyhedra. In *Proceedings of the 8th International Conference on Integer Programming and Combinatorial Optimization, IPCO 2001, Utrecht, The Netherlands*, pages 71–77, 2001.

17 M. Cygan, F. Grandoni, and M. Mastrolilli. How to Sell Hyperedges: The Hypermatching Assignment Problem. In *SODA*, pages 342–351, 2013.

18 W. F. de la Vega and C. Kenyon-Mathieu. Linear programming relaxations of maxcut. In *SODA*, pages 53–61, 2007.

19 M. X. Goemans and L. Tunçel. When Does the Positive Semidefiniteness Constraint Help in Lifting Procedures? *Math. Oper. Res.*, 26(4):796–815, 2001.

20 M. X. Goemans and D. P. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. Assoc. Comput. Mach.*, 42(6):1115–1145, 1995.

21 D. Grigoriev. Complexity of Positivstellensatz proofs for the knapsack. *Comput. Complexity*, 10(2):139–154, 2001.

22 D. Grigoriev, E. A. Hirsch, and D. V. Pasechnik. Complexity of Semi-algebraic Proofs. In *STACS*, pages 419–430, 2002.

23 D. Grigoriev and N. Vorobjov. Complexity of Null-and Positivstellensatz proofs. *Ann. Pure App. Logic*, 113(1-3):153–160, 2001.

24 V. Guruswami and A. K. Sinop. Lasserre hierarchy, higher eigenvalues, and approximation schemes for graph partitioning and quadratic integer programming with psd objectives. In *FOCS*, pages 482–491, 2011.

25 S. B. Hopkins, T. Schramm, J. Shi, and D. Steurer. Fast spectral algorithms from sum-of-squares proofs: tensor decomposition and planted sparse vectors. In *STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 178–191, 2016.

26 P. Kothari, J. Steinhardt, and D. Steurer. Robust Moment Estimation and Improved Clustering via Sum of Squares. In *STOC 2018*, 2018.

27 P. K. Kothari, R. Mori, R. O'Donnell, and D. Witmer. Sum of squares lower bounds for refuting any CSP. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 132–145, 2017.

28 A. Kurpisz, S. Leppänen, and M. Mastrolilli. Sum-of-Squares Hierarchy Lower Bounds for Symmetric Formulations. In *Integer Programming and Combinatorial Optimization - 18th International Conference, IPCO 2016, Liège, Belgium, June 1-3, 2016, Proceedings*, pages 362–374, 2016.

29 A. Kurpisz, S. Leppänen, and M. Mastrolilli. Tight Sum-Of-Squares Lower Bounds for Binary Polynomial Optimization Problems. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 78:1–78:14, 2016.

30 A. Kurpisz, S. Leppänen, and M. Mastrolilli. An unbounded Sum-of-Squares hierarchy integrality gap for a polynomially solvable problem. *Math. Program.*, 166(1-2):1–17, 2017.

31 A. Kurpisz, S. Leppänen, and M. Mastrolilli. On the Hardest Problem Formulations for the 0/1 Lasserre Hierarchy. *Math. Oper. Res.*, 42(1):135–143, 2017.

32 J. B. Lasserre. An Explicit Exact SDP Relaxation for Nonlinear 0-1 Programs. In *Integer Programming and Combinatorial Optimization, 8th International IPCO Conference, Utrecht, The Netherlands, June 13-15, 2001, Proceedings*, pages 293–303, 2001.

33 M. Laurent. A Comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre Relaxations for 0-1 Programming. *Math. Oper. Res.*, 28(3):470–496, 2003.

34 M. Laurent. Lower Bound for the Number of Iterations in Semidefinite Hierarchies for the Cut Polytope. *Math. Oper. Res.*, 28(4):871–883, 2003.

35 M. Laurent. Sums of squares, moment matrices and optimization over polynomials. In *Emerging applications of algebraic geometry*, volume 149 of *IMA Vol. Math. Appl.*, pages 157–270. Springer, New York, 2009.

36 J. R. Lee, P. Raghavendra, and D. Steurer. Lower Bounds on the Size of Semidefinite Programming Relaxations. In *STOC*, pages 567–576, 2015.

37 T. Lee, A. Prakash, R. Wolf, and H. Yuen. On the Sum-of-Squares Degree of Symmetric Quadratic Functions. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 17:1–17:31, 2016.

38 A. Magen and M. Moharrami. Robust Algorithms for on Minor-Free Graphs Based on the Sherali-Adams Hierarchy. In *APPROX-RANDOM*, pages 258–271, 2009.

39 M. Mastrolilli. High Degree Sum of Squares Proofs, Bienstock-Zuckerberg Hierarchy and CG Cuts. In *Integer Programming and Combinatorial Optimization - 19th International Conference, IPCO 2017, Waterloo, ON, Canada, June 26-28, 2017, Proceedings*, pages 405–416, 2017.

40 Raghu Meka, Aaron Potechin, and Avi Wigderson. Sum-of-squares Lower Bounds for Planted Clique. In *Proceedings of the 47th Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA*, pages 87–96, 2015.

41 Y. Nesterov. *Global quadratic optimization via conic relaxation*, pages 363–384. Kluwer Academic Publishers, 2000.

42 R. O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

43 P. Parrilo. *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD thesis, California Institute of Technology, 2000.

44 R. Paturi. On the Degree of Polynomials that Approximate Symmetric Boolean Functions (Preliminary Version). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 468–474, 1992.

45 A. Potechin and D. Steurer. Exact tensor completion with sum-of-squares. In *COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, pages 1619–1673, 2017.

46 P. Raghavendra and N. Tan. Approximating CSPs with global cardinality constraints using SDP hierarchies. In *SODA*, pages 373–387, 2012.

47 T. Rivlin. The chebyshev polynomials. *SERBIULA (sistema Librum 2.0)*, February 1974.

48 S. Sakaue, A. Takeda, S. Kim, and N. Ito. Exact Semidefinite Programming Relaxations with Truncated Moment Matrix for Binary Polynomial Optimization Problems. *SIAM Journal on Optimization*, 27(1):565–582, 2017.

49 T. Schramm and D. Steurer. Fast and robust tensor decomposition with applications to dictionary learning. In *COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, pages 1760–1793, 2017.

50 Alexander A. Sherstov. Approximate Inclusion-Exclusion for Arbitrary Symmetric Functions. *Computational Complexity*, 18(2):219–247, 2009. doi:10.1007/s00037-009-0274-4.

51 N. Shor. Class of global minimum bounds of polynomial functions. *Cybernetics*, 23(6):731–734, 1987.

52 J. Thapper and S. Zivny. The Power of Sherali-Adams Relaxations for General-Valued CSPs. *SIAM J. Comput.*, 46(4):1241–1279, 2017.

53 R. Wolf. A note on quantum algorithms and the minimal degree of ϵ-error polynomials for symmetric functions. *Quantum Information & Computation*, 8(10):943–950, 2010.

Dynamic Time Warping in Strongly Subquadratic Time: Algorithms for the Low-Distance Regime and Approximate Evaluation

William Kuszmaul
Massachusetts Institute of Technology, Cambridge, USA
kuszmaul@mit.edu

Abstract

Dynamic time warping distance (DTW) is a widely used distance measure between time series, with applications in areas such as speech recognition and bioinformatics. The best known algorithms for computing DTW run in near quadratic time, and conditional lower bounds prohibit the existence of significantly faster algorithms.

The lower bounds do not prevent a faster algorithm for the important special case in which the DTW is small, however. For an arbitrary metric space Σ with distances normalized so that the smallest non-zero distance is one, we present an algorithm which computes $\text{dtw}(x, y)$ for two strings x and y over Σ in time $O(n \cdot \text{dtw}(x, y))$. When $\text{dtw}(x, y)$ is small, this represents a significant speedup over the standard quadratic-time algorithm.

Using our low-distance regime algorithm as a building block, we also present an approximation algorithm which computes $\text{dtw}(x, y)$ within a factor of $O(n^\epsilon)$ in time $\tilde{O}(n^{2-\epsilon})$ for $0 < \epsilon < 1$. The algorithm allows for the strings x and y to be taken over an arbitrary well-separated tree metric with logarithmic depth and at most exponential aspect ratio. Notably, any polynomial-size metric space can be efficiently embedded into such a tree metric with logarithmic expected distortion. Extending our techniques further, we also obtain the first approximation algorithm for edit distance to work with characters taken from an *arbitrary metric space*, providing an n^ϵ-approximation in time $\tilde{O}(n^{2-\epsilon})$, with high probability.

Finally, we turn our attention to the relationship between edit distance and dynamic time warping distance. We prove a reduction from computing edit distance over an arbitrary metric space to computing DTW over the same metric space, except with an added null character (whose distance to a letter l is defined to be the edit-distance insertion cost of l). Applying our reduction to a conditional lower bound of Bringmann and Künnemann pertaining to edit distance over $\{0, 1\}$, we obtain a conditional lower bound for computing DTW over a three letter alphabet (with distances of zero and one). This improves on a previous result of Abboud, Backurs, and Williams, who gave a conditional lower bound for DTW over an alphabet of size five.

With a similar approach, we also prove a reduction from computing edit distance (over generalized Hamming Space) to computing longest-common-subsequence length (LCS) over an alphabet with an added null character. Surprisingly, this means that one can recover conditional lower bounds for LCS directly from those for edit distance, which was not previously thought to be the case.

2012 ACM Subject Classification Theory of computation; Theory of computation → Design and analysis of algorithms

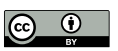
Keywords and phrases dynamic time warping, edit distance, approximation algorithm, tree metrics

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.80

Category Track A: Algorithms, Complexity and Games

Related Version https://arxiv.org/abs/1904.09690

Funding *William Kuszmaul*: Supported by an MIT Akamai Fellowship and a Fannie & John Hertz Foundation Fellowship. Also supported by NSF Grants 1314547 and 1533644. Parts of this research were performed during the Stanford CURIS research program.



46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 80; pp. 80:1–80:15

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

Acknowledgements The author would like to thank Moses Charikar for his mentoring and advice throughout the project, Ofir Geri for his support and for many useful conversations, and Virginia Williams for suggesting the problem of reducing between edit distance and LCS.

1 Introduction

Dynamic Time Warping distance (DTW) is a widely used distance measure between time series. DTW is particularly flexible in dealing with temporal sequences that vary in speed. To measure the distance between two sequences, portions of each sequence are allowed to be warped (meaning that a character may be replaced with multiple consecutive copies of itself), and then the warped sequences are compared by summing the distances between corresponding pairs of characters. DTW's many applications include phone authentication [19], signature verification [35], speech recognition [34], bioinformatics [1], cardiac medicine [15], and song identification [44].

The textbook dynamic-programming algorithm for DTW runs in time $O(n^2)$, which can be prohibitively slow for large inputs. Moreover, conditional lower bounds [13, 3] prohibit the existence of a strongly subquadratic-time algorithm[1], unless the Strong Exponential Time Hypothesis is false.

The difficulty of computing DTW directly has motivated the development of fast heuristics [39, 28, 29, 27, 11, 38] which typically lack provable guarantees.

On the theoretical side, researchers have considered DTW in the contexts of locality sensitive hashing [20] and nearest neighbor search [21], but very little additional progress has been made on the general problem in which one is given two strings x and y with characters from a metric space Σ, and one wishes to compute (or approximate) $\mathrm{dtw}(x, y)$.

To see the spectrum of results one might aim for, it is helpful to consider edit distance, another string-similarity measure, defined to be the minimum number of insertions, deletions, and substitutions needed to get between two strings. Like DTW, edit distance can be computed in time $O(n^2)$ using dynamic programming [42, 37, 41], and conditional lower bounds suggest that no algorithm can do significantly better [7, 13]. This has led to researchers focusing on specialized versions of the problem, especially in two important directions:

- **Low-Distance Regime Algorithms:** In the special case where the edit distance between two strings is small (less than $\sqrt{n}$), the algorithm of Landau, Myers and Schmidt [33] can be used to compute the exact distance in time $O(n)$. In general, the algorithm runs in time $O(n + \mathrm{ed}(x, y)^2)$. Significant effort has also been made to design variants of the algorithm which exhibit small constant overhead in practice [17].
- **Approximation Algorithms:** Andoni, Krauthgamer and Onak introduced an algorithm estimating edit distance within a factor of $(\log n)^{O(1/\varepsilon)}$ in time $O(n^{1+\varepsilon})$ [5], culminating a long line of research on approximation algorithms that run in close to linear time [6, 10, 9, 18]. Recently, Chakraborty et al. gave the first strongly subquadratic algorithm to achieve a *constant* approximation, running time $O(n^{12/7})$ [16].

This paper presents the first theoretical results in these directions for DTW.

[1] An algorithm is said to run in strongly subquadratic time if it runs in time $O(n^{2-\epsilon})$ for some constant $\epsilon > 0$. Although strongly subquadratic time algorithms are prohibited by conditional lower bounds, runtime improvements by subpolynomial factors are not. Such improvements have been achieved [25].

A Low-Distance Regime Algorithm for DTW (Section 3)

We present the first algorithm for computing DTW in the low-distance regime. Our algorithm computes $\mathrm{dtw}(x, y)$ in time $O(n \cdot \mathrm{dtw}(x, y))$ for strings x and y with characters taken from an arbitrary metric space in which the minimum non-zero distance is one. The key step in our algorithm is the design of a new dynamic-programming algorithm for DTW, which lends itself especially well to the low-distance setting.

Our dynamic program relies on a recursive structure in which the two strings x and y are treated asymmetrically within each subproblem: One of the strings is considered as a sequence of letters, while the other string is considered as a sequence of runs (of equal letters). The subproblems build on one-another in a way so that at appropriate points in the recursion, we toggle the role that the two strings play. The asymmetric treatment of the strings limits the number of subproblems that can have return-values less than any given threshold K to $O(nK)$, allowing for a fast algorithm in the low-distance setting.

We remark that the requirement of having the smallest distance between distinct characters be 1 is necessary for the low-distance regime algorithm to be feasible, since otherwise distances can simply be scaled down to make every DTW instance be low-distance.

Approximating DTW Over Well Separated Tree Metrics (Section 4)

We design the first approximation algorithm for DTW to run in strongly subquadratic time. Our algorithm computes $\mathrm{dtw}(x, y)$ within an n^ϵ-approximation in time $\tilde{O}(n^{2-\epsilon})$. The algorithm allows for the strings x and y have characters taken from an arbitrary well-separated tree metric of logarithmic depth and at most exponential aspect ratio.[2] These metric spaces are universal in the sense that any finite metric space M of polynomial size can be efficiently embedded into a well-separated tree metric with expected distortion $O(\log |M|)$ and logarithmic depth [22, 8].

An important consequence of our approximation algorithm is for the special case of DTW over the reals. Exploiting a folklore embedding from $\mathbb{R}$ to a well-separated tree metric metric, we are able to obtain with high probability an $O(n^\epsilon)$-approximation for $\mathrm{dtw}(x, y)$ in time $\tilde{O}(n^{2-\epsilon})$, for any strings x and y of length at most n over a subset of the reals with a polynomial aspect ratio.

In the special case of DTW over the reals, previous work has been done to find approximation algorithms under certain geometric assumptions about the inputs x and y [4, 43]. To the best of our knowledge, our approximation algorithm is the first to not rely on any such assumptions.

It is interesting to note that our results on low-distance regime and approximation algorithms for DTW have bounds very similar to the earliest results for edit distance in the same directions. Indeed, the first algorithm to compute edit distance in the low-distance regime [23] exploited properties of a (now standard) dynamic-programming algorithm in order to compute $\mathrm{ed}(x, y)$ in time $O(n \cdot \mathrm{ed}(x, y))$. This implicitly resulted in the first approximation algorithm for edit distance, allowing one to compute an $O(n^\epsilon)$-approximation in time $O(n^{2-\epsilon})$. Until the work of [6] and [5], which culminated in an algorithm with a polylogarithmic approximation ratio, the best known approximation ratio for edit distance remained polynomial for roughly twenty years [33, 9, 10].

[2] The *aspect* ratio of a metric space is the ratio between the largest and smallest non-zero distances in the space.

The $\tilde{O}(n^{2-\epsilon})$-time $O(n^{\epsilon})$-approximation tradeoff is also the current state-of-the-art for another related distance measurement known as Fréchet distance [14], and is achieved using an algorithm that differs significantly from its edit-distance and DTW counterparts.

Reduction from Edit Distance to DTW (Section 5)

We show that the similarity between our results for DTW and the earliest such results for edit distance is not coincidental. In particular, we prove a simple reduction from computing edit distance over an arbitrary metric space to computing DTW over the same metric space (with an added null character). Consequently, any algorithmic result for computing DTW in the low-distance regime or approximating DTW immediately implies the analogous result for edit distance. The opposite direction is true for lower bounds. For example, the conditional lower bound of Bringmann and Künnemann [13], which applies to edit distance over the alphabet $\{0, 1\}$, now immediately implies a conditional lower bound for DTW over an alphabet of size three (in which characters are compared with distances zero and one). This resolves a direction of work posed by Abboud, Backurs, and Williams [3], who gave a conditional lower bound for DTW over an alphabet of size five, and noted that if one could prove the same lower bound for an alphabet of size three, then the runtime complexity of DTW over generalized Hamming space would be settled (modulo the Strong Exponential Time Hypothesis). Indeed, it is known that over an alphabet of size two, DTW can be computed in strongly subquadratic time [3].

Using a similar approach we also prove a simple reduction from computing edit distance (over generalized Hamming space) to computing the longest-common-subsequence length (LCS) between two strings. Thus conditional lower bounds for computing edit distance directly imply conditional lower bounds for computing LCS (over an alphabet with one additional character). This was not previously though to be the case. Indeed, the first known conditional lower bounds for LCS came after those for edit distance [2, 7], and it was noted by Abboud et al. [2] that *"A simple observation is that the computation of the LCS is equivalent to the computation of the Edit-Distance when only deletions and insertions are allowed, but no substitutions. Thus, intuitively, LCS seems like an easier version of Edit Distance, since a solution has fewer degrees of freedom, and the lower bound for Edit-Distance does not immediately imply any hardness for LCS."* Our reduction violates this intuition by showing that edit distance without substitutions can be used to efficiently simulate edit distance without substitutions.

Approximating Edit Distance Over an Arbitrary Metric (Section 6)

The aforementioned results for approximating edit distance [23, 9, 10, 33, 6, 5, 16, 32] consider only the case in which insertion, deletion, and substitution costs are all constant. To the best of our knowledge, no approximation algorithm is known for the more general case in which characters are taken from an arbitrary metric space and edit costs are assigned based on metric distances between characters. This variant of edit distance is sometimes referred to as *general edit distance* [36]. The study of general edit distance dates back to the first papers on edit distance [42, 40], and allowing for nonuniform costs is important in many applications, including in computational biology [26].

We present an approximation algorithm for edit distance over an an arbitrary metric. Our algorithm runs in time $\tilde{O}(n^{2-\epsilon})$ and computes an $O(n^{\epsilon})$-approximation for $\mathrm{ed}(x, y)$ with high probability. Note that for the case where characters are taken from a well-separated tree metric with logarithmic depth and at most exponential aspect ratio, the result already

follows from our approximation algorithm for DTW, and our reduction from edit distance to DTW. The approach taken in Section 6 is particularly interesting in that it places no restrictions on the underlying metric space.

Both our approximation algorithm for DTW and our approximation algorithm for edit distance exhibit relatively weak runtime/approximation tradeoffs. To the best of our knowledge, however, they are the first such algorithms to run in strongly subquadratic time.

2 Preliminaries

In this section, we present preliminary definitions and background on dynamic time warping distance (DTW) and edit distance.

Dynamic Time Warping Distance

For a metric space Σ, the dynamic time warping distance (DTW) between two strings $x, y \in \Sigma^n$ is a natural measure of similarity between the strings.

Before fully defining DTW, we first introduce the notion of an *expansion* of a string.

▶ **Definition 2.1.** *The* runs *of a string $x \in \Sigma^n$ are the maximal subsequences of consecutive letters with the same value. One can* extend *a run by replacing it with a longer run of the same letter. An* expansion *of the string x is any string which can be obtained from x by extending runs.*

As an example, consider $x = aaaccbbd$. Then the runs of x are aaa, cc, bb, and d. The string $\overline{x} = aaacccccbbdd$ is an expansion of x and extends the runs containing c and d.

Using the terminology of expansions, we now define DTW.

▶ **Definition 2.2.** *Consider strings x and y of length n over a metric (Σ, d). A* correspondence *$(\overline{x}, \overline{y})$ between x and y is a pair of equal-length expansions $\overline{x}$ of x and $\overline{y}$ of y. The* cost *of a correspondence is given by $\sum_i d(\overline{x}_i, \overline{y}_i)$.*

The dynamic time warping distance $\mathrm{dtw}(x, y)$ *is defined to be the minimum cost of a correspondence between x and y.*

When referring to a run r in one of x or y, and when talking about a correspondence $(\overline{x}, \overline{y})$, we will often use r to implicitly refer to the extended run corresponding with r in the correspondence. Whether we are referring to the original run or the extended version of the run should be clear from context.

Note that any minimum-length optimal correspondence between strings $x, y \in \Sigma^n$ will be of length at most $2n$. This is because if a run r_1 in x overlaps a run r_2 in y in the correspondence, then we may assume without loss of generality that at most one of the two runs is extended by the correspondence. (Otherwise, we could un-extend each run by one and arrive at a shorter correspondence with no added cost.)

Edit Distance Over an Arbitrary Metric

The *simple edit distance* between two strings x and y is the minimum number of insertions, deletions, and substitutions needed to transform x into y. In this paper we will mostly focus on a more general variant of edit distance, in which characters are taken from an arbitrary metric:

▶ **Definition 2.3.** *Let x and y be strings over an alphabet Σ, where $(\Sigma \cup \{\emptyset\}, d)$ is a metric space. We say that the* magnitude *$|l|$ of a letter $l \in \Sigma$ is $d(\emptyset, l)$. We define the* edit distance *between x and y to be the minimum cost of a sequence of edits from x to y, where the insertion or deletion of a letter l costs $d(\emptyset, l)$, and the substitution of a letter l to a letter l' costs $d(l, l')$.*

3 Computing DTW in the Low-Distance Regime

In this section, we present a low-distance regime algorithm for DTW (with characters from an arbitrary metric in which all non-zero distances are at least one). Given that $\mathrm{dtw}(x, y)$ is bounded above by a parameter K, our algorithm can compute $\mathrm{dtw}(x, y)$ in time $O(nK)$. Moreover, if $\mathrm{dtw}(x, y) > K$, then the algorithm will conclude as much. Consequently, by doubling our guess for K repeatedly, one can compute $\mathrm{dtw}(x, y)$ in time $O(n \cdot \mathrm{dtw}(x, y))$.

Consider x and y of length n with characters taken from a metric space Σ in which all non-zero distances are at least one. In the textbook dynamic program for DTW [30], each pair of indices $i, j \in [n]$ represents a subproblem $T(i, j)$ whose value is $\mathrm{dtw}(x[1 : i], y[1 : j])$. Since $T(i, j)$ can be determined using $T(i - 1, j), T(i, j - 1), T(i - 1, j - 1)$, and knowledge of x_i and y_j, this leads to an $O(n^2)$ algorithm for DTW. A common heuristic in practice is to construct only a small band around the main diagonal of the dynamic programming grid; by computing only entries $T(i, j)$ with $|i - j| \leq 2K$, and treating other subproblems as having infinite return values, one can obtain a correct computation for DTW as long as there is an optimal correspondence which matches only letters which are within K of each other in position. This heuristic is known as the Sakoe-Chiba Band [39] and is employed, for example, in the commonly used library of Giorgino [24].

The Sakoe-Chiba Band heuristic can perform badly even when $\mathrm{dtw}(x, y)$ is very small, however. Consider $x = abbb \cdots b$ and $y = aaa \cdots ab$. Although $\mathrm{dtw}(x, y) = 0$, if we restrict ourselves to matching letters within K positions of each other for some small K, then the resulting correspondence will cost $\Omega(n)$.

In order to obtain an algorithm which performs well in the low-distance regime, we introduce a new dynamic program for DTW. The new dynamic program treats x and y asymmetrically within each subproblem. Loosely speaking, for indices i and j, there are two subproblems $\mathrm{SP}(x, y, i, j)$ and $\mathrm{SP}(y, x, i, j)$. The first of these subproblems evaluates to the DTW between the first i runs of x and the first j letters of y, with the added condition that the final run of $y[1 : j]$ is not extended. The second of the subproblems is analogously defined as the DTW between the first i runs of y and the first j letters of x with the added condition that the final run of $x[1 : j]$ is not extended.

The recursion connecting the new subproblems is somewhat more intricate than for the textbook dynamic program. By matching the i-th run with the j-th letter, however, we limit the number of subproblems which can evaluate to less than K. In particular, if the j-th letter of y is in y's t-th run, then any correspondence which matches the i-th run of x to the j-th letter of y must cost at least $\Omega(|i - t|)$. (This is formally shown in the extended paper [31].) Thus for a given j, there are only $O(K)$ options for i such that $\mathrm{SP}(x, y, i, j)$ can possibly be at most K, and similarly for $\mathrm{SP}(y, x, i, j)$. Since we are interested in the case of $\mathrm{dtw}(x, y) \leq K$, we can restrict ourselves to the $O(nK)$ subproblems which have the potential to evaluate to at most $O(K)$. Notice that, in fact, our algorithm will work even when $\mathrm{dtw}(x, y) > K$ as long as there is an optimal correspondence between x and y which only matches letters from x from the r_x-th run with letters from y from the r_y-th run if $|r_x - r_y| \leq O(K)$.

Formally we define our recursive problems in a manner slightly different from that described above. Let x and y be strings of length at most n and let K be a parameter which we assume is greater than $\text{dtw}(x, y)$. Our subproblems will be the form $\text{SP}(x, y, r_x, r_y, o_y)$, which is defined as follows. Let x' consist of the first r_x runs of x and y' consist of the first r_y runs of y until the o_y-th letter in the r_y-th run. Then $\text{SP}(x, y, r_x, r_y, o_y)$ is the value of the optimal correspondence between x' and y' such that the r_y-th run in y' is not extended.[3] If no such correspondence exists (which can only happen if $r_y \leq 1$ or $r_x = 0$), then the value of the subproblem is ∞. Note that we allow r_x, r_y, o_y to be zero, and if r_y is zero, then o_y must be zero as well. We also consider the symmetrically defined subproblems of the form $\text{SP}(y, x, r_y, r_x, o_x)$. We will focus on the subproblems of the first types, implicitly treating subproblems of the second type symmetrically.

▶ Example 3.1. Suppose characters are taken from generalized Hamming space, with distances of 0 and 1. The subproblem $\text{SP}(efabbccccd, ffaabcccddd, 5, 4, 2)$ takes the value of the optimal correspondence between $efabbcccc$ and $ffaabcc$ such that the final cc run in the latter is not extended. The subproblem's value turns out to be 3, due to the correspondence:

e f a a b b c c c c
f f a a b b b b c c .

The next lemma presents the key recursive relationship between subproblems. The lemma focuses on the case where $r_x, r_y, o_y \geq 1$.

▶ **Lemma 3.2.** *Suppose that r_x and r_y are both between 1 and the number of runs in x and y respectively; and that o_y is between 1 and the length of the r_y-th run in y. Let l_x be the length of the r_x-th run in x and l_y be the length of the r_y-th run in y. Let d be the distance between the letter populating the r_x-th run in x and the letter populating the r_y-th run in y. Then $\text{SP}(x, y, r_x, r_y, o_y)$ is given by*

$$\begin{cases} \min\left(\text{SP}(x, y, r_x, r_y, o_y - 1) + d, \text{SP}(x, y, r_x - 1, r_y, o_y - l_x) + d \cdot l_x\right) & \text{if } l_x \leq o_y \\ \min\left(\text{SP}(x, y, r_x, r_y, o_y - 1) + d, \text{SP}(y, x, r_y - 1, r_x, l_x - o_y) + d \cdot o_y\right) & \text{if } l_x > o_y. \end{cases}$$

Proof. Consider a minimum-cost correspondence A between the first r_x runs of x and the portion of y up until the o_y-th letter in the r_y-th run, such that the r_y-th run in y is not extended.

If the r_x-th run in A is extended, then the cost of A will be $\text{SP}(x, y, r_x, r_y, o_y - 1) + d$. If the r_x-th run in A is not extended, then we consider two cases.

In the first case, $l_x \leq o_y$. In this case, the entirety of the r_x-th run of x is engulfed by the r_y-th run of y in the correspondence A. Since the r_x-th run is not extended, the cost of the overlap is $l_x \cdot d$. Thus the cost of A must be $\text{SP}(x, y, r_x - 1, r_y, o_y - l_x) + d \cdot l_x$.

Moreover, since A is minimum-cost, as long as $l_x \leq o_y$, the cost of A is at most the above expression, regardless of whether the r_x-th run in x is extended in A.

In the second case, $l_x > o_y$. In this case, the first o_y letters in the r_y-th run of y all overlap the r_x-th run of x in A. Since the r_y-th run is not extended, the cost of the overlap is $d \cdot o_y$. Thus, since the r_x-th run in x is also not extended in A, the cost of A must be $\text{SP}(y, x, r_y - 1, r_x, l_x - o_y) + d \cdot o_y$. Moreover, since A is minimal, as long as $l_x > o_y$, the cost of A is at most the above expression, regardless of whether the r_x-th run in x is extended. ◀

[3] If $o_y = 0$, then the r_y-th run in y' is empty and thus trivially cannot be extended.

The above lemma handles cases where $r_x, r_y, o_y > 0$. In the case where $r_x > 0$, $r_y > 0$, and $o_y = 0$, $\mathrm{SP}(x, y, r_x, r_y, o_y)$ is just the dynamic time warping distance between the first r_x runs of x and the first $r_y - 1$ runs of y, given by $\min\left(\mathrm{SP}(x, y, r_x, r_y - 1, t_1), \mathrm{SP}(y, x, r_y - 1, r_x, t_2)\right)$, where t_1 is the length of the $(r_y - 1)$-th run in y and t_2 is the length of the r_x-th run in x. The remaining cases are edge-cases with $\mathrm{SP}(x, y, r_x, r_y, o_y) \in \{0, \infty\}$. (See the extended paper [31].)

One can show that any correspondence A in which a letter from the r_x-th run of x is matched with a letter from the r_y-th run of y must contain must contain at least $\frac{|r_x - r_y| - 1}{2}$ instances of unequal letters being matched. It follows that if $\mathrm{dtw}(x, y) \leq K$, then we can limit ourselves to subproblems in which $|r_x - r_y| \leq O(K)$. For each of the n options of (r_y, o_y), there are only $O(K)$ options of r_x that must be considered. This limits the total number of subproblems to $O(nK)$. The resulting dynamic program yields the following theorem, the full proof of which appears in the extended paper [31].

▶ **Theorem 3.3.** *Let x and y be strings of length n taken from a metric space Σ with minimum non-zero distance at least one, and let K be parameter such that $\mathrm{dtw}(x, y) \leq K$. Then there exists a dynamic program for computing $\mathrm{dtw}(x, y)$ in time $O(nK)$. Moreover, if $\mathrm{dtw}(x, y) > K$, then the dynamic program will return a value greater than K.*

By repeatedly doubling one's guess for K until the computed value of $\mathrm{dtw}(x, y)$ evaluates to less than K, one can therefore compute $\mathrm{dtw}(x, y)$ in time $O(n \cdot \mathrm{dtw}(x, y))$.

4 Approximating DTW Over Well-Separated Tree Metrics

In this section, we present an $\tilde{O}(n^{2-\epsilon})$-time $O(n^{\epsilon})$-approximation algorithm for DTW over a well-separated tree metric with logarithmic depth. We begin by presenting a brief background on well-separated tree metrics.

▶ **Definition 4.1.** *Consider a tree T whose vertices form an alphabet Σ, and whose edges have positive weights. T is said to be a* well-separated tree metric *if every root-to-leaf path consists of edges ordered by nonincreasing weight. The* distance *between two nodes $u, v \in \Sigma$ is defined as the maximum weight of any edge in the shortest path from u to v.*

Well-separated tree metrics are universal in the sense that any metric Σ can be efficiently embedded (in time $O(|\Sigma|^2)$) into a well-separated tree metric T with expected distortion $O(\log |\Sigma|)$ [22]. Moreover, the tree metric may be made to have logarithmic depth using Theorem 8 of [8]. For strings $x, y \in \Sigma^n$, let $\mathrm{dtw}_T(x, y)$ denote the dynamic time warping distance after embedding Σ into T. Then the tree-metric embedding guarantees that $\mathrm{dtw}(x, y) \leq \mathrm{dtw}_T(x, y)$ and that $\mathbb{E}[\mathrm{dtw}_T(x, y)] \leq O(\log n) \cdot \mathrm{dtw}(x, y)$. (The latter fact may be slightly nonobvious and is further explained in the extended paper [31].)

It follows that any approximation algorithm for DTW over well-separated tree metrics will immediately yield an approximation algorithm over an arbitrary polynomial-size metric Σ, with two caveats: the new algorithm will have its multiplicative error increased by $O(\log n)$; and $O(\log n)$ instances of Σ embedded into a well-separated tree metric must be precomputed for use by the algorithm (requiring, in general, $O(|\Sigma|^2 \log n)$ preprocessing time). In particular, given $O(\log n)$ tree embeddings of Σ, $T_1, \ldots, T_{O(\log n)}$, with high probability $\min_i \left(\mathrm{dtw}_{T_i}(x, y)\right)$ will be within a logarithmic factor of $\mathrm{dtw}(x, y)$.

The remainder of the section will be devoted to designing an approximation algorithm for DTW over a well-separated tree metric. We will prove the following theorem:

▶ **Theorem 4.2.** *Consider $0 < \epsilon < 1$. Suppose that Σ is a well-separated tree metric of polynomial size and at most logarithmic depth. Moreover, suppose that the aspect ratio of Σ is at most exponential in n (i.e., the ratio between the largest distance and the smallest non-zero distance). Then in time $\tilde{O}(n^{2-\epsilon})$ we can obtain an $O(n^\epsilon)$-approximation for $\mathrm{dtw}(x, y)$ for any $x, y \in \Sigma^n$.*

An important consequence of the theorem occurs for DTW over the reals. When Σ is an $O(n)$-point subset of the reals with a polynomial aspect ratio, there exists an $O(n \log n)$-time embedding with $O(\log n)$ expected distortion from Σ to a well-separated tree metric of size $O(n)$ with logarithmic depth. (This is further discussed in the extended paper [31].). This gives the following corollary:

▶ **Corollary 4.3.** *Consider $0 < \epsilon < 1$. Suppose that $\Sigma = [0, n^c] \cap \mathbb{Z}$ for some constant c. Then in time $\tilde{O}(n^{2-\epsilon})$ we can obtain an $O(n^\epsilon)$-approximation for $\mathrm{dtw}(x, y)$ with high probability for any $x, y \in \Sigma^n$.*

In proving Theorem 4.2, our approximation algorithm will take advantage of what we refer to as the *r-simplification* of a string over a well-separated tree metric.

▶ **Definition 4.4.** *Let T be a well-separated tree metric whose nodes form an alphabet Σ. For a string $x \in \Sigma^n$, and for any $r \geq 1$, the* r-simplification *$s_r(x)$ is constructed by replacing each letter $l \in x$ with its highest ancestor l' in T that can be reached from l using only edges of weight at most $r/4$.*

Our approximation algorithm will apply the low-distance regime algorithm from the previous section to $s_r(x)$ and $s_r(y)$ for various r in order to extract information about $\mathrm{dtw}(x, y)$. Notice that using our low-distance regime algorithm for DTW, we get the following useful lemma for free:

▶ **Lemma 4.5.** *Consider $0 < \epsilon < 1$. Suppose that for all pairs l_1, l_2 of distinct letters in Σ, $d(l_1, l_2) \geq \gamma$. Then for $x, y \in \Sigma^n$ there is an $O(n^{2-\epsilon})$ time algorithm which either computes $\mathrm{dtw}(x, y)$ exactly, or concludes that $\mathrm{dtw}(x, y) > \gamma n^{1-\epsilon}$.*

The next lemma states three important properties of r-simplifications. We remark that the same lemma appears in our concurrent work on the communication complexity of DTW, in which we use the lemma in designing an efficient one-way communication protocol [12].

▶ **Lemma 4.6.** *Let T be a well-separated tree metric with distance function d and whose nodes form the alphabet Σ. Consider strings x and y in Σ^n.*

Then the following three properties of $s_r(x)$ and $s_r(y)$ hold:

- *For every letter $l_1 \in s_r(x)$ and every letter $l_2 \in s_r(y)$, if $l_1 \neq l_2$, then $d(l_1, l_2) > r/4$.*
- *For all α, if $\mathrm{dtw}(x, y) \leq nr/\alpha$ then $\mathrm{dtw}(s_r(x), s_r(y)) \leq nr/\alpha$.*
- *If $\mathrm{dtw}(x, y) > nr$, then $\mathrm{dtw}(s_r(x), s_r(y)) > nr/2$.*

The first and second parts of Lemma 4.6 are straightforward from the definitions of $s_r(x)$ and $s_r(y)$. The third part follows from the observation that a correspondence C between x and y can cost at most $|C| \cdot \frac{r}{4}$ more than the corresponding correspondence between $s_r(x)$ and $s_r(y)$, where $|C|$ denotes the length of the correspondence. Since there exists an optimal correspondence between $s_r(x)$ and $s_r(y)$ of length no more than $2n$, it follows that $\mathrm{dtw}(x, y) \leq \mathrm{dtw}(s_r(x), s_r(y)) + nr/2$, which implies the third part of the lemma.

A full proof of Lemma 4.6 appears in the extended paper [31]. Next we prove Theorem 4.2.

Proof of Theorem 4.2. Without loss of generality, the minimum non-zero distance in Σ is 1 and the largest distance is some value m, which is at most exponential in n.

We begin by defining the (r, n^ϵ)-*DTW gap* problem for $r \geq 1$, in which for two strings x and y a return value of 0 indicates that $\mathrm{dtw}(x, y) < nr$ and a return value of 1 indicates that $\mathrm{dtw}(x, y) \geq n^{1-\epsilon}r$. By Lemma 4.6, in order to solve the (r, n^ϵ)-DTW gap problem for x and y, it suffices to determine whether $\mathrm{dtw}(s_r(x), s_r(y)) \leq n^{1-\epsilon}r$. Moreover, because the minimum distance between distinct letters in $s_r(x)$ and $s_r(y)$ is at least $r/4$, this can be done in time $O(n^{2-\epsilon} \log n)$ using Lemma 4.5.[4]

In order to obtain an n^ϵ-approximation for $\mathrm{dtw}(x, y)$, we begin by using Lemma 4.5 to either determine $\mathrm{dtw}(x, y)$ or to determine that $\mathrm{dtw}(x, y) \geq n^{1-\epsilon}$. For the rest of the proof, suppose we are in the latter case, meaning that we know $\mathrm{dtw}(x, y) \geq n^{1-\epsilon}$.

We will now consider the $(2^i, n^\epsilon/2)$-DTW gap problem for $i \in \{0, 1, 2, \ldots, \lceil \log m \rceil\}$. (Recall that m is the largest distance in Σ.) If the $(2^0, n^\epsilon/2)$-DTW gap problem returned 0, then we would know that $\mathrm{dtw}(x, y) \leq n$, and thus we could return $n^{1-\epsilon}$ as an n^ϵ-approximation for $\mathrm{dtw}(x, y)$. Therefore, we need only consider the case where the $(2^0, n^\epsilon/2)$-DTW gap returns 1. Moreover we may assume without computing it that $(2^{\lceil \log m \rceil}, n^\epsilon/2)$-DTW gap returns 0 since trivially $\mathrm{dtw}(x, y)$ cannot exceed nm. Because $(2^i, n^\epsilon/2)$-DTW gap returns 1 for $i = 0$ and returns 0 for $i = \lceil \log m \rceil$, there must be some i such that $(2^{i-1}, n^\epsilon/2)$-DTW gap returns 1 and $(2^i, n^\epsilon/2)$-DTW gap returns 0. Moreover, we can find such an i by performing a binary search on i in the range $R = \{0, \ldots, \lceil \log m \rceil\}$. We begin by computing $(2^i, n^\epsilon/2)$-DTW gap for i in the middle of the range R. If the result is a one, then we can recurse on the second half of the range; otherwise we recurse on the first half of the range. Continuing like this, we can find in time $\tilde{O}(n^{2-\epsilon} \log \log m) = \tilde{O}(n^{2-\epsilon})$ some value i for which $(2^{i-1}, n^\epsilon/2)$-DTW gap returns 1 and $(2^i, n^\epsilon/2)$-DTW gap returns 0. Given such an i, we know that $\mathrm{dtw}(x, y) \geq \frac{2^{i-1}n}{n^\epsilon/2} = 2^i n^{1-\epsilon}$ and that $\mathrm{dtw}(x, y) \leq 2^i n$. Thus we can return $2^i n^{1-\epsilon}$ as an n^ϵ approximation of $\mathrm{dtw}(x, y)$. ◀

5 Reducing Edit Distance to DTW and LCS

In this section we present a simple reduction from edit distance over an arbitrary metric to DTW over the same metric.At the end of the section, we prove as a corollary a conditional lower bound for DTW over three-letter Hamming space, prohibiting any algorithm from running in strongly subquadratic time.

Surprisingly, the *exact same reduction*, although with a different analysis, can be used to reduce the computation of edit distance (over generalized Hamming space) to the computation of longest-common-subsequence length (LCS). Since computing LCS is equivalent to computing edit distance without substitutions, this reduction can be interpreted as proving that edit distance without substitutions can be used to efficiently simulate edit distance with substitutions, also known as *simple edit distance*.

Recall that for a metric $\Sigma \cup \{\emptyset\}$, we define the edit distance between two strings $x, y \in \Sigma^n$ such that the cost of a substitution from a letter l_1 to l_2 is $d(l_1, l_2)$, and the cost of a deletion or insertion of a letter l is $d(l, \emptyset)$. Additionally, define the *simple edit distance* $\mathrm{ed}_S(x, y)$ to be the edit distance using only insertions and deletions.

For a string $x \in \Sigma^n$, define the *padded string* $p(x)$ of length $2n + 1$ to be the string $\emptyset x_1 \emptyset x_2 \emptyset x_3 \cdots x_n \emptyset$. In particular, for $i \leq 2n + 1$, $p(x)_i = \emptyset$ when i is odd, and $p(x)_i = x_{i/2}$ when i is even. The following theorem proves that $\mathrm{dtw}(p(x), p(y)) = \mathrm{ed}(x, y)$.

[4] The logarithmic factor comes from the fact that evaluating distances between points may take logarithmic time in our well-separated tree metric.

▶ **Theorem 5.1.** *Let $\Sigma \cup \{\emptyset\}$ be a metric. Then for any $x, y \in \Sigma^n$, $\mathrm{dtw}(p(x), p(y)) = \mathrm{ed}(x, y)$.*

Proof sketch. A key observation is that when constructing an optimal correspondence between $p(x)$ and $p(y)$, one may w.l.o.g. extend only runs consisting of $\emptyset$ characters. In particular, suppose that one extends a non-$\emptyset$ character a in $p(x)$ to match a non-$\emptyset$ character b in $p(y)$. Then the extended run of a's must not only overlap b, but also the $\emptyset$-character preceding b. The total cost of extending a to overlap b is therefore $d(a, \emptyset) + d(a, b)$, which by the triangle inequality is at least $d(\emptyset, b)$. Thus instead of extending the run containing a, one could have instead extend a run of $\emptyset$-characters to overlap b at the same cost.

The fact that optimal correspondences arise by simply extending runs of $\emptyset$-characters can then be used to prove Theorem 5.1; in particular, given such a correspondence, one can obtain a sequence of edits from x to y by performing a substitution every time the correspondence matches two non-$\emptyset$ characters and a insertion or deletion every time the correspondence matches a non-$\emptyset$ character and a $\emptyset$-character. ◀

Theorem 5.2 proves an analogous reduction from edit distance to LCS.

▶ **Theorem 5.2.** *Let Σ be a generalized Hamming metric. Then for any $x, y \in \Sigma^n$, $\mathrm{ed}_S(p(x), p(y)) = 2\,\mathrm{ed}(x, y)$.*

Proof sketch. Each edit in x can be simulated in $s(x)$ using exactly two insertions/deletions. In particular, the substitution of a character in x corresponds with the deletion and insertion of the same character in $s(x)$; and the insertion/deletion of a character in x corresponds with the insertion/deletion of that character and an additional $\emptyset$-character in $s(x)$.

This establishes that $\mathrm{ed}_S(p(x), p(y)) \leq 2\,\mathrm{ed}(x, y)$. The other direction of inequality is somewhat more subtle, and is differed to the extended paper [31]. ◀

Whereas Theorem 5.2 embeds edit distance into simple edit distance with no distortion, Theorem 5.3 shows that no nontrivial embedding in the other direction exists.

▶ **Theorem 5.3.** *Consider edit distance over generalized Hamming space. Any embedding from edit distance to simple edit distance must have distortion at least 2.*

Combining Theorem 5.1 with known conditional lower bounds for computing edit distance [13] yields a new conditional lower bound for computing DTW over a three-letter alphabet (in which character distances are zero or one). This concludes a direction of work initiated by Abboud, Backurs, and Williams [3], who proved the same result over five-letter alphabet.

▶ **Corollary 5.4.** *Let $\Sigma = \{a, b, c\}$ with distance function $d(a, b) = d(a, c) = d(b, c) = 1$. If we assume the Strong Exponential Time Hypothesis, then for all $\epsilon > 1$, no algorithm can compute $\mathrm{dtw}(x, y)$ for $x, y \in \Sigma^n$ in time less than $O(n^{2-\epsilon})$.*

The full proofs of Theorems 5.1, 5.2, and 5.3, as well as Corollary 5.4 are differed to the extended paper [31].

6 Approximating Edit Distance Over an Arbitrary Metric

In this section we present an approximation algorithm for edit distance over an arbitrary metric space. Our algorithm achieves approximation ratio at most n^ϵ (with high probability) and runtime $\tilde{O}(n^{2-\epsilon})$. Note that when the metric is a well-separated tree metric, such an algorithm can be obtained by combining the approximation algorithm for DTW from Section

4 with the reduction in Section 5. Indeed the algorithm in this section is structurally quite similar to the one in Section 4, but uses a probability argument exploiting properties of edit distance in order to hold over an arbitrary metric.

▶ **Theorem 6.1.** *Let $(\Sigma \cup \{\emptyset\}, d)$ be an arbitrary metric space such that $|l| \geq 1$ for all $l \in \Sigma$. For all $0 < \epsilon < 1$, and for strings $x, y \in \Sigma^n$, there is an algorithm which computes an $O(n^\epsilon)$-approximation for $\mathrm{ed}(x, y)$ (with high probability) in time $\tilde{O}(n^{2-\epsilon})$.*

Using the standard dynamic-programming algorithm for computing $\mathrm{ed}(x, y)$ [42, 37, 41], one can easily obtain the following observation, analogous to Lemma 4.5 in Section 4:

▶ Observation 6.2. Consider $x, y \in \Sigma^n$, and let R be the smallest magnitude of the letters in x and y. There is an $O(n^{2-\epsilon})$-time algorithm which returns a value at least as large as $\mathrm{ed}(x, y)$; and which returns exactly $\mathrm{ed}(x, y)$ when $\mathrm{ed}(x, y) \leq R \cdot n^{1-\epsilon}$.

In order to prove Theorem 6.1, we present a new definition of the r-simplification of a string. The difference between this definition and the one in the preceding section allows the new definition to be useful when studying edit distance rather than dynamic time warping.

▶ **Definition 6.3.** *For a string $x \in \Sigma^n$ and for $r \geq 1$, we construct the* r-simplification $s_r(x)$ *by removing any letter l satisfying $|l| \leq r$.*

In the proof of Theorem 6.1 we use randomization in the selection of r in order to ensure that $s_r(x)$ satisfies desirable properties in expectation. The key proposition follows:

▶ **Proposition 6.4.** *Consider strings x and y in Σ^n. Consider $0 < \epsilon < 1$ and $R \geq 1$. Select r to be a random real between R and $2R$. Then the following three properties hold:*

- *Every letter l in $s_r(x)$ or $s_r(y)$ satisfies $|l| \geq R$.*
- *If $\mathrm{ed}(x, y) \leq \frac{nR}{15n^\epsilon}$ then $\mathbb{E}[\mathrm{ed}(s_r(x), s_r(y))] \leq \frac{nR}{3n^\epsilon}$.*
- *If $\mathrm{ed}(x, y) > 5nR$, then $\mathrm{ed}(s_r(x), s_r(y)) > nR$.*

The full proofs of Proposition 6.4 and of Theorem 6.1 appear in the extended paper [31]. Structurally, both proofs are similar to the analogous results in Section 4. The key difference appears in the proof of the second part of Proposition 6.4, which uses the random selection of r in order to probabilistically upper-bound $\mathrm{ed}(s_r(x), s_r(y)$. This is presented below.

▶ **Lemma 6.5.** *Consider strings x and y in Σ^n. Consider $R \geq 1$ and select r to be a random real between R and $2R$. Then $\mathbb{E}[\mathrm{ed}(s_r(x), s_r(y))] \leq 5\,\mathrm{ed}(x, y)$.*

Proof. Consider an optimal sequence S of edits from x to y. We will consider the cost of simulating this sequence of edits to transform $s_r(x)$ to $s_r(y)$. Insertions and deletions are easily simulated by either performing the same operation to $s_r(x)$ or performing no operation at all (if the operation involves a letter of magnitude less than or equal to r). Substitutions are slightly more complicated as they may originally be between letters $l_1 \in x$ and $l_2 \in y$ of different magnitudes. By symmetry, we may assume without loss of generality that $|l_1| < |l_2|$. We will show that the expected cost of simulating the substitution of l_1 to l_2 in $s_r(x)$ is at most $5d(l_1, l_2)$. Because insertions and deletions can be simulated with no overhead, it follows that $\mathbb{E}[\mathrm{ed}(s_r(x), s_r(y))] \leq 5\,\mathrm{ed}(x, y)$.

If $|l_1| \leq r < |l_2|$ then l_1 does not appear in $s_r(x)$ but l_2 remains in $s_r(y)$. Thus what was previously a substitution of l_1 with l_2 becomes an insertion of l_2 at cost $|l_2|$. On the other hand, if we do not have $|l_1| \leq r < |l_2|$, then either both l_1 and l_2 are removed from $s_r(x)$ and $s_r(y)$ respectively, in which the substitution operation no longer needs to be performed, or both l_1 and l_2 are still present, in which case the substitution operation can still be performed at cost $d(l_1, l_2)$. Therefore, the expected cost of simulating the substitution of l_1 to l_2 in $s_r(x)$ is at most

$$\Pr[|l_1| \leq r < |l_2|] \cdot |l_2| + d(l_1, l_2). \tag{1}$$

Because r is selected at random from the range $[R, 2R]$, the probability that $|l_1| \leq r < |l_2|$ is at most $\frac{|l_2|-|l_1|}{R}$. By the triangle inequality, this is at most $\frac{d(l_1,l_2)}{R}$. If we suppose that $|l_2| \leq 4R$, then it follows by (1) that the expected cost of simulating the substitution of l_1 to l_2 in $s_r(x)$ is at most $\frac{d(l_1,l_2)}{R} \cdot 4R + d(l_1, l_2) \leq 5d(l_1, l_2)$.

If, on the other hand, $|l_2| > 4R$, then in order for $|l_1| \leq r$ to be true, we must have $|l_1| \leq 2R$, meaning by the triangle inequality that $d(l_1, l_2) \geq |l_2|/2$. Thus in this case $|l_2| \leq 2d(l_1, l_2)$, meaning by (1) that the expected cost of simulating the substitution of l_1 to l_2 in $s_r(x)$ is at most three times as expensive as the original substitution. ◀

References

1. John Aach and George M Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17(6):495–508, 2001.
2. Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Quadratic-time hardness of LCS and other sequence similarity measures. *arXiv preprint*, 2015. `arXiv:1501.07053`.
3. Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 59–78, 2015.
4. Pankaj K. Agarwal, Kyle Fox, Jiangwei Pan, and Rex Ying. Approximating Dynamic Time Warping and Edit Distance for a Pair of Point Sequences. In *32nd International Symposium on Computational Geometry (SoCG)*, pages 6:1–6:16, 2016.
5. Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. In *Proceedings of the 51st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 377–386, 2010.
6. Alexandr Andoni and Krzysztof Onak. Approximating Edit Distance in Near-Linear Time. *SIAM J. Comput.*, 41(6):1635–1648, 2012.
7. Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of the 47th Annual Symposium on Theory of Computing (STOC)*, pages 51–58, 2015.
8. Nikhil Bansal, Niv Buchbinder, Aleksander Madry, and Joseph Naor. A polylogarithmic-competitive algorithm for the k-server problem. In *52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 267–276, 2011.
9. Ziv Bar-Yossef, T.S. Jayram, Robert Krauthgamer, and Ravi Kumar. Approximating edit distance efficiently. In *Proceedings of 45th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 550–559, 2004.
10. Tugkan Batu, Funda Ergün, and Süleyman Cenk Sahinalp. Oblivious string embeddings and edit distance approximations. In *Proceedings of the 17th Annual Symposium on Discrete Algorithms (SODA)*, pages 792–801, 2006.
11. Nurjahan Begum, Liudmila Ulanova, Jun Wang, and Eamonn J. Keogh. Accelerating Dynamic Time Warping Clustering with a Novel Admissible Pruning Strategy. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 49–58, 2015.
12. Vladimir Braverman, Moses Charikar, William Kuszmaul, David Woodruff, and Lin Yang. The One-Way Communication Complexity of Dynamic Time Warping Distance. Manuscript submitted for publication, 2018.
13. Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 79–97, 2015.
14. Karl Bringmann and Wolfgang Mulzer. Approximability of the discrete Fréchet distance. *Journal of Computational Geometry*, 7(2):46–76, 2015.
15. EG Caiani, A Porta, G Baselli, M Turiel, S Muzzupappa, F Pieruzzi, C Crema, A Malliani, and S Cerutti. Warped-average template technique to track on a cycle-by-cycle basis the

cardiac filling phases on left ventricular volume. In *Computers in Cardiology 1998*, pages 73–76, 1998.

16 Diptarka Chakraborty, Debarati Das, Elazar Goldenberg, Michal Kouckỳ, and Michael Saks. Approximating Edit Distance Within Constant Factor in Truly Sub-Quadratic Time. In *Proceedings of the 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 979–990, 2018.

17 Diptarka Chakraborty, Elazar Goldenberg, and Michal Kouckỳ. Streaming algorithms for computing edit distance without exploiting suffix trees. *arXiv preprint*, 2016. arXiv:1607.03718.

18 Moses Charikar, Ofir Geri, Michael P. Kim, and William Kuszmaul. On Estimating Edit Distance: Alignment, Dimension Reduction, and Embeddings. In *45th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 34:1–34:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.34.

19 Alexander De Luca, Alina Hang, Frederik Brudy, Christian Lindner, and Heinrich Hussmann. Touch me once and i know it's you!: implicit authentication based on touch screen patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 987–996, 2012.

20 Anne Driemel and Francesco Silvestri. Locality-Sensitive Hashing of Curves. In *33rd International Symposium on Computational Geometry (SoCG)*, pages 37:1–37:16, 2017.

21 Ioannis Z. Emiris and Ioannis Psarros. Products of Euclidean Metrics and Applications to Proximity Questions among Curves. In *34th International Symposium on Computational Geometry, SoCG 2018, June 11-14, 2018, Budapest, Hungary*, pages 37:1–37:13, 2018.

22 Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004.

23 Zvi Galil and Kunsoo Park. An improved algorithm for approximate string matching. *SIAM Journal on Computing*, 19(6):989–999, 1990.

24 Toni Giorgino et al. Computing and visualizing dynamic time warping alignments in R: the DTW package. *Journal of statistical Software*, 31(7):1–24, 2009.

25 Omer Gold and Micha Sharir. Dynamic Time Warping and Geometric Edit Distance: Breaking the Quadratic Barrier. In *44th International Colloquium on Automata, Languages, and Programming, (ICALP)*, pages 25:1–25:14, 2017.

26 Tao Jiang, Guohui Lin, Bin Ma, and Kaizhong Zhang. A general edit distance between RNA structures. *Journal of computational biology*, 9(2):371–388, 2002.

27 Eamonn J. Keogh. Exact Indexing of Dynamic Time Warping. In *28th International Conference on Very Large Data Bases (VLDB)*, pages 406–417, 2002.

28 Eamonn J. Keogh and Michael J. Pazzani. Scaling up Dynamic Time Warping to Massive Dataset. In *Principles of Data Mining and Knowledge Discovery, Third European Conference, (PKDD)*, pages 1–11, 1999.

29 Eamonn J. Keogh and Michael J. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 285–289, 2000.

30 Jon Kleinberg and Eva Tardos. *Algorithm design*. Pearson, 2006.

31 William Kuszmaul. Dynamic Time Warping in Strongly Subquadratic time: Algorithms for the Low-Distance Regime and Approximate Evaluation. *arXiv preprint*, 2019. arXiv:1904.09690.

32 William Kuszmaul. Efficiently Approximating Edit Distance Between Pseudorandom Strings. In *Proceedings of the thirtieth annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2019.

33 Gad M. Landau, Eugene W. Myers, and Jeanette P. Schmidt. Incremental string comparison. *SIAM Journal on Computing*, 27(2):557–582, 1998.

34 Lindasalwa Muda, Mumtaj Begam, and Irraivan Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (MFCC) and dynamic time warping (DTW) techniques. *arXiv preprint*, 2010. `arXiv:1003.4083`.

35 Mario E Munich and Pietro Perona. Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In *Proceedings of 7th International Conference on Computer Vision*, volume 1, pages 108–115, 1999.

36 Gonzalo Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001.

37 Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

38 François Petitjean, Germain Forestier, Geoffrey I. Webb, Ann E. Nicholson, Yanping Chen, and Eamonn J. Keogh. Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm. *Knowl. Inf. Syst.*, 47(1):1–26, 2016.

39 Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

40 Esko Ukkonen. Algorithms for approximate string matching. *Information and control*, 64(1-3):100–118, 1985.

41 Taras K. Vintsyuk. Speech discrimination by dynamic programming. *Cybernetics*, 4(1):52–57, 1968.

42 Robert A. Wagner and Michael J. Fischer. The String-to-String Correction Problem. *J. ACM*, 21(1):168–173, 1974.

43 Rex Ying, Jiangwei Pan, Kyle Fox, and Pankaj K Agarwal. A simple efficient approximation algorithm for dynamic time warping. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 21, 2016.

44 Yunyue Zhu and Dennis Shasha. Warping indexes with envelope transforms for query by humming. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 181–192, 2003.

A Simple Gap-Producing Reduction for the Parameterized Set Cover Problem

Bingkai Lin
National Institute of Informatics, Tokyo, Japan
Nanjing University, Nanjing, China
lin@nii.ac.jp

Abstract

Given an n-vertex bipartite graph $I = (S, U, E)$, the goal of set cover problem is to find a minimum sized subset of S such that every vertex in U is adjacent to some vertex of this subset. It is NP-hard to approximate set cover to within a $(1 - o(1)) \ln n$ factor [14]. If we use the size of the optimum solution k as the parameter, then it can be solved in $n^{k+o(1)}$ time [16]. A natural question is: can we approximate set cover to within an $o(\ln n)$ factor in $n^{k-\epsilon}$ time?

In a recent breakthrough result[24], Karthik, Laekhanukit and Manurangsi showed that assuming the Strong Exponential Time Hypothesis (SETH), for any computable function f, no $f(k) \cdot n^{k-\epsilon}$-time algorithm can approximate set cover to a factor below $(\log n)^{\frac{1}{poly(k, e(\epsilon))}}$ for some function e.

This paper presents a simple gap-producing reduction which, given a set cover instance $I = (S, U, E)$ and two integers $k < h \leq (1 - o(1)) \sqrt[k]{\log |S| / \log \log |S|}$, outputs a new set cover instance $I' = (S, U', E')$ with $|U'| = |U|^{h^k} |S|^{O(1)}$ in $|U|^{h^k} \cdot |S|^{O(1)}$ time such that

- if I has a k-sized solution, then so does I';
- if I has no k-sized solution, then every solution of I' must contain at least h vertices.

Setting $h = (1 - o(1)) \sqrt[k]{\log |S| / \log \log |S|}$, we show that assuming SETH, for any computable function f, no $f(k) \cdot n^{k-\epsilon}$-time algorithm can distinguish between a set cover instance with k-sized solution and one whose minimum solution size is at least $(1 - o(1)) \cdot \sqrt[k]{\frac{\log n}{\log \log n}}$. This improves the result in [24].

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases set cover, FPT inapproximability, gap-producing reduction, (n, k)-universal set

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.81

Category Track A: Algorithms, Complexity and Games

Related Version https://arxiv.org/abs/1902.03702

Funding This work was supported in part by the National Key R&D Program of China 2018YFB1003202 and JSPS KAKENHI Grant Number JP18H05291.

Acknowledgements The author wishes to thank the anonymous referees for their detailed comments.

1 Introduction

We consider the *set cover* problem (SetCover): given an n-vertex bipartite graph $I = (S, U, E)$, where U is the underlying universe set and S represents the set family, find a minimum sized subset C of S such that every vertex of U is adjacent to some vertex of C. We use $S(I)$, $U(I)$ and $opt(I)$ to denote the sets S, U and the minimum size of the solution of I respectively. A vertex $u \in U$ is *covered* by a subset $C \subseteq S$ if u is adjacent to some vertex of C. The set cover problem is NP-hard [23]. Unless $P = NP$, we do not expect to solve it in polynomial time. One way to handle NP-hard problems is to use approximation algorithms. An algorithm of SetCover achieves an r-approximation if for every input instance I, it returns a subset C of $S(I)$ such that C covers $U(I)$ and $|C| \leq r \cdot opt(I)$. The polynomial time approximability of SetCover is well-understood: the greedy algorithm can output a solution of size at most $opt(I) \cdot (1 + \ln n)$ [10, 21, 28, 34, 35] and it was shown that

no polynomial time algorithm can achieve an approximation factor within $(1 - o(1)) \ln n$ unless $P = NP$ [4, 14, 17, 29, 32]. On the other hand, if we take the optimum solution size $k = opt(I)$ as a parameter, then the simple brute-force searching algorithm can solve this problem in n^{k+1} time. Assuming the exponential time hypothesis (ETH) [19, 20], i.e., 3-SAT on n variables cannot be solved in $2^{o(n)}$ time, there is no $n^{o(k)}$ time algorithm for SetCover. Under the strong exponential time hypothesis (SETH) [19, 20], which claims that for any $\epsilon \in (0, 1)$ there exists a $d \geq 3$ such that d-SAT on n variables cannot be solved in $2^{(1-\epsilon)n}$ time, we can further rule out $n^{k-\epsilon}$-time algorithm for set cover for any $\epsilon > 0$ [31]. It is quite natural to ask [11]:

Is there any $o(\ln n)$-approximation algorithm for the parameterized set cover problem (or dominating set problem) with running time $n^{k-\epsilon}$?

Exponential time approximation algorithms for the unparameterised version of set cover problem were studied in [7, 13]. It was shown that for any ratio r, there is a $(1 + \ln r)$-approximation algorithm for SetCover with running time $2^{n/r} n^{O(1)}$. No $n^{k-\epsilon}$ time algorithm for SetCover achieving an approximation ratio in $o(\ln n)$ is known in literature. On the other hand, proving inapproximability for a parameterized problem is not an easy task. In fact, even the constant FPT-approximability, i.e., the existence of $f(k) \cdot n^{O(1)}$-time algorithm for any computable function f (henceforth referred to as FPT-algorithm) with constant approximation, has been open for many years [30]. Lacking techniques like PCP-theorem [5], many results on the parameterized inapproximability of set cover problem had to use strong conjectures [6, 8] to create a gap in the first place. It is of great interest to develop techniques to prove hardness of approximation for parameterized problems only using hypothesis such as $SETH$, ETH or even weaker assumptions like W[1] $\neq FPT$ or W[2] $\neq FPT$ [15, 18] from the parameterized complexity theory. The success of this quest might extend the arsenal of methods for proving hardness of approximation and lead to PCP-like theorems for Fine-Grained Complexity [3].

The first constant FPT-inapproximability result for parameterized SetCover based on W[1] $\neq FPT$ was given by [9] using the one-sided gap of Biclique from [26]. In fact, [9] deals with dominating set problem, which is essentially the same as SetCover. Recently, Karthik, Laekhanukit and Manurangsi [24] significantly improved the FPT-inapproximation factor to $(\log n)^{1/k^{O(1)}}$ under the hypothesis W[1] $\neq FPT$. They also rule out the existence of $(\log n)^{1/k^{O(1)}}$-approximation algorithm with running time $f(k) \cdot n^{o(k)}$ for any computable function f, assuming ETH, and the existence of $(\log n)^{\frac{1}{(k+e(\epsilon))^{O(1)}}}$-approximation algorithms with running time $f(k) \cdot n^{k-\epsilon}$, assuming SETH. Their approach is to first establish a $(\log n)^{\frac{1}{\Omega(k)}}$ gap for MaxCover, then reduce MaxCover to SetCover and obtain a $(\log n)^{\frac{1}{\Omega(k^2)}}$-gap. This paper presents a new technique which allows us to design simple reductions improving the inapproximation factor to $(1 - \epsilon) \cdot \sqrt[k]{\frac{\log n}{\log \log n}}$. The reduction in [8] can get the ratio $(\log n)^{\Omega(1/k)}$ but it has to assume Gap-ETH.

▶ **Theorem 1.** *Assuming SETH, for every $\epsilon, \delta \in (0, 1)$, sufficiently large k[1] and computable function $f : \mathbb{N} \to \mathbb{N}$, there is no $f(k) \cdot N^{k-\epsilon}$ time algorithm that can, given an N-vertex set cover instance I, distinguish between*

- $opt(I) \leq k$,
- $opt(I) > \frac{1}{1+\delta} \left(\frac{\log N}{\log \log N} \right)^{\frac{1}{k}}$.

[1] We need large k to get the $\frac{1}{1+\delta} \left(\frac{\log N}{\log \log N} \right)^{\frac{1}{k}}$ gap for small δ. If we want to obtain an $\Theta\left(\sqrt[k]{\frac{\log N}{\log \log N}} \right)$ gap, then our reduction works for all $k \geq 2$.

▶ **Theorem 2.** *Assuming ETH, there is a constant $\epsilon \in (0,1)$ such that for every $\delta \in (0,1)$ and computable function $f : \mathbb{N} \to \mathbb{N}$, no $f(k) \cdot N^{\epsilon k}$ time algorithm that can, given an N-vertex set cover instance I , distinguish between*

- $opt(I) \leq k$,
- $opt(I) > \frac{1}{1+\delta} \cdot \left(\frac{\log N}{\log \log N}\right)^{\frac{1}{k}}$.

Behind these results is a reduction which, given an integer k, an n-vertex set cover instance I and an integer $h \leq O(\log n / \log \log n)$, produces an $n^{O(1)} \cdot (|U(I)|)^{O(h^k)}$-vertex instance I' in $n^{O(1)} \cdot |U(I)|^{O(h^k)}$ time such that if $opt(I) \leq k$ then $opt(I') \leq k$, otherwise $opt(I') > h$. Therefore, to prove the h-factor parameterized inapproximability of SetCover, it suffices to show the hardness of SetCover when the input instances have $n^{O(1/h^k)}$-size universe set. Note that the standard reduction for the SETH-hardness of set cover parameterized by the solution size k produces instances I with $|U(I)| = O(k \log |S(I)|)$. With our reduction, this immediately yields the above theorems. Let us not fail to mention that the results of [24] also imply the hardness of SetCover with logarithmic sized universe set assuming the k-SUM hypothesis and W[1] $\neq FPT$ hypothesis respectively. Similarly, we can obtain the corresponding inapproximability for set cover based on each of these hypotheses as well. In particular, using a simple trick, we can even rule out $(\log N)^{1/\epsilon(k)}$-approximation FPT-algorithm of set cover for any unbounded computable function ϵ under W[1] $\neq FPT$.

▶ **Theorem 3.** *Assuming k-SUM hypothesis for any $\delta, \epsilon \in (0,1)$, sufficiently large k and computable function $f : \mathbb{N} \to \mathbb{N}$, there is no $f(k) \cdot N^{\lceil k/2 \rceil - \epsilon}$ time algorithm that can, given an N-vertex set cover instance I, distinguish between*

- $opt(I) \leq k$,
- $opt(I) > \frac{1}{1+\delta} \left(\frac{\log N}{\log \log N}\right)^{\frac{1}{k}}$.

▶ **Theorem 4.** *Assuming W[1] $\neq FPT$, for and computable function $f : \mathbb{N} \to \mathbb{N}$ and unbounded computable function $\epsilon : \mathbb{N} \to \mathbb{N}$, there is no $f(k) \cdot N^{O(1)}$-time algorithm that can, given an N-vertex set cover instance I, distinguish between*

- $opt(I) \leq k$,
- $opt(I) > \log N^{1/\epsilon(k)}$.

Technique contribution. The main technique contribution of this paper is to introduce a gadget that can be used to design gap-producing reductions from the set cover problem to its approximation version and provide a construction of this gadget using (n,k)-*universal sets*. Compared to the reductions in [24], the gap amplification step in this paper is independent of the starting assumptions. This simplifies the proof for showing the inapproximability of the set cover problem. In particular, the inapproximability result in [24] assuming SETH needs some heavy machinery like AG codes to create the gap, while our reduction is completely elementary.

In addition to it simplicity, an important feature of our reduction is that it can be computed by constant depth circuits. Combining this observation with Rossman's $\Omega(n^{k/4})$ size lower bound for constant depth circuits detecting k-clique [33], Wenxin Lai [25] showed that there is no constant-depth circuits of size $f(k)n^{o(\sqrt{k})}$ that can distinguish between a set cover instance with solution size at most k and one whose minimum solution size is at least $(\log n / \log \log n)^{1/\binom{k}{2}}$.

Another advantage of our reduction is that it can give hardness approximation result from assumptions that the distributed PCP technique cannot. If we assume that k-set-cover with large universe set, say $|U| = n^{1/h(k)^k}$, has no $n^{k-\epsilon}$-time algorithm, then our reduction gives

$h(k)$ factor hardness of approximation k-set-cover in $n^{k-\epsilon}$ time. This cannot be achieved by the distributed PCP technique used in [24] due to known lower bounds in communication complexity of set disjointness.

The gap-gadget we introduce in this paper is similar to the bipartite graphs with threshold property in [26, 27]. Such kind of gadgets may have further applications in proving hardness of approximation for other parameterized problems.

2 Preliminaries

For $n, k \in \mathbb{N}$, an (n, k)-universal set is a set of binary strings with length n, such that the restriction to any k indices contains all the 2^k possible binary configurations.

▶ **Lemma 5.** *[See Sections 10.5 and 10.6 of [22]] For $k2^k \leq \sqrt{n}$, (n, k)-universal sets of size n can be computed in $O(n^3)$ time.*

Hypotheses. Below is a list of hardness hypotheses we will use in this paper.

- W[1] $\neq$ *FPT*: for any computable function $f : \mathbb{N} \to \mathbb{N}$, no algorithm can, given an n-vertex graph G and an integer k, decide if G contains a k-clique in $f(k) \cdot n^{O(1)}$ time.
- W[2] $\neq$ *FPT*: for any computable function $f : \mathbb{N} \to \mathbb{N}$, there is no algorithm which, given an n-vertex set cover instance I and an integer k, decides if $opt(I) \leq k$ in $f(k) \cdot n^{O(1)}$ time.
- Exponential Time Hypothesis (ETH)[19, 20]: there exists a $\delta \in (0, 1)$ such that 3-SAT on n variables cannot be solved in $O(2^{\delta n})$ time.
- Strong Exponential Time Hypothesis (SETH)[19, 20] for any $\epsilon \in (0, 1)$ there exists $d \geq 3$ such that d-SAT on n variables cannot be solved in $O(2^{(1-\epsilon)n})$ time.
- k-SUM hypothesis (k-SUM) [1]: for every $k \geq 2$ and $\epsilon > 0$, no $O(n^{\lceil k/2 \rceil - \epsilon})$ time algorithm can, given k sets $S_1, \ldots, S_k$ each with n integers in $[-n^{2k}, n^{2k}]$, decide if there are k integers $x_1 \in S_1, \ldots, x_k \in S_k$ such that $\sum_{i \in [k]} x_i = 0$.

We refer the reader to [18, 15] for more information about the parameterized complexity hypotheses. Using the Sparsification lemma [20], we can assume that the instances of 3-SAT in ETH have Cn clauses for some constant C and the instances of d-SAT in SETH have $C_{d,\epsilon}n$ clauses where $C_{d,\epsilon}$ depends on d and ϵ.

3 Reductions

We start with the definition of (k, n, m, ℓ, h)-gap-gadgets. In Lemma 7, we show how to use theses gadgets to create an (h/k)-gap for the set cover problem. Lemma 10 gives a polynomial time construction of gap-gadgets with $h \leq O(\log n / \log \log n)$ and $\ell = h^k$. Since for every input instance $I = (U, S, E)$ of set cover, our reduction runs in time $|S|^{O(1)}|U|^{\ell}$. If $|U| = \Omega(n)$, we can not afford such running time. Our next step is to prove the hardness of set cover with $U = f(k) \cdot (\log n)^{O(1)}$ based on each of the aforementioned hypotheses.

▶ **Definition 6** ((k, n, m, ℓ, h)-Gap-Gadget)**.** *A (k, n, m, ℓ, h)-Gap-Gadget is a bipartite graph $T = (A, B, E)$ satisfying the following conditions.*

(G1) *A is partitioned into $(A_1, A_2, \ldots, A_m)$. For every $i \in [m]$, $|A_i| = \ell$.*

(G2) *B is partitioned into $(B_1, B_2, \ldots, B_k)$. For every $j \in [k]$, $|B_j| = n$.*

(G3) *For all $b_1 \in B_1, b_2 \in B_2, \ldots b_k \in B_k$, there exist $a_1 \in A_1, \ldots, a_m \in A_m$ such that for all $i \in [m]$ and $j \in [k]$, a_i is adjacent to b_j.*

(G4) *For all $X \subseteq B$ and $a_1 \in A_1, \ldots, a_m \in A_m$, if every a_i has at least $k + 1$ neighbors in X, then $|X| > h$.*

To use this gadget, given a set cover instance $I = (S, U, E)$, we will identify the set B with the set S. Then we construct a new set cover instance $I' = (S', U', E')$ with $S' = S$ such that

(⋆) for any subset X of S' that can cover U', there must exist a vertex $a_i \in A_i$ for every $i \in m$ *witnessing* that X contains a solution of I, i.e., there exists $C \subseteq X$ that can cover U in the instance I and all the vertices of C are adjacent to a_i in the gap-gadget.

It is easy to check the correctness of this reduction:

If there is a k-vertex set X that can cover U, then by (G3) we can pick $a_i \in A_i$ for all $i \in [m]$ such that a_i is adjacent to all vertices in X. This means that X is also a solution of I'.

If $opt(I) > k$, then no matter how we pick $a_i \in A_i$, each a_i must have $k+1$ neighbors in X. This implies that $X > h$ by (G4).

To achieve $(\star)$, we will use the idea of *hypercube set system* from Feige's work [17] (which is also used in [24, 8]). For each $i \in [m]$, we construct a set U^{A_i}. Each element in U^{A_i} can be regarded as a function $f : A_i \to U$. In the new set cover instance, f is covered by $s \in S$ if there exists $a_i \in A_i$ such that a_i is adjacent to s in the gap-gadget and $f(a_i)$ is covered by s in I. More details can be found in the proof of the following lemma.

▶ **Lemma 7.** *There is an algorithm which, given an integer k, an instance $I = (S, U, E)$ of* SetCover*, where $S = S_1 \cup S_2 \ldots \cup S_k$ and $|S_i| = n$ for all $i \in [k]$, and a (k, n, m, ℓ, h)-Gap-Gadget, outputs a set cover instance $I' = (S', U', E')$ with $S' = S$ and $U' = m|U|^\ell$ in $|U|^\ell \cdot n^{O(1)}$ time such that*

- *if there exist $s_1 \in S_1, \ldots, s_k \in S_k$ that can cover U, then $opt(I') \le k$;*
- *if $opt(I) > k$, then $opt(I') > h$.*

Proof. Let $T = (A, B, E_T)$ be the (k, n, m, ℓ, h)-Gap-Gadget. Without loss of generality, assume that for all $i \in [k]$ $B_i = S_i$. The new instance $I' = (S', U', E')$ is defined as follows.

- $S' = S$.
- $U' = (\bigcup_{i \in [m]} U^{A_i})$.
- For all $s \in S'$ and $f \in U^{A_i}$ where $i \in [m]$, E' contains $\{s, f\}$ if there exists an $a \in A_i$ such that
  - **(E'1)** $\{s, f(a)\} \in E$,
  - **(E'2)** $\{a, s\} \in E_T$.

Completeness. If $opt(I) \le k$, then there exist $s_1 \in S_1, \ldots, s_k \in S$ that can cover the whole set U. We will show that for every $f \in U'$, f is covered by some vertex in $\{s_1, s_2, \cdots, s_k\}$. Firstly, by (G3), there exist $a_1 \in A_1, \ldots, a_m \in A_m$ such that $a_i s_j \in E_T$ for all $i \in [m]$ and $j \in [k]$. Assume that $f \in U^{A_i}$ for some $i \in [m]$. Observe that $f(a_i) \in U$ must be covered by some s_j with $j \in [k]$, i.e., $\{s_j, f(a_i)\} \in E$. Since $\{a_i, s_j\} \in E_T$ and $\{s_j, f(a_i)\} \in E$, according to the definition of E', we must have $\{s_j, f\} \in E'$.

Soundness. Suppose $opt(I) > k$. Let $X \subseteq S'$ be a set covering U'. For every $a \in A$, let $N^T(a)$ be the set of neighbors of a in T. We have the following claim.

▷ Claim 8. For every $i \in [m]$ there exists $a_i \in A_i$ such that $|N^T(a_i) \cap X| \ge k + 1$.

Proof. Suppose there exists an $i \in [m]$ such that for all $a \in A_i$, $|N^T(a) \cap X| \le k$. Since $opt(I) > k$, every solution of I has size at least $k + 1$. It follows that for every $a \in A_i$, there exists some $u_a \in U$ such that u_a is not covered by $N^T(a) \cap X$ in the set cover instance I. Define a function $f \in U^{A_i}$ such that $f(a) = u_a$ for every $a \in A_i$. We claim that f is not covered by X. Otherwise, suppose there exists an $s \in X$ that can cover f. According to the definition of E', there must exists an $a \in A_i$ such that (E'1) and (E'2) hold. However, if

$s \in N^T(a) \cap X$, then $\{s, f(a)\} = \{s, u_a\} \notin E$. On the other hand, if $s \notin N^T(a) \cap X$, then $\{a, s\} \notin E_T$. In both cases, we obtain contradictions. ◁

By Claim 8, we can pick $a_i \in A_i$ for each $i \in [m]$ such that every a_i has at least $k+1$ neighbors in X. By the property of Gap-Gadget, $|X| > h$. ◀

▶ Remark 9. Recall that the greedy algorithm can approximate the set cover problem within a $(1+\ln|U|)$-approximation ratio. If one could construct a gap-gadget for parameters satisfying

$$k(1+\ln|U'|) = k(1+\ell\ln|U|+\ln m) < h,$$

then applying the greedy algorithm on input I' could decide whether $opt(I) = k$ in $|U|^\ell \cdot n^{O(1)}$ time.

It is well known that given a CNF formula ϕ on n variables, one can construct a set cover instance $I = (S, U, E)$ with $|U| = O(n)$ and $|S| = \Theta(k2^{n/k})$ in $2^{O(n/k)}$ time such that ϕ is satisfiable if and only if $opt(I) = k$. This implies that, assuming ETH there is no algorithm that can construct $(k, |S|, m, \ell, h)$-gap-gadgets with $k(1+\ell ln|U|+\ln m) < h$ and $|U|^\ell \le 2^{o(n)}$ in $2^{o(n)}$ time.

3.1 Construction of Gap-Gadgets

In [27], a similar gadget is used to prove the parameterized complexity of k-Biclique. One would wonder if the randomized construction from [27] can be used to construct the gap-gadget in this paper. Informally, the gadget in [27] is a bipartite random graph $T = (A, B, E)$ satisfying the following properties with high probability:

(T1) a k-vertex set in B has $m = n^{\Theta(1/k)}$ common neighbors;

(T2) any $(k+1)$-vertex set in B has at most $O(k^2)$ common neighbors.

It is not hard to show that if $Y \subseteq A$ is an m-vertex set and every vertex in Y has at least $k+1$ neighbors in $X \subseteq B$, then $|X| \ge \sqrt[k+1]{\frac{|Y|}{O(k^2)}}$ by (T2) and the pigeonhole principle. We may partition the vertex set A into m parts. Each part contains $n^{1-\Theta(1/k)}$ vertices. This gives us a gap-gadget with large gap $h = \sqrt[k+1]{\frac{m}{O(k^2)}}$ and $\ell = n^{1-\Theta(1/k)}$. Unfortunately, such gadget does not suit our purpose. We need a gap-gadget with $\ell \le \log n/\log\log n$. In this section, we provide a construction using universal sets.

▶ **Lemma 10.** *There is an algorithm that can, for every $k, h, n \in \mathbb{N}$ with $k\log\log n \le \log n$ and $h \le \frac{\log n}{(2+\epsilon)\log\log n}$, compute a $(k, n, n\log h, h^k, h)$-Gap-Gadget in $O(n^4)$ time.*

Proof. Let $m = n\log h$ and $K = h\log h$. Note that $(\log m)/2 = (\log n + \log\log h)/2 \ge (2+\epsilon)h\log h/2 \ge \log h + \log\log h + h\log h = \log K + K$, i.e., $K2^K \le \sqrt{m}$. By Lemma 5, an (m, K)-universal set $S = \{s_1, s_2, \ldots, s_m\}$ can be constructed in $O(m^3) \le O(n^4)$ time. Partition every $s \in S$ into $n = \frac{m}{\log h}$ blocks so that each block has length $\log h$. Interpret the values of blocks as integers in $[h]$. We obtain an $m \times n$ matrix M by setting the value $M_{r,c}$ equal to the value of the c-th block of s_r. The matrix M satisfies the following conditions.

(M1) For all $r \in [m]$ and $c \in [n]$, $M_{r,c} \in [h]$.

(M2) For any set $C \subseteq [n]$ with $|C| \le h$, there exists a row $r \in [m]$ such that $|\{M_{r,c} : c \in C\}| = |C|$.

Condition (M1) is obvious. To see why (M2) holds, for each $C \subseteq [n]$ with $|C| \le h$, let C' be the set of indices corresponding to the blocks in C. Note that $|C'| = |C|\log h \le h\log h = K$. By the property of (m, K)-universal set, there exists an $s_r \in S$ such that each block in C takes distinct value. It follows that $|\{M_{r,c} : c \in C\}| = |C|$.

For each $i \in [m]$, let

$$A_i = \{(a_1, a_2, \ldots, a_k) : \text{for all } j \in [k],\ a_j \in [h]\}.$$

Note that $|A_i| = h^k$. For each $j \in [k]$, let $B_j = [n]$. Let $T = (A, B, E)$ be a bipartite graph with

- $A = \bigcup_{i\in[m]} A_i$.
- $B = \bigcup_{j\in[k]} B_j$.
- $E = \{\{\vec{a}, b\} : \vec{a} \in A_i, b \in B_j \text{ and } M_{i,b} = \vec{a}[j] \text{ for all } j \in [k]\}$.

We will show that T is an (k, n, m, h^k, h)-gap-gadget. Obviously, T satisfies (G1) and (G2).

T satisfies (G3). For any $b_1 \in B_1, b_2 \in B_2, \ldots, b_k \in B_k$. We define $\vec{a}_i \in A_i$ by setting

$$\vec{a}_i = (M_{i,b_1}, M_{i,b_2}, \ldots, M_{i,b_k}).$$

It is routine to check that $\{\vec{a}_i, b_j\} \in E$ for all $i \in [m]$ and $j \in [k]$.

T satisfies (G4). Let $X \subseteq B$ and $\vec{a}_1 \in A_1, \vec{a}_2 \in A_2, \ldots, \vec{a}_m \in A_m$. Suppose for every $i \in [m]$, $\vec{a}_i$ has at least $k+1$ neighbors in X and $|X| \leq h$. By (M2), there exists an $r \in [m]$ such that $|\{M_{r,c} : c \in X\}| = |X|$. Since $\vec{a}_r$ has at least $k+1$ neighbors in X, there exists an $j \in [k]$ such that $\vec{a}_r$ has two neighbors b, b' in $X \cap B_j$. According to the definition of E, we must have

$$M_{r,b} = M_{r,b'} = \vec{a}_r[j].$$

This contradicts the fact that $|\{M_{r,c} : c \in X\}| = |X|$. ◀

The construction above produces gap-gadgets with $\ell = h^k$. Note that the parameter h is related to the inapproximation factor we will get for the set cover problem and the running time of our reduction is $n^{O(1)}|U|^\ell$. We want to set h as large as possible while keeping the running time of reduction in $f(k) \cdot n^{O(1)}$. Assuming $|U| = g(k) \cdot (\log n)^{O(1)}$, the best we can achieve is $h = (\log n / \log\log n)^{1/k}$.

On the probabilistic construction. A natural question is, can we construct gap-gadgets with better parameters h and ℓ, say $\ell = h = o(\log n)$, using the probabilistic method?

Consider the probability space of bipartite random graphs on the vertex sets $A = A_1 \cup A_2 \cup \cdots \cup A_m$ and $B = B_1 \cup B_2 \cup \cdots B_k$, where $|A_i| = \ell$ and $|B_j| = n$. Let p be the edge probability. Each bipartite graph T on $A \cup B$ has probability $Pr[T] = p^{|E(T)|}(1-p)^{|A|\cdot|B|-|E(T)|}$. Fix k vertices $b_1, b_2, \ldots, b_k$ in B. Let X_{good} be the random variable that for every bipartite graph T, $X_{good}(T)$ is the number of complete bipartite subgraphs of T which contains exactly one vertex in each A_i and the k vertices $b_1, b_2, \ldots, b_k$ in B. Let X_{bad} be the random variable that for every bipartite graph T, $X_{bad}(T)$ is the number of subgraphs of T with h vertices in B and one vertex in each A_i such that each vertex in A_i has at least $k+1$ neighbors in B. We want to set the edge probability p so that $\Pr[X_{bad}(T) \geq 1] + \Pr[X_{good}(T) = 0] \leq 1 - n^{-c}$ for some constant $c > 0$. One way to bound $\Pr[X_{bad}(T) \geq 1]$ above is to use Markov's inequality, which gives us $\Pr[X_{bad}(T) \geq 1] \leq E[X_{bad}]$. So we might assume that $E[X_{bad}] < 1$. On the other hand, we have $E[X_{good}] \geq \Pr[X_{good}(T) \geq 1] \geq n^{-c}$. Note that expectations of these two random variables are $E[X_{good}] = \ell^m p^{km}$ and $E[X_{bad}] = \ell^m \binom{n}{h} p^{(k+1)m} \binom{h}{k+1}^m$. We deduce that

$$m \log \ell + mk \log p > -c \log n$$

and

$$m \log \ell + h \log n + m(k+1) \log p + m(k+1) \log h < 0.$$

Thus

$$\frac{c \log n}{mk} + \frac{\log \ell}{k} > \frac{\log \ell}{(k+1)} + \frac{h \log n}{m(k+1)} + \log h. \tag{1}$$

We might choose m large enough so that the terms $\frac{c \log n}{mk}$ and $\frac{h \log n}{m(k+1)}$ in (1) become relatively small. In order to make (1) hold, we have to set $\ell \geq h^{O(k^2)}$. This does not give us better (k, n, m, ℓ, h)-gap-gadgets.

3.2 Proofs of Theorem 1 and Theorem 2

▶ **Lemma 11.** *There is an algorithm, which given $k \in \mathbb{N}$, $\delta > 0$ with $(1 + 1/k^3)^{1/k} \leq (1+\delta)/(1+\delta/2)$ and $(1+\delta/2)^k \geq 2k^4$ and a SAT instance ϕ with n variables and Cn clauses, where n is much larger than k and C, outputs an integer $N \leq 2^{n/k+n/k^3}$ and a set cover instance I satisfying the following conditions in $2^{5n/k}$ time.*

- $|S(I)| + |U(I)| \leq N$.
- *If ϕ is satisfiable, then $opt(I) \leq k$.*
- *If ϕ is not satisfiable, then $opt(I) > \frac{1}{1+\delta} \cdot \sqrt[k]{\frac{\log N}{\log \log N}}$.*

Proof. Let k be a positive integer and ϕ be a CNF with n variables and Cn clauses. We first construct a set cover instance $I' = (S', U', E')$ as follows. Partition the variable set into k parts, each having at most $\lceil n/k \rceil$ variables. For each $i \in [k]$, let S_i be the set of assignments to the i-th part. Let $S' = S_1 \cup \cdots \cup S_k$. Let U' be the set consisting of all the clauses of ϕ and k additional nodes $u_1, u_2, \ldots, u_k$. For every $i \in [k]$ and assignment $s \in S_i$, we add an edge between s and u_i. If the assignment $s \in S'$ satisfies a clause $u \in U'$, we also add an edge between u and s. The set cover instance I' has the following properties.

- If ϕ is satisfiable, then $opt(I') = k$. Moreover, there exist k vertices $s_1 \in S_1, \cdots, s_k \in S_k$ that can cover the whole set U'.
- If ϕ is not satisfiable, then $opt(I') > k$.
- $|U'| = k + Cn$.
- $|S'| \leq k2^{n/k}$.

Let $M = k2^{n/k} \geq |S|$ and $N = M^{1+1/k^3} \leq 2^{n/k+n/k^3}$. Note that $\log M / \log \log M \geq n/(k \log n) \geq k$. Applying Lemma 10 with $k \leftarrow k$, $n \leftarrow M$, $\ell \leftarrow \frac{\log M}{(1+\delta/2)^k \log \log M}$, $h \leftarrow \frac{1}{1+\delta/2} \cdot \sqrt[k]{\frac{\log M}{\log \log M}}$ and $m \leftarrow M \log h \leq M \log \log M$, we obtain a gap-gadget T in $O(M^4 \leq 2^{5n/k})$ time. Using Lemma 7 on I' and T, we obtain our target set cover instance $I = (S, U, E)$ satisfying the following properties.

- If ϕ is a yes-instance, then $opt(I) \leq k$.
- If ϕ is a no-instance, then $opt(I) > \frac{1}{1+\delta/2} \cdot \sqrt[k]{\log M / \log \log M}$. Using $(1 + 1/k^3)^{1/k} \leq (1 + \delta)/(1 + \delta/2)$, we get $opt(I) > \frac{1}{1+\delta} \cdot \sqrt[k]{\log N / \log \log N}$.
- $|S| = |S| \leq k2^{n/k}$.
- $|U| \leq M \log \log M \cdot |U|^{\frac{\log M}{(1+\delta/2)^k \log \log M}} = M \log \log M \cdot (k + Cn)^{\frac{\log M}{(1+\delta/2)^k \log \log M}}$.

The number of vertices in I is

$$\begin{aligned}
|S(I)| + |U(I)| &\leq M + M \log\log M \cdot (k + Cn)^{\frac{\log M}{(1+\delta/2)^k \log\log M}} \\
&\leq M + M \log\log M \cdot (2Ck \log M)^{\frac{\log M}{(1+\delta/2)^k \log\log M}} \\
&\leq M + M \log\log M \cdot (\log M)^{\frac{2\log\log M}{(1+\delta/2)^k \log\log M}} \quad \text{(using } \log M \geq 2Ck \text{ for large } n\text{)} \\
&\leq M + M \log\log M \cdot M^{\frac{2}{(1+\delta/2)^k}} \\
&\leq M + M \log\log M \cdot M^{1/k^4} \quad \text{(using } (1+\delta/2)^k \geq 2k^4\text{)} \\
&\leq M^{1+1/k^3} \quad \text{(using } M^{1/k^3} \geq 1 + M^{1/k^4} \log\log M \text{ for large } n\text{)} \\
&= N.
\end{aligned}$$

◀

Now we are ready to prove Theorem 1. Suppose for some computable function f, there is an $f(k) \cdot N^{k-\epsilon}$-time algorithm that can, for every N-vertex set cover instance I and every integer k, distinguish between $opt(I) \leq k$ and $opt(I) \geq \frac{1}{1+\delta} \cdot \sqrt[k]{\frac{\log N}{\log\log N}}$. For every $\delta \in (0,1)$, choose $k \in \mathbb{N}$ large enough so that $(1+1/k^3)^{1/k} \leq (1+\delta)/(1+\delta/2)$ and $(1+\delta/2)^k \geq 2k^4$ hold. Let $\epsilon' = 1 - \epsilon/k + 1/k^2$, by SETH, there exists an integer d such that d-SAT with n variables cannot be solved in $2^{n(1-\epsilon')}$-time. Given an instance ϕ of d-SAT with n variables and m clauses. By the sparsification lemma [20], we can assume that $m = C_{d,\epsilon'} \cdot n$ for some constant $C_{d,\epsilon'}$ depending on d and ϵ'. Without loss of generality, assume that n is much larger than k. Applying Lemma 11 on ϕ and k, we obtain a set cover instance I with $N \leq 2^{n/k+n/k^3}$ vertices in time $2^{5n/k} \leq 2^{\epsilon n}$ for $k \geq 5/\epsilon$. Then we use the approximation algorithm to decide if $opt(I) \leq k$ or $opt(I) \geq \frac{1}{1+\delta} \cdot \sqrt[k]{\frac{\log N}{\log\log N}}$. Thus we can solve d-SAT in time $2^{\epsilon n} + f(k) \cdot N^{k-\epsilon} \leq 2^{\epsilon n} + f(k) \cdot 2^{(n/k+n/k^3)(k-\epsilon)} \leq 2^{n(1-\epsilon/k+1/k^2)} = 2^{n(1-\epsilon')}$, which contradicts SETH.

Theorem 2 can be proved similarly. By ETH, there exists $\epsilon > 0$ such that 3-SAT on n variables cannot be solved in $2^{\epsilon n}$ time. Let $\epsilon' = \epsilon/2$. For every 3-SAT instance ϕ with n variable and Cn clause, where n is much larger than k, apply Lemma 11 to obtain a set cover instance I with $N = 2^{n/k+n/k^3}$ vertices in $2^{5n/k} \leq 2^{\epsilon' n}$ time. If there is an $f(k) \cdot N^{\epsilon' k}$-time algorithm that can distinguish between $opt(I) \leq k$ and $opt(I) > \frac{1}{1+\delta} \cdot \sqrt[k]{\frac{\log N}{\log\log N}}$, then we can decide whether ϕ is satisfiable in time $2^{\epsilon' n} + f(k) \cdot 2^{(n/k+n/k^3)\cdot\epsilon' k} \leq 2^{\epsilon n}$.

3.3 Proof of Theorem 3

We use a lemma in [2] to reduce k-SUM to k-VECTOR-SUM over small numbers. Then we present a reduction from k-VECTOR-SUM to set cover.

▶ **Lemma 12** (Lemma 3.1 of [2]). *Let $k, p, d, s, M \in \mathbb{N}$ satisfy $k < p$, $p^d \geq kM + 1$, and $s = (k+1)^{d-1}$. There is a collection of mappings $f_1, \ldots, f_s : [0, M] \times [0, kM] \to [-kp, kp]^d$, each computable in time $O(poly \log M + k^d)$, such that for all numbers $x_1, \ldots, x_k \in [0, M]$ and targets $t \in [0, kM]$,*

$$\sum_{j=1}^{k} x_j = t \Leftrightarrow \exists i \in [s] \text{ such that } \sum_{j=1}^{k} f_i(x_j, t) = \vec{0}.$$

▶ **Lemma 13.** *There is an algorithm which, given k sets $S_1, S_2, \ldots, S_k$ where S_i is a set of n vectors in $[-f(k), f(k)]^{g(k)\log n}$ for some computable functions f and g, outputs a set cover instance $I = (S, U, E)$ with $|U| \leq k^{(2f(k))^{k-1}} g(k) \log n$ and $S = S_1 \cup S_2 \cup \ldots \cup S_k$ in $k^{(2f(k))^{k-1}} g(k) n^{O(1)}$-time such that*

(i) *if there exist* $\vec{x}_1 \in S_1, \ldots, \vec{x}_k \in S_k$ *such that* $\sum_{i\in[k]} \vec{x}_i = \vec{0}$*, then* $\{\vec{x}_1, \ldots, \vec{x}_k\}$ *covers* U*;*
(ii) *if the sum of any* k *vectors* $\vec{x}_1 \in S_1, \ldots \vec{x}_k \in S_k$ *is not zero, then* $opt(I) > k$*.*

Proof. Let $D = \{(d_1, \ldots, d_k) \in [-f(k), f(k)]^k : \sum_{i\in[k]} d_i = 0\}$. Note that $|D| \leq (2f(k))^{k-1}$. Suppose $D = \{\vec{a}_1, \ldots, \vec{a}_{|D|}\}$. For every $j \in [g(k)\log n]$, let $U_j = [k]^{|D|}$. We define the target set cover instance $I = (S, U, E)$ as follows.

- $S = S_1 \cup \cdots \cup S_k$.
- $U = \bigcup_{i\in[g(k)\log n]} U_i$.
- For every $\vec{x} \in S_i$ and every $\vec{u} \in U_j$, we add an edge $\{\vec{x}, \vec{u}\}$ into E if there exists $\ell \in [|D|]$ such that $\vec{u}[\ell] = i$ and $\vec{x}[j] = \vec{a}_\ell[i]$.

Completeness. Suppose there exist $\vec{x}_1 \in S_1, \ldots, \vec{x}_k \in S_k$ such that $\sum_{i\in[k]} \vec{x}_i = \vec{0}$. Then for all $j \in [g(k)\log n]$ we have $\vec{x}_1[j] + \vec{x}_2[j] + \ldots + \vec{x}_k[j] = 0$, i.e.,

$$(\vec{x}_1[j], \vec{x}_2[j], \ldots, \vec{x}_k[j]) = \vec{a}_\ell \in D \text{ for some } \ell \in [|D|]. \tag{2}$$

For all $\vec{u} \in U_j$, let $i = \vec{u}[\ell] \in [k]$. Then by (2), $\vec{x}_i[j] = \vec{a}_\ell[i]$. It follows that $\{\vec{x}_i, \vec{u}\} \in E$.

Soundness. Suppose the sum of any k vectors in $S_1 \cup \cdots \cup S_k$ is not zero. Let X be a subset of S with $|X| \leq k$, we need to show that X does not cover U. Firstly, we note that if $X \cap S_i = \emptyset$ for some $i \in [k]$, then the vector $\vec{u} = (i, i, \ldots, i) \in [k]^{|D|}$ is not covered by any vector in X. Now assume that $X = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_k\}$ and $\vec{x}_i \in S_i$ for all $i \in [k]$. Since $\sum_{i\in[k]} \vec{x}_i \neq \vec{0}$, there exists a $j \in [g(k)\log n]$ such that

$$\sum_{i\in[k]} \vec{x}_i[j] \neq 0.$$

We deduce that

$$(\vec{x}_1[j], \vec{x}_2[j], \ldots, \vec{x}_k[j]) \notin D.$$

In other word, for all $\ell \in [|D|]$, there exists an $i_\ell \in [k]$ such that

$$\vec{x}_{i_\ell}[j] \neq \vec{a}_\ell[i_\ell]. \tag{3}$$

Define a vector $\vec{u} \in U_j$ such that for all $\ell \in [|D|]$,

$$\vec{u}[\ell] = i_\ell. \tag{4}$$

Suppose $\vec{u}$ is covered by $x_i \in X$, then by the definition, there exists $\ell \in [|D|]$ such that $i = \vec{u}[\ell] = i_\ell$ and $\vec{x}_{i_\ell}[j] = \vec{a}_\ell[i_\ell]$, which contradicts (3) and (4). ◀

Proof of Theorem 3. Given k sets $S_1, \ldots, S_k$ of integers in $[-n^{2k}, n^{2k}]$. Let $p = k^{4k^{c+1}}$, $M = 2n^{2k}$ and $d = \log n/k^c$. Without loss of generality, assume that k is large and n is much larger than k, we have $p^d = k^{4k\log n} \geq n^{4k} \geq 2kn^{2k} + 1$. On the other hand, for any $\epsilon > 0$, we can pick c such that $s = (k+1)^d = n^{\log(k+1)/k^c} \leq n^{\epsilon/4}$. Applying Lemma 12, we obtain a collection of mappings $f_1, \ldots, f_s : [0, M] \times [0, kM] \to [-kp, kp]^d$ in $O(poly\log M + k^d)$ time such that

- there exist $x_1 \in S_1, \ldots, x_k \in S_k$ with $\sum_{j\in[k]} x_j = 0$ if and only if there exist $i \in [s]$ such that $\sum_{j\in[k]} f_i(x_j + n^{2k}, kn^{2k}) = \vec{0}$.

Using Lemma 10, we construct a $(k, n, O(n\log\log n), \frac{\log n}{(1+\delta/2)^k \log\log n}, \frac{1}{(1+\delta/2)} \cdot (\frac{\log n}{\log\log n})^{1/k})$-gap-gadget T for some small $\delta > 0$. For every $i \in [s]$, and $j \in [k]$, let $S_j^i = \{f_i(x + n^{2k}, kn^{2k}) : x \in S_j\}$. Applying Lemma 7 to $S_1^i, S_2^i, \ldots, S_k^i$ and T, we obtain a set cover instance I_i with $S(I_i) = S_1^i \cup S_2^i \ldots S_k^i$ and $|U(I_i)| \leq n\log\log n \cdot (g(k)\log n)^{\frac{\log n}{(1+\delta/2)^k \log\log n}} \leq n^{1+1/k^3}$. The set cover instances $I_1, \ldots, I_s$ satisfy the following properties.

- If there exist $x_1 \in S_1, \ldots, x_k \in S_k$ with $\sum_{j\in[k]} x_j = 0$, then there exist $i \in [s]$ and $y_1 = f_i(x_1 + n^{2k}, n^{2k}) \in S_1^i \ldots y_k = f_i(x_k + n^{2k}, n^{2k}) \in S_k^i$ such that $y_1, \ldots, y_k$ cover $U(I_i)$.
- If there are no $x_1 \in S_1, \ldots, x_k \in S_k$ with $\sum_{j\in[k]} x_j = 0$, then for all $i \in [s]$, $opt(I_i) > \frac{1}{1+\delta/2} \cdot \left(\frac{\log n}{\log\log n}\right)^{1/k}$.

Let $N = n^{1+1/k^2}$. We have

$$|S(I_i)| + |U(I_i)| \leq kn + n^{1+1/k^3} \leq N,$$

$$f(k) \cdot N^{\lceil k/2\rceil - \epsilon} \leq n^{\lceil k/2\rceil - \epsilon + 1/k},$$

and

$$\frac{1}{(1+\delta)} \left(\frac{\log N}{\log\log N}\right)^{1/k} \leq \frac{1}{(1+\delta/2)} \left(\frac{\log n}{\log\log n}\right)^{1/k}.$$

For every $i \in [s]$, we apply the $f(k) \cdot N^{\lceil k/2\rceil-\epsilon}$-time algorithm to decide if $opt(I_i) \leq k$ or $opt(I_i) > \frac{1}{1+\delta} \cdot (\log N/\log\log N)^{1/k}$. If for some $i \in [s]$, it found that $opt(I_i) \leq k$, then we know that the input instance of k-SUM is a yes-instance. The running time is $O(poly\log M + k^d) + f(k) \cdot N^{\lceil k/2\rceil-\epsilon} \leq O(poly\log M + k^d) + s \cdot n^{\lceil k/2\rceil-\epsilon+1/k} \leq n^{\lceil k/2\rceil-\epsilon/2}$ for large k. ◀

3.4 Proof of Theorem 4

Firstly, we give a reduction from Clique to SetCover which produces instances with logarithmic sized universe set. The main idea of this reduction is due to Karthik et al. [24].

▶ **Lemma 14.** *There is an $n^{O(1)}$-time algorithm which, given an integer k, an n-vertex graph G with $V(G) = V_1 \cup V_2 \cup \cdots \cup V_k$ such that $G[V_i]$ is an independent set for all $i \in [k]$, outputs a set cover instance $I = (S, U, E)$ with $|U| = k^{O(1)}\log n$ and $S = E(G) = \bigcup_{\{i,j\}\in\binom{[k]}{2}} S_{\{i,j\}}$, where each $S_{\{i,j\}}$ is the set of edges between V_i and V_j, such that*

- **(i)** *if G contains a k-clique, then $opt(I) \leq \binom{k}{2}$. Moreover, there exists a $\binom{k}{2}$-sized subset of S, which contains exactly one vertex from each $S_{\{i,j\}}$ ($\{i,j\} \in \binom{[k]}{2}$), that can cover U;*
- **(ii)** *if G contains no k-clique, then $opt(I) > \binom{k}{2}$.*

Proof. We will construct a set cover instance I such that if G has a k-clique, then we can select its $\binom{k}{2}$ edges to cover the whole universe set. For every $v \in V(G)$, denote by $encode(v) \in \{0,1\}^{\log n}$ the binary string representation of v. For every $\ell \in [\log n]$, the ℓth bit of $encode(v)$ is $encode(v)[\ell]$. For every $i \in [k]$, let $\sigma_i : [k] \setminus \{i\} \to [k-1]$ be an arbitrary bijection. Our target set cover instance $I = (S, U, E)$ is defined as follows.

- $S = E(G) = \bigcup_{\{i,j\}\in\binom{[k]}{2}} S_{\{i,j\}}$, where $S_{\{i,j\}} = \{\{v_i, v_j\} : v_i \in V_i, v_j \in V_j, \{v_i, v_j\} \in E(G)\}$.
- $U = [k] \times [k-1]^{\{0,1\}} \times [\log n]$.
- For $s = \{v_i, v_j\} \in S$ and $u = (i, f, \ell) \in U$ we add $\{s, u\}$ into E if

$$v_i \in V_i,\ v_j \in V_j \text{ and } f(encode(v_i)[\ell]) = \sigma_i(j).$$

The set cover instance I satisfies the following conditions.

- If G contains a k-clique, then there exists a $\binom{k}{2}$-sized subset of S which contains exactly one vertex from each $S_{\{i,j\}}$ ($\{i,j\} \in \binom{[k]}{2}$) that can cover U. Suppose that $v_1 \in V_1, \ldots, v_k \in V_k$ induce a k-clique. Let $X = \{\{v_i, v_j\} : \{i,j\} \in \binom{[k]}{2}\}$. We will show that X covers the whole set U. For any $(i,f,\ell) \in U$, let $b = \mathit{encode}(v_i)[\ell]$. Since $f(b) \in [k-1]$, there must exist a $j \in [k] \setminus \{i\}$ such that $\sigma_i(j) = f(b)$. By the definition of E, $\{v_i, v_j\}$ is adjacent to (i,f,ℓ) .
- If G does not contain a k-clique, then $\mathit{opt}(I) > \binom{k}{2}$. Let $X \subseteq S$ be a set such that $|X| \leq \binom{k}{2}$ and X covers U.
  For each $\{i,j\} \in \binom{[k]}{2}$, define

  $$X_{\{i,j\}} = \{\{v_i, v_j\} : v_i \in V_i, v_j \in V_j, \{v_i, v_j\} \in X\}.$$

  We claim that for every $\{i,j\} \in \binom{[k]}{2}$, $|X_{\{i,j\}}| > 0$. Otherwise let $f(0) = f(1) = \sigma_i(j)$ and consider the vertex $(i,f,1) \in U$. According to the definition of E, if a vertex $\{v,u\} \in S$ covers $(i,f,1)$, then either v or u must be in V_i. Let us assume $v \in V_i$ and $u \in V_{j'}$ for some $j' \in [k] \setminus \{i\}$. We must have $f(\mathit{encode}(v_i)[1]) = \sigma_i(j')$. However, if $j \neq j'$, then $f(0) = f(1) = \sigma_i(j) \neq \sigma_i(j')$.
  Since $\binom{k}{2} \geq |X| = \sum_{\{i,j\} \in \binom{[k]}{2}} |X_{\{i,j\}}|$ and $|X_{\{i,j\}}| > 0$, we conclude that $|X_{\{i,j\}}| = 1$ for all $\{i,j\} \in \binom{[k]}{2}$.
  For every $i \in [k]$ and distinct $j, j' \in [k] \setminus \{i\}$, let $\{\{v, v_j\}\} = X_{\{i,j\}}$ and $\{\{v', v_{j'}\}\} = X_{i,j'}$, where $v, v' \in V_i$, we claim that $v = v'$. Otherwise, since $v \neq v'$ there exists $\ell \in [\log n]$ such that $\mathit{encode}(v)[\ell] \neq \mathit{encode}(v')[\ell]$. Now consider a function f with $f(\mathit{encode}(v')[\ell]) = \sigma_i(j)$ and $f(\mathit{encode}(v)[\ell]) = \sigma_i(j')$. The vertex (i,f,ℓ) must be covered by some $\{x,y\}$ with $x \in V_i$ and $y \in V_h$ such that $\sigma_i(h) = f(\mathit{encode}(v)[\ell]) \in \{\sigma_i(j), \sigma_i(j')\}$. We must have $y \in V_j$ or $y \in V_{j'}$. Since $|X_{\{i,j\}}| = |X_{\{i,j'\}}| = 1$, we deduce that either $\{x,y\} = \{v, v_j\}$ or $\{x,y\} = \{v', v_{j'}\}$. However, if $\{x,y\} = \{v, v_j\}$, we must have $\sigma_i(j) = f(\mathit{encode}(v)[\ell]) = \sigma_i(j') \neq \sigma_i(j)$, a contradiction. Similarly, if $\{x,y\} = \{v', v_{j'}\}$, then $\sigma_i(j') = f(\mathit{encode}(v')[\ell]) = \sigma_i(j) \neq \sigma_i(j')$. We conclude that the vertex (i,f,ℓ) can not be covered by X.
  Now we have for every $i \in [k]$, there exists a $v_i \in V_i$ such that

  $$\{v_i\} = \bigcap_{j \in [k] \setminus \{i\}, e \in X_{\{i,j\}}} e.$$

  Obviously, for every $\{i,j\} \in \binom{[k]}{2}$, $\{\{v_i, v_j\}\} = X_{\{i,j\}}$. This implies that $\{v_1, v_2, \ldots, v_k\}$ is a k-clique in G. ◀

Proof of Theorem 4. Given an n-vertex graph G and a positive integer k, we invoke Lemma 14 to obtain a set cover instance $I = (S, U, E)$ with $|S| = |E(G)|$ and $|U| \leq k^3 \log n$ satisfying (i) and (ii). Let $m = |S|$. Then we use Lemma 10 to construct a $(\binom{k}{2}, m, n^{O(1)}, \frac{\log m}{\log\log m}, \frac{\log m}{\log\log m}^{1/\binom{k}{2}})$-gap-gadget T in $m^{O(1)} = n^{O(1)}$ time. Applying Lemma 7 on I and T, we finally obtain our target set cover instance $I' = (S', U', E')$ with the following properties:

- if G has a k-clique, then $\mathit{opt}(I') = \binom{k}{2}$,
- if G has no k-clique, then $\mathit{opt}(I') > \left(\frac{\log m}{\log\log m}\right)^{1/\binom{k}{2}}$,
- $|S'| = |E(G)| = m$,
- $|U'| = (k^3 \log n)^{\log m / \log\log m} = m^{1+o(1)}$.

Let $N = |U'| + |S'|$. We have $N = n^{O(1)}$. Since ϵ is an unbounded computable function, there is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that $k' = g(k) > \binom{k}{2}$ and $\epsilon(k') > \binom{k}{2}$. When n is large enough,

$$\frac{\log m}{\log\log m}^{1/\binom{k}{2}} \geq \frac{\log N}{O(\log\log N)}^{1/\binom{k}{2}} \geq (\log N)^{1/\epsilon(k')}.$$

Any $f(k') \cdot N^{O(1)}$ time algorithm that can distinguish between $opt(I') \leq k'$ and $opt(I') > (\log N)^{\frac{1}{\epsilon(k')}}$ can be used to decide if an input graph G has k-clique in $f(g(k))n^{O(1)}$ time. ◀

4 Conclusion

We have improved the hardness approximation factor for the parameterized set cover problem using a simple reduction. Our result shows that in order to prove inapproximability of parameterized set cover, it suffices to prove the hardness of set cover problem with small universe set. A natural question is:

Is there any algorithm that can, given an n-vertex set cover instance I and an integer k, outputs a new instance I' and an integer k' in $f(k) \cdot n^{O(1)}$ time for some computable function $f : \mathbb{N} \to \mathbb{N}$ such that

- $k' = g(k)$ for some computable function $g : \mathbb{N} \to \mathbb{N}$,
- $opt(I) \leq k$ if and only if $opt(I') \leq k'$,
- $|U(I')| \leq h(k) \cdot (\log |S(I')|)^{O(1)}$ for some computable function $h : \mathbb{N} \to \mathbb{N}$.

A positive answer to the above question would imply that SetCover parameterized by the optimum solution size has no $(\log n)^{1/\epsilon(k)}$-approximation FPT algorithm assuming W[2] $\neq FPT$. Of course, if we just want a ρ-factor hardness of approximation, then it suffices to have $|U(I')| \leq h(k)|S(I')|^{O(1/\rho^k)}$. Note that using Dynamic Programming, SetCover can be solved in $2^{|U(I)|}(|U(I)| + |S(I)|)^{O(1)}$ time [12]. We do not expect to reduce the size of universe set below $o(k \log n)$ under ETH.

Our hardness result is far from matching the $(1+\ln n)$ approximation ratio of the greedy algorithm in polynomial time. Could it be the case that there exists a $(\ln n)^{1/\rho(k)}$-approximation algorithm for SetCover with running time $n^{k-\epsilon}$? What is the best approximation ratio we can achieve for parameterized set cover in $n^{k-\epsilon}$ time?

References

1. Amir Abboud and Kevin Lewi. Exact weight subgraphs and the k-sum conjecture. In *International Colloquium on Automata, Languages, and Programming*, pages 1–12. Springer, 2013.
2. Amir Abboud, Kevin Lewi, and Ryan Williams. Losing weight by gaining edges. In *European Symposium on Algorithms*, pages 1–12. Springer, 2014.
3. Amir Abboud, Aviad Rubinstein, and Ryan Williams. Distributed PCP theorems for hardness of approximation in P. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 25–36. IEEE, 2017.
4. Noga Alon, Dana Moshkovitz, and Shmuel Safra. Algorithmic construction of sets for k-restrictions. *ACM Transactions on Algorithms (TALG)*, 2(2):153–177, 2006.
5. Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM (JACM)*, 45(3):501–555, 1998.

6 E. Bonnet, B. Escoffier, E. Kim, and V. Th. Paschos. On Subexponential and FPT-Time Inapproximability. In *Parameterized and Exact Computation - 8th International Symposium, IPEC 2013, Sophia Antipolis, France, September 4-6, 2013, Revised Selected Papers*, pages 54–65, 2013.

7 Nicolas Bourgeois, Bruno Escoffier, and Vangelis Paschos. Efficient approximation of MIN SET COVER by “low-complexity" exponential algorithms, 2008.

8 Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. From gap-ETH to FPT-inapproximability: Clique, dominating set, and more. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 743–754. IEEE, 2017.

9 Yijia Chen and Bingkai Lin. The constant inapproximability of the parameterized dominating set problem. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 505–514. IEEE, 2016.

10 V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations*, 4(3):233–235, 1979.

11 Marek Cygan, Fedor V. Fomin, Danny Hermelin, and Magnus Wahlström. Randomization in Parameterized Complexity (Dagstuhl Seminar 17041). *Dagstuhl Reports*, 7(1):103–128, 2017.

12 Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 4(8). Springer, 2015.

13 Marek Cygan, Łukasz Kowalik, and Mateusz Wykurz. Exponential-time approximation of weighted set cover. *Information Processing Letters*, 109(16):957–961, 2009.

14 I. Dinur and D. Steurer. Analytical approach to parallel repetition. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 624–633, 2014.

15 R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.

16 Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1-3):57–67, 2004.

17 Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

18 J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

19 Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *Journal of Computer and System Sciences*, 62:367–375, 2001.

20 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

21 D. S. Johnson. Approximation Algorithms for Combinatorial Problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974.

22 Stasys Jukna. *Extremal combinatorics: with applications in computer science*. Springer Science & Business Media, 2011.

23 R. M. Karp. Reducibility Among Combinatorial Problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York.*, pages 85–103, 1972.

24 CS Karthik, Bundit Laekhanukit, and Pasin Manurangsi. On the Parameterized Complexity of Approximating Dominating Set. In *STOC*, 2018.

25 Wenxing Lai. The Inapproximability of k-DominatingSet for Parameterized AC^0 Circuits. In *Frontiers in Algorithmics - 13th International Workshop, FAW 2019, Sanya, China, April 29 - May 3, 2019, Proceedings*, pages 133–143, 2019.

26 Bingkai Lin. The Parameterized Complexity of k-Biclique. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 605–615, 2015.

27 Bingkai Lin. The Parameterized Complexity of the k-Biclique Problem. *J. ACM*, 65(5):34:1–34:23, 2018.

28 L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13(4):383–390, 1975.

29 C. Lund and M. Yannakakis. On the Hardness of Approximating Minimization Problems. *Journal of the ACM*, 41(5):960–981, 1994.

30 D. Marx. Parameterized Complexity and Approximation Algorithms. *The Computer Journal*, 51(1):60–78, 2008.

31 Mihai Pătraşcu and Ryan Williams. On the possibility of faster SAT algorithms. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1065–1075. SIAM, 2010.

32 R. Raz and S. Safra. A Sub-Constant Error-Probability Low-Degree Test, and a Sub-Constant Error-Probability PCP Characterization of NP. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing, STOC 1997, El Paso, Texas, USA, May 4-6, 1997*, pages 475–484, 1997.

33 Benjamin Rossman. On the constant-depth complexity of k-clique. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 721–730. ACM, 2008.

34 P. Slavík. A Tight Analysis of the Greedy Algorithm for Set Cover. *Journal of Algorithms*, 25(2):237–254, 1997.

35 S. K. Stein. Two Combinatorial Covering Theorems. *Journal of Combinatorial Theory, Series A*, 16(3):391–397, 1974.

Maintaining Perfect Matchings at Low Cost

Jannik Matuschke
Research Center for Operations Management, KU Leuven, Leuven, Belgium
jannik.matuschke@kuleuven.be

Ulrike Schmidt-Kraepelin
Institute of Software Engineering and Theoretical Computer Science, TU Berlin, Berlin, Germany
u.schmidt-kraepelin@tu-berlin.de

José Verschae
Institute of Engineering Sciences, Universidad de O'Higgins, Rancagua, Chile
jose.verschae@uoh.cl

Abstract

The min-cost matching problem suffers from being very sensitive to small changes of the input. Even in a simple setting, e.g., when the costs come from the metric on the line, adding two nodes to the input might change the optimal solution completely. On the other hand, one expects that small changes in the input should incur only small changes on the constructed solutions, measured as the number of modified edges. We introduce a two-stage model where we study the trade-off between quality and robustness of solutions. In the first stage we are given a set of nodes in a metric space and we must compute a perfect matching. In the second stage $2k$ new nodes appear and we must adapt the solution to a perfect matching for the new instance.

We say that an algorithm is (α, β)-robust if the solutions constructed in both stages are α-approximate with respect to min-cost perfect matchings, and if the number of edges deleted from the first stage matching is at most βk. Hence, α measures the quality of the algorithm and β its robustness. In this setting we aim to balance both measures by deriving algorithms for constant α and β. We show that there exists an algorithm that is $(3, 1)$-robust for any metric if one knows the number $2k$ of arriving nodes in advance. For the case that k is unknown the situation is significantly more involved. We study this setting under the metric on the line and devise a $(10, 2)$-robust algorithm that constructs a solution with a recursive structure that carefully balances cost and redundancy.

2012 ACM Subject Classification Mathematics of computing → Combinatorial algorithms

Keywords and phrases matchings, robust optimization, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.82

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1811.10580`.

Funding The authors were supported by Project Fondecyt Nr. 1181527, the Alexander von Humboldt Foundation with funds of the German Federal Ministry of Education and Research (BMBF), BAYLAT, and the Deutsche Forschungsgemeinschaft under grant BR 4744/2-1.

Acknowledgements We thank anonymous reviewers for their helpful feedback.

1 Introduction

Weighted matching is one of the founding problems in combinatorial optimization, playing an important role in the settling of the area. The work by Edmonds [8] on this problem greatly influenced the role of polyhedral theory on algorithm design [23]. On the other hand, the problem found applications in several domains [1, 2, 6, 22, 26]. In particular

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 82; pp. 82:1–82:14

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

Figure 1 Example instance on the line. Vertices in Stage 1 are depicted by light grey dots and second stage arrivals are indicated as dark grey crosses. First and second state optimum are depicted by solid and dotted edges, respectively. The arrival of two new vertices leads to a significant drop in the cost of the optimum.

routing problems are an important area of application, and its procedures often appeared as subroutines of other important algorithms, the most notable being Christofides' algorithm [5] for the *traveling salesperson problem (TSP)*.

An important aspect of devising solution methods for optimization problems is studying the sensitivity of the solution towards small changes in the input. This sensitivity analysis has a long history and plays an important role in practice [10]. Min-cost matching is a problem that has particularly sensitive optimal solutions. Assume for example that nodes lie on the real line at points ℓ and $\ell + 1 - \varepsilon$ for some $0 < \varepsilon < 1$ and all $\ell \in \{1, \ldots, n\}$, see Figure 1. The min-cost matching, for costs equal the distance on the line, is simply the edges $\{\ell, \ell + 1 - \varepsilon\}$. However, even under a minor modification of the input, e.g., if two new nodes appear at points $1 - \varepsilon$ and $n + 1$, the optimal solution changes all of its edges, and furthermore the cost decreases by a $\Theta(1/\varepsilon)$ factor. Rearranging many edges in an existing solution is often undesirable and may incur large costs, for example in an application context where the matching edges imply physical connections or binding commitments between nodes. A natural question in this context is whether we can avoid such a large number of rearrangements by constructing a *robust* solution that is only slightly more expensive. In other words, we are interested in studying the trade-off between robustness and the cost of solutions.

We consider a two-stage robust model with recourse. Assume we are given an underlying metric space $(\mathcal{X}, c)$. The input for the *first stage* is a complete graph G_1 whose node set $V(G_1)$ is a finite, even subset of $\mathcal{X}$. The cost of an edge $\{v, w\}$ is given by the corresponding cost $c(v, w)$ in the metric space[1]. In a second stage we get an extended complete graph G_2 containing all nodes in $V(G_1)$ plus $2k$ additional nodes. As before, costs of edges in G_2 are given by the underlying metric. In the first stage we must create a perfect matching M_1 for G_1. In the second stage, after G_2 is revealed, we must adapt our solution by constructing a new perfect matching M_2 for G_2, called the *second stage reply*. We say that a solution M_1 is two-stage (α, β)-robust if for any instantiation of the second stage there exists a solution M_2 such that two conditions hold. First, the total cost of edges in M_i must satisfy $c(M_i) \leq \alpha \cdot c(O_i)$ for $i \in \{1, 2\}$, where O_i denotes a min-cost perfect matching in G_i. Second, it must hold that $|M_1 \setminus M_2| \leq \beta k$. An algorithm is two-stage (α, β)-robust if, given G_1 and c, it returns a two-stage (α, β)-robust matching and, given the set of new arrivals, a corresponding second stage reply. We refer to α as the *competitive factor* and β as the *recourse factor* of the algorithm. Our main goal is to balance cost and recourse, and thus we aim to obtain algorithms where α and β are constants.

Our model is closely related to an online model with recourse. Consider a graph whose nodes are revealed online two by two. Our objective is to maintain a perfect matching at all times. As above, irrevocable decisions do not allow for constant competitive factors. This suggests a model where in each iteration we are allowed to modify a constant number of edges.

[1] Graphs with arbitrary cost functions do not allow for $(O(1), O(1))$-robust matchings in general, e.g., consider a variant of the example in Figure 1 in which all omitted edges have infinite cost.

An online algorithm that maintains an α-approximation at all time while deleting at most β edges per iteration can be easily transformed into a two-stage (α, β)-robust algorithm. Given an instance of the two-stage model, we choose an arbitrary order for the nodes available in the first stage and create M_1 by following the updates proposed by an online algorithm. Then, we repeat the procedure for the arrivals in Stage 2. Thus, our two-stage model is also the first necessary step for understanding this more involved online model. Megow et al. [20] study a similar online model for minimum spanning trees and the TSP in metric graphs. After giving a $(1+\varepsilon)$-competitive algorithm with recourse factor $\frac{1}{\varepsilon}\log(\frac{1}{\varepsilon})$ for the former problem, they are able to derive a $(2+\varepsilon)$-competitive algorithm with constant recourse factor for the latter problem by combining their results with a modified version of the well-known double-tree algorithm. An algorithm for the online variant of our proposed model together with the aforementioned results, would give rise to an online variant of Christofide's algorithm which can yield an improved competitiveness factor for the considered online TSP.

Our Results and Techniques. We distinguish two variants of the model. In the *k-known* case we assume that in Stage 1 we already know the number of new nodes $2k$ that will arrive in Stage 2. For this case we present a simple two-stage $(3,1)$-robust algorithm.

▶ **Theorem 1.** *Let $(\mathcal{X}, c)$ be a metric space, $V_1 \subseteq \mathcal{X}$ with $|V_1|$ even, and G_1 be the complete graph on V_1. For $k \in \mathbb{N}$ known in advance, there is a perfect matching M_1 in G_1 that is two-stage $(3,1)$-robust for $2k$ arrivals. Such a matching and corresponding second stage reply can be computed in time poly$(|V_1|, k)$.*

The example in Figure 1 illustrates a worst case scenario for the strategy of choosing O_1 as the first stage matching for $k = 1$. The reason for this is that the nodes arriving in Stage 2 induce a path in $O_1 \Delta O_2$ that incurs a significant drop in the optimal cost. Our algorithm is designed towards preparing for such bad scenarios. To this end, we define the notion of *gain* for a path P with respect to a matching X as follows:

$$\text{gain}_X(P) := c(P \cap X) - c(P \setminus X).$$

In Stage 1, our algorithm chooses k edge-disjoint O_1-alternating paths of maximum total gain with respect to O_1. For each such path P we modify O_1 by removing $P \cap O_1$ and adding $(P \setminus O_1) \cup \{e(P)\}$, where $e(P)$ is the edge that connects the endpoints of P. Our choice of paths of maximum gain implies that $P \cap O_1$ is larger than $P \setminus O_1$. Therefore we can bound the cost of the solution in the first stage against that of O_1 and also infer that most of its costs is concentrated on the edges $e(P)$. For the second stage we construct a solution for the new instance by removing the k edges of the form $e(P)$ and adding new edges on top of the remaining solution. The algorithm is described in detail in Section 2.

For the case where k is unknown the situation is considerably more involved as a first stage solution must work for any number of arriving nodes simultaneously. In this setting we restrict our study to the real line and give an algorithm that is two-stage $(10,2)$-robust.

▶ **Theorem 2.** *Let $\mathcal{X} = \mathbb{R}$ and $c = |\cdot|$, $V_1 \subseteq \mathcal{X}$ with $|V_1|$ even, and let G_1 be the complete graph on V_1. Then there is a perfect matching M_1 in G_1 that is two-stage $(10,2)$-robust. Such a matching, as well as the second stage reply, can be computed in time poly$(|V_1|, k)$.*

The first stage solution M is constructed iteratively, starting from the optimal solution. We will choose a path P greedily such that it maximizes $\text{gain}_M(P)$ among all alternating paths that are *heavy*, i.e., the cost of $P \cap M$ is a factor 2 more expensive than the cost of $P \setminus M$. Then M is modified by augmenting along P and adding edge $e(P)$, which we fix to

be in the final solution. We iterate until M only consists of fixed edges. As we are on the line, each path P corresponds to an interval and we can show that the constructed solution form a laminar family. Furthermore, our choice of heavy paths implies that their lengths satisfy an exponential decay property. This allows us to bound cost of the first stage solution. For the second stage, we observe that the symmetric difference $O_1 \Delta O_2$ induces a set of intervals on the line. For each such an interval, we remove on average at most two edges from the first stage matching and repair the solution with an optimal matching for the exposed vertices. A careful choice of the removed edges, together with the greedy construction of the first stage solution, enables us to bound the cost of the resulting second stage solution within a constant factor of the optimum. See Sections 3 and 4 for a detailed description of this case.

Related Work. Intense research has been done on several variants of the *online bipartite matching* problem [17, 16, 18, 4, 21]. In this setting we are given a known set of *servers* while a set of *clients* arrive online. In the *online bipartite metric matching problem* servers and clients correspond to points from a metric space. Upon arrival, each client must be matched to a server irrevocably, at cost equal to their distance. For general metric spaces, there is a tight bound of $(2n-1)$ on the competitiveness factor of deterministic online algorithms, where n is the number of servers [18, 16]. Recently, Raghvendra presented a deterministic algorithm [24] with the same competitiveness factor, that in addition is $O(\log(n))$-competitive in the random arrival model. Also, its analysis can be parameterized for any metric space depending on the length of a TSP tour and its diameter [21]. For the special case of the metric on the line, Raghvendra [25] recently refined the analysis of the competitive ratio to $O(\log(n))$. This gives a deterministic algorithm that matches the previously best known bound by Gupta and Lewi [15], which was attained by a randomized algorithm. As the lower bound of 9.001 [9] could not be improved for 20 years, the question whether there exists a constant competitive algorithm for the line remains open.

The *online matching with recourse problem* considers an unweighted bipartite graph. Upon arrival, a client has to be matched to a server and can be reallocated later. The task is to minimize the number of reallocations under the condition that a maximum matching is always maintained. The problem was introduced by Grove, Kao and Krishnan [11]. Chaudhuri et al. [4] showed that for the random arrival model a simple greedy algorithm uses $O(n \log(n))$ reallocations with high probability and proved that this analysis is tight. Recently, Bernstein, Holm and Rotenberg [3] showed that the greedy algorithm needs $O(n \log^2 n)$ allocations in the adversarial model, leaving a small gap to the lower bound of $O(n \log n)$. Gupta, Kumar and Stein [14] consider a related problem where servers can be matched to more than one client, aiming to minimize the maximum number of clients that are assigned to a server. They achieve a constant competitive factor server while doing in total $O(n)$ reassignments.

Online min-cost problems with reassignments have been studied in other contexts. For example in the *online Steiner tree problem with recourse* a set of points on a metric space arrive online. We must maintain Steiner trees of low cost by performing at most a constant (amortized) number of edge changes per iteration. While the pure online setting with no reassignment only allows for $\Omega(\log(n))$ competitive factors, just one edge deletion per iteration is enough to obtain a constant competitive algorithm [12]; see also [13, 20].

The concept of *recoverable robustness* is also related to our setting [19]. In this context the perfect matching problem on unweighted graphs was considered by Dourado et. al. [7]. They seek to find perfect matchings which, after the failure of some edges, can be recovered to a perfect matching by making only a small number of modifications. They establish computational hardness results for the question whether a given graph admits a robust recoverable perfect matching.

2 Known Number of Arrivals

In this section, we consider the setting where k (number of arrival pairs in Stage 2) is already known in Stage 1. Let G_1 be the complete graph given in Stage 1 (with edge costs c induced by an arbitrary metric) and let O_1 be a min-cost perfect matching in G_1. Without loss of generality assume that $|O_1| > 2k$, as otherwise, we can remove all edges of M_1 in Stage 2.

Algorithm 1.1 works as follows:

(i) Let $P_1, \dots, P_k$ be edge-disjoint, O_1-alternating paths maximizing $\sum_{i=1}^{k} \text{gain}_{O_1}(P_i)$.
(ii) Set $\overline{M} := O_1 \Delta P_1 \Delta \dots \Delta P_k$.
(iii) Return $M_1 := \overline{M} \cup \{e(P_i) : i \in [k]\}$.

It is easy to see that each path P_i starts and ends with an edge from O_1 and $\text{gain}_{O_1}(P_i) \geq 0$. As a consequence, M_1 is a perfect matching and

$$c(\overline{M}) = c(O_1) - \sum_{i=1}^{k} \text{gain}_{O_1}(P_i) \leq c(O_1).$$

Using $c(e(P_i)) \leq c(P_i)$ and $\bigcup_{i=1}^{k} P_i = O_1 \Delta \overline{M} \subseteq O_1 \cup \overline{M}$ we obtain

$$c(M_1) \leq c(\overline{M}) + \sum_{i=1}^{k} c(P_i) \leq c(\overline{M}) + c(\overline{M}) + c(O_1) \leq 3 \cdot c(O_1).$$

Now consider the arrival of $2k$ new vertices, resulting in the graph G_2 with min-cost matching O_2. Note that $O_2 \Delta \overline{M}$ is a U-join, where U is the union of the endpoints of the paths $P_1, \dots, P_k$ and the $2k$ newly arrived vertices.

Algorithm 1.2 works as follows:

(i) Let $P'_1, \dots, P'_{2k}$ be the $2k$ maximal paths from $O_2 \Delta \overline{M}$.
(ii) Return $M_2 := \overline{M} \cup \{e(P'_i) : i \in [2k]\}$.

Note that $O_1 \Delta O_2$ consists of k alternating paths $R_1, \dots, R_k$, from which we remove the starting and ending O_2-edge. Then these paths would have been a feasible choice for $P_1, \dots, P_k$, implying that the total gain of the R_i's is at most that of the P_i's. We conclude that

$$c(\overline{M}) = c(O_1) - \sum_{i=1}^{k} \text{gain}_{O_1}(P_i) \leq c(O_1) - \sum_{i=1}^{k} \text{gain}_{O_1}(R_i) \leq c(O_2).$$

Applying $\bigcup_{i=1}^{2k} P'_i \subseteq O_2 \Delta \overline{M} \subseteq O_2 \cup \overline{M}$, we obtain

$$c(M_2) \leq c(\overline{M}) + \sum_{i=1}^{2k} c(P'_i) \leq c(\overline{M}) + c(\overline{M}) + c(O_2) \leq 3 \cdot c(O_2).$$

As $|M_1 \setminus M_2| \leq |M_1 \setminus \overline{M}| = k$, we conclude that M_1 is indeed two-stage $(3, 1)$-robust.

We remark that $\overline{M}$ is always a min-cost matching of cardinality $|V(G_1)| - 2k$ in G_1. Thus, alternatively to Algorithm 1.1, we can compute a min-cost matching of cardinality $|V(G_1)| - 2k$ and match uncovered nodes at minimum cost. Finding $\overline{M}$ directly as well as finding gain-maximizing paths as described in Algorithm 1.1 can be done efficiently by solving a min-cost T-join problem in an extension of G_1. This concludes our proof of Theorem 1.

3 Unknown Number of Arrivals – Stage 1

In this section, we consider the case that the underlying metric corresponds to the real line. This implies that there is a Hamiltonian path L in G_1 such that $c(v, w) = c(L[v, w])$ for all $v, w \in V(G_1)$, where $L[v, w]$ is the subpath of L between nodes v and w. We will refer to L as *the line* and call the subpaths of L *intervals*. The restriction to the metric on the line results in a uniquely defined min-cost perfect matching O_1 with a special structure. All proofs omitted due to space constraints can be found in the appendix of the full version.

Figure 2 For fixed $\alpha, \beta \in O(1)$, we construct an instance for which Algorithm 1.1 is not (α, β)-robust for any $k \in O(1)$. G_1 is constructed such that O_1 contains $\beta + 1$ edges of size c_1 and $\beta^3 + \beta^2$ of size c_3 that are equally distributed between c_1-edges. The distance between any two consecutive edges in O_1 is c_2. Values c_1, c_2, c_3 are chosen depending on α, guaranteeing $c_1 \gg \beta^3 c_2 \gg \beta^6 c_3$. Algorithm 1.1 with $k \geq \beta + 1$ chooses $M_1 = O_1$. Now, assume there are two arrivals at the extremes of the line: while the optimal costs in the second stage decrease heavily to $c(O_2) \in \Theta(\beta^3 c_2)$, only β deletions are allowed within M_1. As a result, there does not exist a feasible second stage reply. Now, consider Algorithm 1.1 with $k \leq \beta$. Then, M_1 contains more than $\beta^2 + \beta$ edges of size c_2. If in the second stage $2(\beta + 1)$ nodes arrive right next to the endpoints of c_1-edges, the optimal costs drop to $\Theta(\beta^3 c_3)$ while only $\beta^2 + \beta$ deletions are allowed. Again, no feasible second stage reply exists.

▶ **Lemma 3.** *O_1 is the unique perfect matching contained in L.*

When the number of arrivals is not known in the first stage, the approach for constructing the first stage matching introduced in Section 2 does not suffice anymore. Figure 2 illustrates a class of instances for which Algorithm 1.1 cannot achieve $(O(1), O(1))$-robustness, no matter how we choose k. For a matching M, define $g(M) := \max_{e \in L} |\{\{v, w\} \in M : e \in L[v, w]\}|$. Informally speaking, $g(M)$ captures the maximal number of times a part of the line is traversed by edges in M. The example in Figure 2 can be generalized to show that we cannot restrict ourselves to constructing matchings M_1 such that $g(M_1)$ is bounded by a constant.

In view of the example in Figure 2, we adapt the approach from Section 2 as follows. Instead of creating a fixed number of paths, our algorithm now iteratively and greedily selects a path P of maximum gain with respect to a dynamically changing matching X (initially $X = O_1$). In order to bound the total cost incurred by adding edges of the form $e(P)$, we only consider paths P for which $X \cap P$ contributes a significant part to the total cost of P.

▶ **Definition 4.** *Let $X, P \subseteq E(G_1)$.*

1. *We say that P is X-heavy if $c(P \cap X) \geq 2 \cdot c(P \setminus X)$.*
2. *We say that P is X-light if $c(P \cap X) \leq \frac{1}{2} \cdot c(P \setminus X)$.*

Algorithm 2.1 works as follows:

(i) Set $M_1 := \emptyset$ and $X := O_1$.

(ii) While $X \neq \emptyset$: Let P be an X-heavy X-alternating path maximizing $\text{gain}_X(P)$ and update $M_1 \leftarrow M_1 \cup \{e(P)\}$ and $X \leftarrow X \Delta P$.

(iii) Return M_1.

Note that in each iteration, the path P starts and ends with an edge from X as it is gain-maximizing (if P ended with an edge that is not in X, we could simply remove that edge and obtain a path of higher gain). Therefore it is easy to see that $X \cup M_1$ is always a perfect matching, and in each iteration the cardinality of X decreases by 1.

Now number the iterations of the while loop in Algorithm 2.1 from 1 to n. Let $X^{(i)}$ be the state of X at the beginning of iteration i. Let $P^{(i)}$ be the path chosen in iteration i and let $e^{(i)} = e(P^{(i)})$ be the corresponding edge added to M. The central result in this section is Lemma 7, in which we show that the paths $P^{(i)}$ form a laminar family of intervals on the line.

Within the proof we will make use of observations stated in Lemmas 5 and 6. For convenience, we define the projection $\psi(e) := L[v, w]$ that maps an edge $e = \{v, w\} \in E(G_1)$ to the corresponding subpath $L[v, w]$.

Figure 3 A minimal example of a situation in which Lemma 7 would be violated. There are two iterations $i < j$ with paths $P^{(i)}$ (depicted in blue) and $P^{(j)}$ (depicted in red) such that $X \cap P^{(j)}$ was not modified between iteration i and iteration j. Then, extending the blue path $P^{(i)}$ with the rightmost edge yields an $X^{(i)}$-heavy path with higher gain than $P^{(i)}$, a contradiction.

▶ **Lemma 5.** *Let $X \subseteq E(G_1)$.*
1. *Let $A, B \subseteq E(G_1)$ be two X-heavy (X-light, respectively) sets with $A \cap B = \emptyset$. Then $A \cup B$ is X-heavy (X-light, respectively).*
2. *Let $A, B \subseteq E(G_1)$ with $B \subseteq A$. If A is X-heavy and $\operatorname{gain}_X(B) < 0$, then $A \setminus B$ is X-heavy. If A is X-light and $\operatorname{gain}_X(B) > 0$, then $A \setminus B$ is X-light.*

▶ **Lemma 6.** *Let $X \subseteq L$ be a matching.*
1. *Let P be an X-heavy X-alternating path maximizing $\operatorname{gain}_X(P)$. If X covers all vertices in $V(\psi(P))$, then $P = \psi(P)$.*
2. *Let $I \subseteq L$ be an interval. Then there is an X-alternating path P such that $c(P \cap X) = c(X \cap I)$ and $c(P \setminus X) = c(I \setminus X)$.*

▶ **Lemma 7.**
1. *$X^{(i)}, P^{(i)} \subseteq L$ for all $i \in [n]$.*
2. *For all $i, j \in [n]$ with $i < j$, either $P^{(i)} \cap P^{(j)} = \emptyset$ or $P^{(j)} \subset P^{(i)}$.*

Proof. We say a pair (i, j) with $i < j$ is *violating* if $\psi(P^{(i)}) \cap \psi(P^{(j)}) \neq \emptyset$ and $\psi(P^{(j)}) \setminus \psi(P^{(i)}) \neq \emptyset$. We will show that no violating pair exists. This proves the lemma as the following claim asserts.

▷ Claim. If $P^{(j)} \neq \psi(P^{(j)})$, then there is a violating pair (i', j') with $i' < j' \leq j$.

Proof. Let j' be minimal with $P^{(j')} \neq \psi(P^{(j')})$. Note that minimality of j' implies that $X^{(j')} = O_1 \Delta P^{(1)} \Delta \ldots \Delta P^{(j'-1)} \subseteq L$. Then Lemma 6 implies that there must be a vertex $v \in V(\psi(P^{(j')}))$ not covered by $X^{(j')}$. Because $X^{(j')}$ covers exactly those vertices not covered by $\{e^{(i')} : i' < j'\}$, there must be an $i' < j'$ such that v is an endpoint of $P^{(i')}$. The vertex v cannot be an endpoint of $P^{(j')}$, because v is exposed in $X^{(j')}$ and $P^{(j')}$ starts and ends with edges from $X^{(j')}$. This implies that (i', j') is a violating pair. ◁

Now let us assume there are no violating pairs. Then $P^{(i)} = \psi(P^{(i)}) \subseteq L$ for all $i \in [n]$ by the claim, which also implies $X^{(i)} \subseteq L$. This implies the lemma as, in this situation, the condition for violating pairs coincides with the condition in point 2 of the lemma.

By contradiction assume there is a violating pair. Choose j such that j is minimal among all possible choices of violating pairs. Then choose i such that it is maximal for that j among all violating pairs.

Note that the claim implies that $P^{(i)}, X^{(i)}, X^{(j)} \subseteq L$. Furthermore, our choice of i and j implies that $\psi(P^{(j')}) \cap \psi(P^{(j)}) = \emptyset$ for all j' with $i < j' < j$, as otherwise (i, j') or (j', j) would be a violating pair. In particular, $P^{(j')} \cap P^{(j)} = \emptyset$ for all j' with $i < j' < j$ and thus

$$X^{(j)} \cap P^{(j)} = X^{(i+1)} \cap P^{(j)} = (X^{(i)} \Delta P^{(i)}) \cap P^{(j)}. \tag{1}$$

Now consider $I_1 := \psi(P^{(j)}) \cap P^{(i)}$ and $I_2 := \psi(P^{(j)}) \setminus P^{(i)}$, both of which are non-empty since (i, j) is a violating pair. Then (1) implies $X^{(j)} \cap I_1 = I_1 \setminus X^{(i)}$ and $I_1 \setminus X^{(j)} = X^{(i)} \cap I_1$. We conclude that $\operatorname{gain}_{X^{(j)}}(I_1) = -\operatorname{gain}_{X^{(i)}}(I_1) \leq 0$, as I_1 is a prefix of the gain-maximizing path $P^{(i)}$ (see the appendix of the full version for a formal proof).

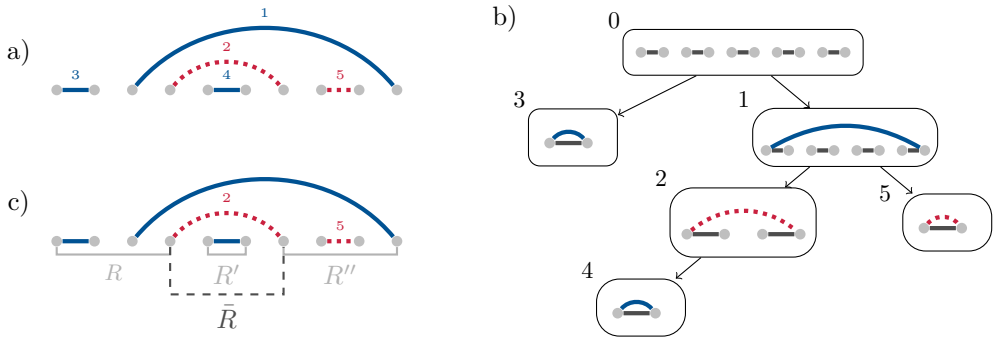


Figure 4 a) Illustration of the matching created by an example execution of Algorithm 2.1. Edges added to M_1 in an iteration from W_{H} are depicted by blue solid lines and edges created in an iteration from W_{L} are illustrated by red dotted lines. b) Illustration of the corresponding tree. For every tree-node $i \in W$, grey edges indicate $X^{(i)}$ and an arc illustrates the edge connecting the end nodes of $P^{(i)}$. c) Illustration of example assignment of requests to iterations (defined in Section 4). Requests $R, R', R'' \in \mathcal{R}$ are assigned such that $R, R'' \in \mathcal{R}(0)$ and $R' \in \mathcal{R}(2)$. $\bar{R} \in \bar{\mathcal{R}}(0)$ is a gap between two requests associated with tree-node 0.

Therefore $I_2 = \psi(P^{(j)}) \setminus I_1$ is $X^{(j)}$-heavy by Lemma 5. But then I_2 is also $X^{(i)}$-heavy because (1) implies $X^{(j)} \cap I_2 = X^{(i)} \cap I_2$. Hence $I' := P^{(i)} \cup I_2$ is $X^{(i)}$-heavy by Lemma 5 and further

$$\mathrm{gain}_{X^{(i)}}(I') = \mathrm{gain}_{X^{(i)}}(P^{(i)}) + \mathrm{gain}_{X^{(j)}}(I_2) > \mathrm{gain}_{X^{(i)}}(P^{(i)}),$$

because $\mathrm{gain}_{X^{(j)}}(I_2) \geq \frac{1}{3}c(I_2)$. By Lemma 6 there is an $X^{(i)}$-heavy, $X^{(i)}$-alternating path with higher gain than $P^{(i)}$, a contradiction. ◀

Tree structure. Lemma 7 induces a tree structure on the paths selected by Algorithm 2.1. We define the directed tree $T = (W, A)$ as follows. We let $W := \{0, \dots n\}$ and define $P^{(0)} := L$. For $i, j \in W$ we add the arc (i, j) to A if $P^{(j)} \subset P^{(i)}$ and there is no $i' \in W$ with $P^{(j)} \subset P^{(i')} \subset P^{(i)}$. It is easy to see that T is an out-tree with root 0. We let $T[i]$ be the unique 0-i-path in T. We define the set of children of $i \in W$ by $\mathrm{ch}(i) := \{j \in W : (i, j) \in A\}$. Furthermore, let $W_{\mathrm{H}} := \{i \in W : |T[i]| \text{ is odd}\}$ and $W_{\mathrm{L}} := \{i \in W : |T[i]| \text{ is even}\}$ be the set of *heavy* and *light* nodes in the tree, respectively. These names are justified by the following lemma. See Figure 4 a)-b) for an illustration.

▶ **Lemma 8.** *If $i \in W_H$, then $P^{(i)} \cap X^{(i)} = P^{(i)} \cap O_1$ and, in particular, $P^{(i)}$ is O_1-heavy. If $i \in W_L \setminus \{0\}$, then $P^{(i)} \cap X^{(i)} = P^{(i)} \setminus O_1$ and, in particular, $P^{(i)}$ is O_1-light.*

Proof. Let $i \in W \setminus \{0\}$. From Lemma 7 we know that for every iteration $i' \in W \setminus \{0\}$ with $i' < i$ it holds that either $P^{(i)} \cap P^{(i')} = \emptyset$ or $P^{(i)} \subset P^{(i')}$. In the first case it holds that $P^{(i)} \cap X^{(i'+1)} = P^{(i)} \cap X^{(i')}$, in the latter case it holds that $P^{(i)} \cap X^{(i'+1)} = P^{(i)} \cap (P^{(i)} \Delta X^{(i')})$. Moreover, it is easy to see that $i' < i$ and $P^{(i)} \subset P^{(i')}$ holds if and only if $i' \in V(T[i]) \setminus \{0, i\}$. If $i \in W_{\mathrm{H}}$, this implies that there exist an even number of iterations $i' < i$ for which $P^{(i)} \subset P^{(i')}$ holds. Hence, we obtain

$$P^{(i)} \cap X^{(i)} = P^{(i)} \cap (\underbrace{P^{(i)} \Delta \dots \Delta P^{(i)}}_{\text{evenly often}} \Delta\, O_1) = P^{(i)} \cap O_1.$$

If $i \in W_{\mathrm{L}}$, this implies that there exist an odd number of iterations $i' < i$ for which $P^{(i)} \subset P^{(i')}$ holds. Hence, we can deduce that

$$P^{(i)} \cap X^{(i)} = P^{(i)} \cap (\underbrace{P^{(i)} \Delta \dots \Delta P^{(i)}}_{\text{oddly often}} \Delta\, O_1) = P^{(i)} \setminus O_1.$$ ◀

The fact that nested paths are alternatingly O_1-heavy and O_1-light implies an exponential decay property. As a consequence we can bound the cost of M_1.

▶ **Lemma 9.** *Let $i \in W \setminus \{0\}$. Then $\sum_{j \in \mathrm{ch}(i)} c(P^{(j)}) \leq \frac{1}{2} \cdot c(P^{(i)})$.*

Proof. Let $i \in W \setminus \{0\}$. Then

$$\sum_{j \in \mathrm{ch}(i)} c(P^{(j)}) \leq \tfrac{3}{2} \sum_{j \in \mathrm{ch}(i)} c(P^{(j)} \cap X^{(j)}) \leq \tfrac{3}{2} c(P^{(i)} \setminus X^{(i)}) \leq \tfrac{1}{2} c(P^{(i)}),$$

where the first inequality follows from the fact that $P^{(j)}$ is $X^{(j)}$-heavy; the second inequality follows from the fact that $P^{(j)} \cap X^{(j)} \subseteq P^{(i)} \setminus X^{(i)}$ for $j \in \mathrm{ch}(i)$ and the fact that the intervals $P^{(j)}$ for all children are disjoint; the last inequality follows from the fact that $P^{(i)}$ is $X^{(i)}$-heavy. ◀

▶ **Lemma 10.** *$c(M_1) \leq 3c(O_1)$.*

Proof. Note that $c(M_1) = \sum_{i=1}^{n} c(e^{(i)}) = \sum_{i \in W \setminus \{0\}} c(P^{(i)})$. For $\ell \in \mathbb{N}$, let

$$W_\ell := \{i \in W : |T[i]| = \ell\}.$$

Observe that Lemma 9 implies that $\sum_{i \in W_\ell} c(P^{(i)}) \leq \left(\frac{1}{2}\right)^{\ell-1} \sum_{i \in W_1} c(P^{(i)})$ for all $\ell \in \mathbb{N}$. Furthermore $\sum_{i \in W_1} c(P^{(i)}) \leq \frac{3}{2} c(O_1)$, because $W_1 \subseteq W_{\mathrm{H}}$. Hence

$$c(M_1) = \sum_{\ell=1}^{\infty} \sum_{i \in W_\ell} c(P^{(i)}) \leq \sum_{\ell=1}^{\infty} \left(\frac{1}{2}\right)^{\ell-1} \sum_{i \in W_1} c(P^{(i)}) = 2 \cdot \frac{3}{2} c(O_1).$$ ◀

4 Unknown Number of Arrivals – Stage 2

We now discuss how to react to the arrival of $2k$ additional vertices. We let O_2 be the min-cost perfect matching in the resulting graph G_2 and define

$$\mathcal{R} := \{P : P \text{ is a maximal path in } (O_1 \Delta O_2) \cap L\}.$$

We call the elements of $\mathcal{R}$ *requests*. An important consequence of our restriction to the metric space on the line is that $|\mathcal{R}| \leq k$ (in fact, each of the k maximal paths of $O_1 \Delta O_2$ is contained in L after removing its first and last edge).

▶ **Lemma 11.** *$|\mathcal{R}| \leq k$ and each $R \in \mathcal{R}$ starts and ends with an edge of O_1.*

For simplification of the analysis we make the following assumptions on the structure of the request set.

▶ **Assumption A.** *For all $i \in W_L$ and all $R \in \mathcal{R}$, either $P^{(i)} \cap R = \emptyset$ or $P^{(i)} \subseteq R$, or $R \subseteq P^{(i)}$.*

▶ **Assumption B.** *For all $j \in W_H$, if $\bigcup_{R \in \mathcal{R}} R \cap P^{(j)} \neq \emptyset$, then the first and last edge of $P^{(j)}$ are in $\bigcup_{R \in \mathcal{R}} R$.*

In the appendix of the full version we prove formally that these are without loss of generality. For intuition, we give a short sketch of the proof. Assume we are confronted with a set of requests $\mathcal{R}$ that violates at least one of the assumptions. Based on $\mathcal{R}$, we can construct a modified set of requests $\mathcal{R}'$ complying with the assumptions and fulfilling two properties: First, $\mathrm{gain}_{O_1}(\bigcup_{R \in \mathcal{R}'} R) > \mathrm{gain}_{O_1}(\bigcup_{R \in \mathcal{R}} R)$, i.e., R' induces smaller second

stage optimal costs than $\mathcal{R}$ and secondly, $|\mathcal{R}'| \leq |\mathcal{R}|$, i.e., $\mathcal{R}'$ allows for at most as many modifications as $\mathcal{R}$ does. As a consequence, if we run our proposed second stage algorithm for $\mathcal{R}'$ and construct M_2 accordingly, the analysis of the approximation and recourse factor carry over from the analysis of $\mathcal{R}'$ to the actual set of requests $\mathcal{R}$. Hence, we can assume w.l.o.g. that $\mathcal{R}$ fulfills the assumptions.

From the set of requests $\mathcal{R}$, we will determine a subset of at most $2k$ edges that we delete from M_1. To this end, we assign each request to a light node in W_{L} as follows. For $R \in \mathcal{R}$ we define $i_R := \max\{i \in W_{\mathrm{L}} : R \subseteq P^{(i)}\}$, i.e., $P^{(i_R)}$ is the inclusionwise minimal interval of a light node containing R. For $i \in W_{\mathrm{L}}$, let

$$\mathcal{R}(i) := \big\{R \in \mathcal{R} : i_R = i\big\}.$$

Furthermore, we also keep track of the gaps between the requests in $\mathcal{R}(i)$ as follows. For $i \in W_{\mathrm{L}}$, let

$$\bar{\mathcal{R}}(i) := \big\{\bar{R} \subseteq P^{(i)} : \bar{R} \text{ is a maximal path in } P^{(i)} \setminus \textstyle\bigcup_{R\in\mathcal{R}(i)} R \text{ and } \bar{R} \subseteq P^{(j)} \text{ for some } j \in \mathrm{ch}(i)\big\}.$$

Note that $R' \cap R'' = \emptyset$ for all $R', R'' \in \mathcal{R}(i) \cup \bar{\mathcal{R}}(i), i \in W_{\mathrm{L}}$. However, $\bar{R} \in \bar{\mathcal{R}}(i)$ may contain a request $R \in \mathcal{R}(j)$ from descendants j of i. See Figure 4 c) for an illustration of the assignment.

For $i \in W_{\mathrm{L}}$, let $W_{\mathrm{H}}(i) := \mathrm{ch}(i)$ and $W_{\mathrm{L}}(i) := \{i' \in W : i' \in \mathrm{ch}(j) \text{ for some } j \in W_{\mathrm{H}}(i)\}$. Note that $W_{\mathrm{H}}(i) \subseteq W_{\mathrm{H}}$ and $W_{\mathrm{L}}(i) \subseteq W_{\mathrm{L}}$. Before we can state the algorithm for computing the second stage reply, we need one final lemma.

▶ **Lemma 12.** *Let $i \in W_L$. For every $R \in \mathcal{R}(i)$, there is a $j \in W_H(i)$ with $P^{(j)} \cap R \neq \emptyset$. For every $\bar{R} \in \bar{\mathcal{R}}(i)$, there is an $i' \in W_L(i)$ with $P^{(i')} \cap R \neq \emptyset$.*

We are now ready to state the algorithm. We first describe and discuss a simplified version, which yields an approximation guarantee of 19. At the end of the paper, we discuss how to slightly adapt the algorithm so as to obtain the factor of 10 given in Theorem 2.

Algorithm 2.2 works as follows:

(i) Create the matching M' by removing the following edges from M_1 for each $i \in W_{\mathrm{L}}$:
  1. The edge $e^{(i)}$ if $i \neq 0$ and $\mathcal{R}(i) \neq \emptyset$.
  2. For each $R \in \mathcal{R}(i)$ the edge $e^{(j^*_R)}$ where $j^*_R := \min\{j \in W_{\mathrm{H}}(i) : P^{(j)} \cap R \neq \emptyset\}$.
  3. For each $\bar{R} \in \bar{\mathcal{R}}(i)$ the edge $e^{(i^*_{\bar{R}})}$ where $i^*_{\bar{R}} := \min\{i' \in W_{\mathrm{L}}(i) : P^{(i')} \cap R \neq \emptyset\}$.

(ii) Let M'' be a min-cost matching on all vertices not covered by M' in G_2.

(iii) Return $M_2 := M' \cup M''$.

Let Z be indices of the edges removed in step (i). It is not hard to see that $|\bar{\mathcal{R}}(i)| \leq |\mathcal{R}(i)| - 1$ for each $i \in W_{\mathrm{L}}$ and therefore $|Z| \leq 2k$, bounding the recourse of Algorithm 2.2 as intended.

▶ **Lemma 13.** $|Z| \leq 2k$.

Now let $Y := W \setminus (Z \cup \{0\})$ be the nodes corresponding to edges that have not been removed and

$$\bar{Y} := \{i \in Y : T[i] \setminus \{0, i\} \subseteq Z\}$$

the nodes that correspond to maximal intervals that have not been removed.

The following lemma is a consequence of the exponential decay property. It shows that in order to establish a bound on the cost of M_2, it is enough to bound the cost of all paths $P^{(i)}$ for $i \in \bar{Y}$.

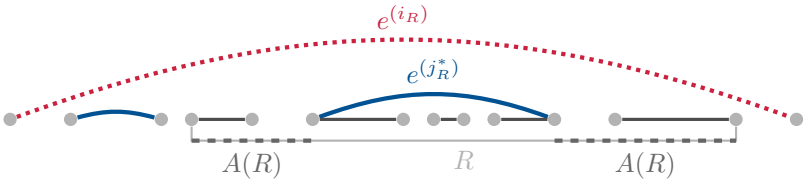


Figure 5 Illustration of the proof of Lemma 15. The solid dark grey lines depict edges in $O_1 \cap R$. At the time when Algorithm 2.1 constructed $P^{(j_R^*)}$, no other child interval of i_R intersecting with R was present. Thus $X^{(j_R^*)} \cap R = X^{(i_R+1)} \cap R = O_1 \cap R$. If $A(R)$ was O_1-heavy, then $P^{(j_R^*)} \cup A(R)$ would be an O_1-heavy path of higher O_1-gain than $P^{(j_R^*)}$, contradicting the greedy construction.

▶ **Lemma 14.** $c(M_2) \leq c(O_2) + 3\sum_{i \in \bar{Y}} c(P^{(i)})$.

It remains to bound the cost of the paths associated with the tree nodes in $\bar{Y}$. We establish a charging scheme by partitioning the line into three areas A, B, C:

1. For $R \in \mathcal{R}$, let $A(R) := R \setminus P^{(j_R^*)}$. We define $A := \bigcup_{R \in \mathcal{R}} A(R)$.
2. For $i \in W_{\mathrm{L}}$ and $\bar{R} \in \bar{\mathcal{R}}(i)$, let $B(\bar{R}) := \bar{R} \setminus \bigcup_{i' \in W_{\mathrm{L}}(i) \cap Z} P^{(i')}$.
   We define $B := \bigcup_{\bar{R} \in \bar{\mathcal{R}}} B(\bar{R})$.
3. We define $C := L \setminus (A \cup B)$.

Consider a set $A(R)$ for some $R \in \mathcal{R}$. Recall that i_R is the index of the smallest light interval constructed by Algorithm 2.1 containing R and that j_R^* is the first child interval of i_R created by Algorithm 2.1 that intersects R. From the choice of j_R^* and the greedy construction of $P^{(j_R^*)}$ as a path of maximum $X^{(j_R^*)}$-gain we can conclude that $A(R)$ is not O_1-heavy; see Figure 5 for an illustration. Therefore $c(A(R) \setminus O_1) > \frac{1}{3}c(A(R))$. Note that $O_2 \cap A(R) = A(R) \setminus O_1$, because $A(R) \subseteq R$. Hence we obtain the following lemma.

▶ **Lemma 15.** *Let $R \in \mathcal{R}$. Then $\frac{1}{3}c(A(R)) \leq c(O_2 \cap A(R))$.*

A similar argument implies the same bound for all sets of the type $B(\bar{R})$ for some $\bar{R} \in \bar{\mathcal{R}}(i)$ and $i \in W_{\mathrm{L}}$.

▶ **Lemma 16.** *Let $\bar{R} \in \bar{\mathcal{R}}$. Then $\frac{1}{3}c(B(\bar{R})) \leq c(O_2 \cap B(\bar{R}))$.*

Furthermore, one can show that the sets of the form $A(R)$ and $B(\bar{R})$ and the set C form a partition of L (see the appendix of the full version). We define $x : L \to \mathbb{R}_+$ by

$$x(e) := \begin{cases} \frac{1}{3} & \text{if } e \in A \cup B \\ 1 & \text{if } e \in C \cap O_2 \\ 0 & \text{if } e \in C \setminus O_2 \end{cases}$$

and obtain the following lemma as a consequence of Lemmas 15 and 16.

▶ **Lemma 17.** $\sum_{e \in L} c(e)x(e) \leq c(O_2 \cap L)$.

We are now able to bound the cost of each path $P^{(i)}$ for $i \in \bar{Y}$ against its local budget $\sum_{e \in P^{(i)}} c(e)x(e)$. We consider the cases where $i \in \bar{Y}$ corresponds to a heavy and light edge in Lemma 18 and Lemma 19, respectively.

▶ **Lemma 18.** *Let $j \in \bar{Y} \cap W_H$. Then $c(P^{(j)}) \leq 6\sum_{e \in P^{(j)}} c(e)x(e)$.*

Proof. We first show that each request intersecting with $P^{(j)}$ is either contained in a child interval of j or the edges in $P^{(j)}$ intersecting with the request are covered by A.

▷ Claim. Let $R \in \mathcal{R}$ with $R \cap P^{(j)} \neq \emptyset$. Then $R \cap P^{(j)} \subseteq A(R)$ or $R \subseteq P^{(i')}$ for some $i' \in \mathrm{ch}(j)$.

Proof. Assume $R \not\subseteq P^{(i')}$ for any $i' \in \mathrm{ch}(j)$. In particular, this implies that $i_R \notin \mathrm{desc}(j)$. Let i be the parent node of j in T. We first exclude the possibility that i_R is an ancestor of i. Indeed, if this was the case then $R \not\subseteq P^{(i)}$ and thus by Assumption A, $P^{(i)} \subseteq R$. But then $P^{(i)}$ can neither contain other request intervals, nor can it intersect any non-request intervals. Therefore $\mathcal{R}(i) = \emptyset$ and $i \neq i^*_{\bar{R}}$ for all $\bar{R} \in \bigcup_{i' \in W_{\mathrm{L}}} \bar{\mathcal{R}}(i')$. Therefore, $i \notin Z$, a contradiction to $j \in \bar{Y}$. This implies that $i_R = i$. Further note that $j \in \bar{Y}$ implies $j \neq j^*_R$ for all $R \in \mathcal{R}$, as otherwise, j would have been tagged for removal. We conclude that $j^*_R \in \mathrm{ch}(i) \setminus \{j\}$ and thus $R \cap P^{(j)} \subseteq R \setminus P^{(j^*_R)} = A(R)$, as $P^{(j)}$ and $P^{(j^*_R)}$ are disjoint. ◁

Let $Q := \bigcup_{i' \in \mathrm{ch}(j)} P^{(i')}$. Consider $e \in P^{(j)} \cap O_1$. Note that the claim implies that if $e \notin A \cup Q$, then $e \in O_2$. We conclude that $(P^{(j)} \cap O_1) \setminus Q \subseteq A \cup B \cup (C \cap O_2)$. Further, a variant of Lemma 9 implies that $c(P^{(j)} \cap O_1) \leq \frac{4}{3} c((P^{(j)} \cap O_1) \setminus Q)$ (see appendix of the full version for details). We obtain

$$c(P^{(j)}) \leq \tfrac{3}{2} c(P^{(j)} \cap O_1) \leq 2c((P^{(j)} \cap O_1) \setminus Q) \leq 6 \textstyle\sum_{e \in P^{(j)}} c(e)x(e),$$

where the first inequality follow from the fact that $P^{(j)}$ is O_1-heavy and the last inequality follows from the fact that $x(e) \geq 1/3$ for every $e \in A \cup B \cup (C \cap O_2)$. This proves the lemma. ◀

▶ **Lemma 19.** *Let $i \in \bar{Y} \cap W_L$. Then $c(P^{(i)}) \leq 3 \sum_{e \in P^{(i)}} c(e)x(e)$.*

The proof of Lemma 19 is given in the appendix of the full version. We state a short proof sketch for intuition. Let $i \in \bar{Y} \cap W_{\mathrm{L}}$. Due to the removal of light edges with associated requests and Assumption A, we know that $P^{(i)}$ either does not intersect with any request or $P^{(i)}$ is completely covered by a request. In the former case one can show that $P^{(i)}$ is included in B and hence the lemma holds. In the latter case, $P^{(i)}$ is either included in A and hence the lemma holds or it is included in C in which case the lemma holds since $P^{(i)}$ is O_1-light.

As a consequence, we can now show a first constant factor bound. Because the paths $P^{(i)}$ for $i \in \bar{Y}$ are pairwise disjoint, Lemmas 17 to 19 imply $\sum_{i \in \bar{Y}} c(P^{(i)}) \leq 6c(O_2)$. Plugging this into Lemma 14 we obtain

$$c(M_2) \leq c(O_2) + 3 \textstyle\sum_{i \in \bar{Y}} c(P^{(i)}) \leq 19c(O_2).$$

Improvement of approximation factor. We can improve the approximation factor from 19 to 10 by a slight modification of the set of edges removed in Stage 2. To this end, note that the factor for the bound given in Lemma 18 is greater than the one given in Lemma 19. Indeed, the only reason for the weaker bound is that edges in $\bar{Y} \cap W_{\mathrm{H}}$ can have descendants with associated requests. Excluding this case improves the factor within the bound of the lemma from 6 to 3. We formalize this in the following lemma.

▶ **Lemma 20.** *Let $j \in \bar{Y} \cap W_H$ such that $\mathrm{ch}(j) \cap Z = \emptyset$. Then $c(P^{(j)}) \leq 3 \sum_{e \in P^{(j)}} c(e)x(e)$.*

We now modify Algorithm 2.2 as follows: Compute the set Z of edges tagged for removal by Algorithm 2.2. Now construct the set Z' by defining

$$\bar{H} := \{j \in W_{\mathrm{H}} \setminus Z \,:\, \mathrm{ch}(j) \cap Z \neq \emptyset\} \qquad \text{and} \qquad Z' := (Z \cup \bar{H}) \setminus \textstyle\bigcup_{j \in \bar{H}} \mathrm{ch}(j).$$

Now execute step 2 of Algorithm 2.2 with Z' instead of Z, i.e., remove the edges with indices in Z' and connect the unmatched vertices by a min-cost matching. It is easy to see that $|Z'| \leq |Z|$ and therefore the recourse factor is still bounded by 2. For analyzing the approximation factor, we define $Y' := W \setminus (Z' \cup \{0\})$ the nodes corresponding to edges that have not been removed and $\bar{Y}' := \{i \in Y : T[i] \setminus \{0, i\} \subseteq Z'\}$ in analogy to the original analysis. It is easy to see that $\mathrm{ch}(j) \cap Z = \emptyset$ for every $j \in \bar{Y}' \cap W_{\mathrm{H}} = \bar{Y} \setminus \bar{H}$ and hence $c(P^{(j)}) \leq 3 \sum_{e \in P^{(j)}} c(e)x(e)$ by Lemma 20. Furthermore, if $i \in \bar{Y}' \cap W_{\mathrm{L}}$ then either $i \in \bar{Y}$ and $c(P^{(i)}) \leq 3 \sum_{e \in P^{(i)}} c(e)x(e)$ by Lemma 19, or $i \in \mathrm{ch}(j)$ for some $j \in \bar{Y} \setminus \bar{Y}' = \bar{H}$. Note that Lemmas 9 and 18 imply

$$\sum_{j \in \bar{H}} \sum_{i \in \mathrm{ch}(j)} c(P^{(i)}) \leq \tfrac{1}{2} \sum_{j \in \bar{H}} c(P^{(j)}) \leq 3 \sum_{j \in \bar{H}} \sum_{e \in P^{(j)}} c(e)x(e).$$

We thus obtain $\sum_{i \in \bar{Y}'} c(P^{(i)}) \leq 3c(O_2)$ and hence for the modified algorithm it holds that

$$c(M_2) \leq c(O_2) + 3 \sum_{i \in \bar{Y}'} c(P^{(i)}) \leq 10c(O_2).$$

References

1. Michael Ball, Lawrence Bodin, and Robert Dial. A Matching Based Heuristic for Scheduling Mass Transit Crews and Vehicles. *Transportation Science*, 17:4–31, 1983.
2. Colin E. Bell. Weighted matching with vertex weights: An application to scheduling training sessions in NASA space shuttle cockpit simulators. *European Journal of Operational Research*, 73:443–449, 1994.
3. Aaron Bernstein, Jacob Holm, and Eva Rotenberg. Online Bipartite Matching with Amortized $O(log^2 n)$ Replacements. In *Proceedings of the Twenty-ninth Annual ACM SIAM Symposium on Discrete Algorithms (SODA '18)*, pages 947–959, 2018.
4. Kamalika Chaudhuri, Constantinos Daskalakis, Robert D. Kleinberg, and Henry Lin. Online Bipartite Perfect Matching With Augmentations. In *Proceedings of the Twenty-eight IEEE Conference on Computer Communications (INFOCOM '09)*, pages 1044–1052, 2009.
5. Nicos Christofides. Worst-Case Analysis of a New Heuristic for the Traveling Salesman Problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.
6. U. Derigs and A. Metz. A matching-based approach for solving a delivery/pick-up vehicle routing problem with time constraints. *Operations-Research-Spektrum*, 14:91–106, 1992.
7. Mitre Costa Dourado, Dirk Meierling, Lucia D. Penso, Dieter Rautenbach, Fabio Protti, and Aline Ribeiro de Almeida. Robust recoverable perfect matchings. *Networks*, 66:210–213, 2015.
8. Jack Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards, Section B*, 69:125–130, 1965.
9. Bernhard Fuchs, Winfried Hochstättler, and Walter Kern. Online Matching on a Line. *Theoretical Computer Science*, 332:251–264, 2005.
10. A. M. Geoffrion and R. Nauss. Parametric and Postoptimality Analysis in Integer Linear Programming. *Management Science*, 23:453–466, 1977.
11. Edward F. Grove, Ming-Yang Kao, P. Krishnan, and Jeffrey Scott Vitter. Online perfect matching and mobile computing. In *Algorithms and Data Structures (WADS '95)*, volume 955 of *Lecture Notes in Computer Science*, pages 194–205. Springer, 1995.
12. A. Gu, A. Gupta, and A. Kumar. The power of deferral: maintaining a constant-competitive Steiner tree online. In *Proceedings of the Forty-fifth Annual ACM Symposium on Symposium on Theory of Computing (STOC '13)*, pages 525–534, 2013.
13. A. Gupta and A. Kumar. Online Steiner Tree with Deletions. In *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '14)*, pages 455–467, 2014.
14. Anupam Gupta, Amit Kumar, and Cliff Stein. Maintaining Assignments Online: Matching, Scheduling, and Flows. In *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '14)*, pages 468–479, 2014.

15 Anupam Gupta and Kevin Lewi. The Online Metric Matching Problem for Doubling Metrics. In *Proceedings of the Thirty-ninth International Colloquium on Automata, Languages and Programming (ICALP '12)*, pages 424–435, 2012.

16 Bala Kalyanasundaram and Kirk Pruhs. Online Weighted Matching. *Journal of Algorithms*, 14:478–488, 1993.

17 Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An Optimal Algorithm for On-line Bipartite Matching. In *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing (STOC '90)*, pages 352–358, 1990.

18 Samir Khuller, Stephen G. Mitchell, and Vijay V. Vazirani. On-line Algorithms for Weighted Bipartite Matching and Stable Marriages. *Theoretical Computer Science*, 127:255–267, 1994.

19 Christian Liebchen, Marco Lübbecke, Rolf Möhring, and Sebastian Stiller. *The Concept of Recoverable Robustness, Linear Programming Recovery, and Railway Applications*, pages 1–27. Springer Berlin Heidelberg, 2009.

20 Nicole Megow, Martin Skutella, José Verschae, and Andreas Wiese. The power of recourse for online MST and TSP. *SIAM Journal on Computing*, 45:859–880, 2016.

21 K. Nayyar and S. Raghvendra. An Input Sensitive Online Algorithm for the Metric Bipartite Matching Problem. In *Proceedings of the Fifty-eight Annual IEEE Symposium on Foundations of Computer Science (FOCS '17)*, pages 505–515, 2017.

22 Snjólfur Ólafsson. Weighted Matching in Chess Tournaments. *The Journal of the Operational Research Society*, 41:17–24, 1990.

23 William R. Pulleyblank. *Edmonds, Matching and the Birth of Polyhedral Combinatorics*, pages 181–197. Springer Berlin Heidelberg, 2012.

24 Sharath Raghvendra. A Robust and Optimal Online Algorithm for Minimum Metric Bipartite Matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM '16)*, volume 60 of *Leibniz International Proceedings in Informatics*, pages 18:1–18:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.

25 Sharath Raghvendra. Optimal Analysis of an Online Algorithm for the Bipartite Matching Problem on a Line, 2018. `arXiv:1803.07206`.

26 E. Reingold and R. Tarjan. On a Greedy Heuristic for Complete Matching. *SIAM Journal on Computing*, 10:676–681, 1981.

The Minimum Cost Query Problem on Matroids with Uncertainty Areas

Arturo I. Merino
Dept. of Mathematical Engineering and CMM, Universidad de Chile & UMI-CNRS 2807, Santiago, Chile
amerino@dim.uchile.cl

José A. Soto[1]
Dept. of Mathematical Engineering and CMM, Universidad de Chile & UMI-CNRS 2807, Santiago, Chile
www.dim.uchile.cl/~jsoto
jsoto@dim.uchile.cl

Abstract

We study the minimum weight basis problem on matroid when elements' weights are uncertain. For each element we only know a set of possible values (an uncertainty area) that contains its real weight. In some cases there exist bases that are uniformly optimal, that is, they are minimum weight bases for every possible weight function obeying the uncertainty areas. In other cases, computing such a basis is not possible unless we perform some queries for the exact value of some elements.

Our main result is a polynomial time algorithm for the following problem. Given a matroid with uncertainty areas and a query cost function on its elements, find the set of elements of minimum total cost that we need to simultaneously query such that, no matter their revelation, the resulting instance admits a uniformly optimal base. We also provide combinatorial characterizations of all uniformly optimal bases, when one exists; and of all sets of queries that can be performed so that after revealing the corresponding weights the resulting instance admits a uniformly optimal base.

2012 ACM Subject Classification Mathematics of computing → Matroids and greedoids

Keywords and phrases Minimum spanning tree, matroids, uncertainty, queries

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.83

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at https://arxiv.org/abs/1904.11668.

Funding Work supported by Conicyt via Fondecyt Grant 1181180 and PIA AFB-170001.

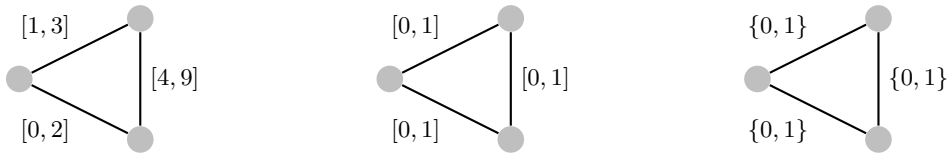
1 Introduction

We study fundamental combinatorial optimization problems on weighted structures where the numerical data is uncertain but it can be queried at a cost. We focus on the problem of finding a minimum weight base of a matroid under uncertainty, a problem that includes finding the smallest k elements of a list and the minimum spanning tree (MST) problem. In our setting, for every element e of the matroid we know a set $\mathcal{A}(e)$, called uncertainty area, of possible values that contains its real weight w_e. We can reveal this real weight by paying some query cost c_e. We assume that the queries are done in a non-adaptive way, or equivalently, that all the elements queried reveal their values at the same time. A set of elements F is a *feasible query* if for every possible revelation of the weights of F it is possible to compute a minimum weight base T of the resulting instance. The task of the *Minimum Cost Query Problem on Matroids* is to determine a minimum-cost feasible query.

[1] Corresponding author

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 83; pp. 83:1–83:14


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



(a) The empty set is feasible. **(b)** Only the full set is feasible. **(c)** Any 2-set is feasible.

Figure 1 Feasible queries.

To better illustrate this, consider the problem of computing an MST of a triangle graph in three possible situations as shown in Figure 1. In the first situation, the edges with areas $[0, 2]$ and $[1, 3]$ always form an MST, so we don't need to query any element. In this case we say that the matroid (the graph) admits a *uniformly optimal basis*, that is, a basis (a spanning tree) having minimum weight for every possible realization of the elements' weights. In the second situation, all edges have uncertainty area $[0, 1]$ and the only feasible query is the entire set of edges. For if we only query two edges, we could be in a situation where both have weight $1/2$. With that information we cannot compute an MST: if the unqueried edge e had weight 0, then e must be in the MST. However, if it had weight 1, then e cannot be in an MST. In the last situation, all uncertainty areas are finite: they have two elements $\{0, 1\}$. Here, every set of size two is feasible. Indeed, if both elements reveal a weight of 0, then they form an MST. Otherwise, the set obtained by deleting any edge with weight 1 is an MST.

Paper outline and results

In Section 2 we give formal definitions and show a simple but strong result: the uniformly optimal bases (UOB) of an uncertainty matroid form the bases of a second matroid. In Section 3 we extend the classic MST red and blue rules to uncertainty matroids, introducing the concept of *colored* elements, and study their properties. We then show that an uncertainty matroid admits a UOB if and only if all its elements are colored, and use this to give a combinatorial description of the matroid of uniformly optimal bases. We provide polynomial time algorithms for testing the existence of a UOB and for finding one if it exists. In Section 4 we study the minimum cost feasible query problem in detail. By using our coloring framework we construct a partition of the elements into groups that characterize minimal feasible queries. We show that every minimal feasible query set is formed by taking the first group (denoted as the core) completely and by deleting exactly one element from each other group. Our main result is an algorithm to find this partition that we use to fully solve the minimum cost feasible query problem on matroids. Unlike related work on the MST, the uncertainty areas in our setting can be arbitrary sets of real numbers, and not just intervals. Our algorithms only assume access to an independence oracle for the matroid, and a very mild type of access to the uncertainty areas. At the end of our study we show that the interval uncertainty area case is special, as there is a unique minimal size feasible query. We also relate the solutions for the MST with $\{0, 1\}$-areas case with the 2-connected components of the graph.

Related Work

Traditional research in optimization with uncertain data mostly focuses on finding solutions whose value is good, either in the worst case (robust optimization) or in some probabilistic sense (stochastic optimization), without gaining new information about the uncertain data. Our problem contributes to the *query model setting*, a different approach that has gained some strength in the last years. In this model one assumes that the algorithm can learn the

exact value of an uncertain input data by paying some query cost in order to solve a certain problem P (e.g., determining the MST of a graph). The aim is to minimize the cost of the queries while guaranteeing that an exact/approximate solution can be computed. Work in this area (see [3] for a survey) falls into three main categories:

Adaptive online. In this setting, the algorithm can query elements one by one, using the information revealed until a certain step to guide the decision of the next element to query. Even though algorithms for this version can be analyzed with a traditional worst-case approach (e.g., minimizing the depth of the decision tree associated to the algorithm's strategy), most of the work in this setting prefers to measure performance in terms of competitive analysis, comparing the number of queries an algorithm makes with the minimum number of queries that an adversary algorithm (that knows the real values beforehand) would make in order to verify that an answer is correct. Probably the first one to consider this model is Kahan [9] who provide optimal online algorithms to compute *all the elements* achieving the maximum, median and minimum gap of a list of closed intervals. Feder et al. [7] devise optimal online competitive algorithms to determine the numerical *value* of the median, and more generally, the K-th top element of the list within some prescribed tolerance. Bruce et al. [1] introduce a general method, called the *witness algorithm*, for adaptive online problems with open intervals and singletons uncertainty areas. They apply this method to geometric problems such as finding all maximum points in the plane from a family with uncertain coordinates. Erlebach et al. [5] studied the MST problem under two types of uncertainty, the edge uncertainty one (which is the same as ours) and the vertex uncertainty setting, in which the graph is complete, the vertices are points in the plane whose coordinates are uncertain, and the weight of an edge is the distance between its endpoints. They get 2 and 4 competitive algorithms, respectively, for both types of uncertainty, under open intervals and singletons areas, which is optimal for deterministic algorithms. The algorithm for edge uncertainty, denoted as U-RED, was later extended by Erlebach, Hoffmann and Kammer [4] to the minimum weight base problem on matroids achieving an optimal 2 competitive algorithm. Megow, Meißner and Skutella [13] show that by using randomization one can do better, lowering the competitive ratio down to $1 + 1/\sqrt{2}$. They also studied the non-uniform cost case. Gupta et al. [8] studied variants where queries return refined estimates of the areas, instead of a single value.

Verification. The verification problem is the one the offline adversary of the previous setting has to solve. That is, given both the uncertainty areas and a family of assumed real values, to determine the minimum number of queries one has to make so that, if the values obtained from the queries and the assumed values coincide, then no more queries are needed in order to obtain an optimal solution. Charalambous and Hoffman [2] show that the verification problem for maximal points in the plane is NP-hard for uncertainty areas of size at most 2. Erlebach and Hoffmann [3] show that the verification problem of MST with (open interval and singleton) uncertainty in the edges is in P, while that for vertex uncertainty is NP-hard.

Non-adaptive online. This setting encompasses our work and is sometimes called the *offline* problem. In it, an algorithm must determine a set F of queries to perform simultaneously, in order to have enough information to solve the problem. The only conceptual difference between the non-adaptive online problem and the verification one, is that in the latter, the algorithm can make use of the real values of the elements to guide the decision of which queries to perform, while in the former, that information is not available. Feder et al. [7] provide optimal algorithms for finding the K-th top element of a list up to additive tolerance.

Later, Feder et al. [6] consider the problem of finding the shortest s-t path on a DAG with closed-intervals uncertainty on the edges. They show that determining the length of the shortest s-t path within a given additive error is neither in NP nor in co-NP unless NP=co-NP, and provide exact algorithms for some special cases. To the best of our knowledge, the MST and, more generally, the matroid case have not been considered before this work.

Further related work. A common approach to deal with closed-interval uncertainty for a problem, without querying extra information, is to find a solution that minimizes the maximal regret, that is, the difference between the (real) weight of the chosen solution and the weight of the best solution that could have been picked had the true weights been known. Zero maximal regret bases and uniformly optimal bases of a matroid coincide. In the context of the MST problem, Yaman et al. [16] characterize trees with zero regret when they exists. In [10], Kaspersky and Zielinsky give a 2-approximation algorithm for the minmax regret problem on general matroids with interval data, and in [11] they give algorithms to find zero maximal regret bases. It is worth noting that all these results assume interval areas on the elements, while we allow for arbitrary uncertainty areas.

2 Preliminaries – Uniformly Optimal Bases

We assume familiarity with basic concepts in matroid theory such as bases, independent sets, span, circuit, cocircuits and duality, contraction, deletion and matroid connectivity. For an introduction and specific results, we refer to Oxley's book [14]. However, most of this paper can be understood by having the graphic matroid of a connected graph in mind. This is a matroid whose elements are the edges. The bases, independent sets, circuits and cocircuits, are the spanning trees, forests, cycles and minimal edge cut-sets, respectively. An element e is in the span (cospan) of a set F if there is a circuit (cocircuit) in $F \cup \{e\}$ containing e. We also use the standard notation $X + e$ and $X - e$ to denote $X \cup \{e\}$ and $X \setminus \{e\}$.

▶ **Definition 1** (Uncertainty Matroid). *An uncertainty matroid is a pair $(\mathcal{M}, \mathcal{A})$ where $\mathcal{M} = (E, \mathcal{I})$ is a matroid and $\mathcal{A}\colon E \to 2^{\mathbb{R}} \setminus \{\emptyset\}$ is a function mapping each element $e \in E$ in the ground set to a nonempty set $\mathcal{A}(e)$ of real numbers denoted the uncertainty area of e. We denote $\inf \mathcal{A}(e)$ by L_e and $\sup \mathcal{A}(e)$ by U_e.*

If $\mathcal{A}(e)$ is a singleton, we say that the element e is *certain*, otherwise, we say it is *uncertain*. If all the elements are certain, then we can identify $\mathcal{A}$ with the associated weight function $w\colon E \to \mathbb{R}$, such that $\mathcal{A}(e) = \{w_e\}$ so that $(\mathcal{M}, w)$ becomes a weighted matroid.

▶ Remark 2. For our algorithms, we assume access to matroid $\mathcal{M}$ via an independence oracle. We also need a very mild access to the uncertainty areas. More precisely, we assume that both L_e and U_e are known for every element, and that for every pair of (not necessarily distinct) elements e and f we can test if $(L_e, U_e) \cap \mathcal{A}(f)$ is empty in constant time. A polynomial time algorithm will, therefore, only use a polynomial number of calls to the independence oracle and to the uncertainty areas. Furthermore, for simplicity we will assume that all the infima and suprema L_e and U_e are finite. Otherwise we can apply a suitable strictly increasing function mapping the reals to a bounded set (such as arctan), and work in its image instead.

Intuitively, an uncertainty matroid models the situation in which we do not know the actual weight of every element e in the matroid, but we do know a set $\mathcal{A}(e)$ containing it. We can learn the actual weight of e by querying it. In this work we are concerned with non-adaptive (simultaneous) queries. The new uncertainty area function obtained after querying a subset of elements will be called a *revelation* of $\mathcal{A}$. The formal definition is below.

▶ **Definition 3** (Revelations and realizations). *Let $X \subseteq E$. A revelation of X in $(\mathcal{M}, \mathcal{A})$ is a function $\mathcal{B}\colon E \to 2^{\mathbb{R}} \setminus \{\emptyset\}$ such that:*

(i) *$\forall e \in X$, $\mathcal{B}(e) = \{b_e\} \subseteq \mathcal{A}(e)$ is a singleton, and*

(ii) *$\forall e \in E \setminus X$, $\mathcal{B}(e) = \mathcal{A}(e)$.*

In particular, $(\mathcal{M}, \mathcal{B})$ is also an uncertainty matroid. A realization *is a revelation of the entire ground set E. The collection of all revelations of X in $(\mathcal{M}, \mathcal{A})$ is denoted by $\mathcal{R}(X, \mathcal{A})$.*

Suppose we want to compute a minimum weight basis of a matroid, but we only know uncertainty areas for its elements. In certain situations (e.g., Figure 1 (a)), we can find sets that are optimal bases for every realization, we call them uniformly optimal bases.

▶ **Definition 4** (Uniformly optimal bases). *A set $T \subseteq E$ is a uniformly optimal basis of $(\mathcal{M}, \mathcal{A})$ (or simply, an $\mathcal{A}$-basis) if for every realization w, T is a minimum weight basis (a w-basis).*

Recall that a nonempty family of sets form the bases of a matroid if and only if they satisfy the strong basis exchange axiom. Our first basic result is the following:

▶ **Lemma 5.** *Let $\mathfrak{B}$ be the collection of all uniformly optimal bases of $(\mathcal{M}, \mathcal{A})$. Suppose $\mathfrak{B} \neq \emptyset$ and let T_1 and T_2 be two sets in $\mathfrak{B}$. If e is an element in $T_1 \setminus T_2$ then:*

(i) *e is certain, and*

(ii) *strong basis exchange holds, i.e., there is an element f in $T_2 \setminus T_1$ such that both $T_1 - e + f$ and $T_2 - f + e$ are in $\mathfrak{B}$.*

In particular, if $\mathfrak{B} \neq \emptyset$, then $\mathfrak{B}$ is the set of bases of a matroid, that we denote by $\mathtt{mat}(\mathcal{M}, \mathcal{A})$

Proof. Let $e \in T_1 \setminus T_2$. By strong basis exchange of $\mathcal{M}$, there is an element $f \in T_2 \setminus T_1$ such that both $T_1 - e + f$ and $T_2 + e - f$ are bases of $\mathcal{M}$. Assume first by contradiction that e is uncertain, then there is a realization $w \in \mathcal{R}(E, \mathcal{A})$ such that $w_e \neq w_f$. If $w_e > w_f$ then $w(T_1 - e + f) < w(T_1)$ contradicting the fact that T_1 is uniformly optimal. On the other hand, if $w_e < w_f$ then $w(T_2 + e - f) < w(T_2)$ contradicting that T_2 is uniformly optimal. We conclude not only that e is certain, but also that in every realization $w_f = w_e$. In particular, for every realization w, $w(T_1 - e + f) = w(T_1) = w(T_2) = w(T_2 - f + e)$, i.e. both $T_1 - e + f$ and $T_2 - f + e$ are uniformly optimal bases. ◀

3 Blue and red rules for uncertainty matroids

By Lemma 5, we conclude that if an uncertain element e is in some uniformly optimal basis then it is in *every* uniformly optimal basis. Our next task is to characterize the set of uncertain elements that are in every uniformly optimal basis. Now it is useful to remember the classic blue and red rules for computing an MST. An edge of a weighted graph is called blue if it is in at least one MST, and it is called red if it is outside at least one MST. Virtually every algorithm for the MST work in steps: if it detects a red edge from the current graph, it deletes it from the graph, and if it detects a blue edge then it adds it to the solution and contracts it. The following definitions extend the coloring notion to uncertainty matroids.

▶ **Definition 6** (Blue and red elements). *An element $e \in E$ is blue (resp. red) if for every realization $w \in \mathcal{R}(E, \mathcal{A})$ there exists a w-basis T such that $e \in T$ (resp. $e \notin T$). We say that e is colored if e is red or blue (or both at the same time), otherwise we say that e is uncolored.*

Note that e is blue (resp. red) on $(\mathcal{M}, \mathcal{A})$ if and only if e is blue (resp. red) on $(\mathcal{M}, w)$ for every realization $w \in \mathcal{R}(E, \mathcal{A})$. Standard matroid arguments show that for any such w, if an element e is any heaviest element of a circuit, then it avoids some w-basis (i.e., it is red on $(\mathcal{M}, w)$), and if it is the unique heaviest element, then it cannot be in a w-basis (i.e., it is

not blue on $(\mathcal{M}, w)$). Blue elements in graphs / matroids with interval uncertainty areas have been studied before under the name of strong edges [16] or necessarily optimal elements [11]. We start with a basic result on colored elements, they are preserved under revelations.

▶ **Lemma 7.** *Let $X \subseteq E$ and $\mathcal{B} \in \mathcal{R}(X, \mathcal{A})$. If e is blue (resp. red) in $(\mathcal{M}, \mathcal{A})$, then it is blue (resp. red) in $(\mathcal{M}, \mathcal{B})$.*

Proof. Suppose e is blue in $(\mathcal{M}, \mathcal{A})$ and consider any $w \in \mathcal{R}(E, \mathcal{B})$. Since $\mathcal{R}(E, \mathcal{B}) \subseteq \mathcal{R}(E, \mathcal{A})$ we get that there exists some w-basis T such that $e \in T$, implying that e is blue in $(\mathcal{M}, \mathcal{B})$. The proof works analogously if e is red in $(\mathcal{M}, \mathcal{A})$. ◀

We can characterize blue and red elements of an uncertainty matroid using its span and cospan functions[2] of a set X in the matroid $\mathcal{M}$ (resp., in the dual matroid $\mathcal{M}^*$). The span and cospan of a set can be computed using a polynomial number of calls to the independence oracle.

▶ **Lemma 8.** *Let $F(e) = \{f \in E - e : L_f < U_e\}$ and $F^*(e) = \{f \in E - e : L_e < U_f\}$.*
(i) *$e \in E$ is blue if and only if $e \notin \operatorname{span} F(e)$.*
(ii) *$e \in E$ is red if and only if $e \notin \operatorname{cospan} F^*(e)$.*
In particular, we can test in polynomial time if any element is blue, red, both or none.

▶ Remark 9. Proofs involving blue and red elements follow the same main ideas. Most of the time we only consider the blue case, since the red case follows by *duality arguments*. More precisely, consider the dual matroid $\mathcal{M}^*$ with inverted uncertainty area function $-\mathcal{A}(e) = \{-x \colon x \in \mathcal{A}(e)\}$. Since the bases of $\mathcal{M}^*$ are complements of the bases of $\mathcal{M}$ we also get that the uniformly optimal bases of $(\mathcal{M}, \mathcal{A})$ are the complements of those in $(\mathcal{M}^*, -\mathcal{A})$, that the red elements in $(\mathcal{M}, \mathcal{A})$ are the blue elements of $(\mathcal{M}^*, -\mathcal{A})$ and vice versa.

Proof of Lemma 8. We only prove (i), since (ii) follows by duality arguments using that $F^*(e)$ is the analogue of $F(e)$ for $(\mathcal{M}^*, -\mathcal{A})$.

Let e be a blue element and let $K = \min_{f \in F(e)} (U_e - L_f) > 0$. For each $f \in E - e$ choose $\varepsilon_f \in [0, K/2)$ such that $L_f + \varepsilon_f \in \mathcal{A}(f)$. In a similar way, choose $\varepsilon_e \in [0, K/2)$ such that $U_e - \varepsilon_e \in \mathcal{A}(e)$. Consider the realization $w \in \mathcal{R}(E, \mathcal{A})$ given by $w_f = \begin{cases} U_e - \varepsilon_e & \text{if } f = e, \\ L_f + \varepsilon_f & \text{if } f \neq e. \end{cases}$

By construction, for every $f \in F(e)$, $w_f < w_e$. Suppose now that $e \in \operatorname{span} F(e)$. Then, there exists a circuit $C \subseteq F(e) + e$ such that e is the unique heaviest (with respect to w) element of C. This implies that e is outside every w-basis, which contradicts that e was blue.

Now let e be an element outside $\operatorname{span} F(e)$ and suppose that e is not blue. Then, there exists a realization $w \in \mathcal{R}(E, \mathcal{A})$ such that e is not in any w-basis. Choose T to be any w-basis and let C be the fundamental circuit[3] of $T + e$. As $e \notin \operatorname{span} F(e)$ we have that $C - e \not\subseteq F(e)$. Select any $f \in (C - e) \setminus F(e)$, then: $w_f \geq L_f \geq U_e \geq w_e$. We conclude that $T - f + e$ is a w-basis that contains e, which is a contradiction. ◀

Even though certain elements can be blue and red at the same time (e.g., in a circuit in which every element has the same weight every element has both colors), this is not possible for uncertain ones as shown by the next lemma whose proof is deferred to the full version.

---

[2] We recall that $\operatorname{span}(X)$ for $X \subseteq E$ is the unique maximal set $U \supseteq X$ with the same rank as X and $\operatorname{cospan}(X)$ is its span in the dual matroid. The span and cospan of a set in a matroid are related by the expression $\operatorname{cospan}(X) = X \cup \{e \in E : e \notin \operatorname{span}[(E - e) \setminus X]\}$

[3] Recall that if T is a basis and e an element, the fundamental circuit C of $T + e$ is the only circuit of $T + e$. For any $f \in C$, $T - f + e$ is a basis.

▶ **Lemma 10.** *If e is red and blue, then e is certain.*

The definition of blue and red will be useful even if e is certain, or if the uncertainty matroid has no uniformly optimal basis. The following lemma, whose proof is in the full version, highlights the utility of this definition.

▶ **Lemma 11.** *Let $(\mathcal{M}, \mathcal{A})$ be an uncertainty matroid such that an $\mathcal{A}$-basis exists.*

(i) *If e is blue (resp. red) then there exists an $\mathcal{A}$-basis T such that $e \in T$ (resp. $e \notin T$).*

(ii) *Let e be an uncertain element. Then e is blue (resp. red) if and only if e is inside (resp. outside) every $\mathcal{A}$-basis.*

The previous lemma shows that if the set of uniformly optimal bases is nonempty then all uncertain elements are colored either red or blue. Furthermore the uncertain elements contained in every $\mathcal{A}$-basis are exactly the blue uncertain elements. The next theorem, which is the main result of this section, shows that a converse also holds.

▶ **Theorem 12.** *An $\mathcal{A}$-basis exists if and only if every uncertain element is colored.*

In order to prove this theorem we need the following simple lemmas, whose proofs are deferred to the full version. Basically, they show that contracting blue elements and/or deleting red elements preserves structure and colors.

▶ **Lemma 13.** *Let $e \in E$ be a colored element (certain or uncertain).*

(i) *If e is blue: T is an $(\mathcal{M}/e, \mathcal{A}|_{E-e})$-basis if and only if $T + e$ is an $(\mathcal{M}, \mathcal{A})$-basis.*

(ii) *If e is red: T is an $(\mathcal{M} \setminus e, \mathcal{A}|_{E-e})$-basis if and only if T is an $(\mathcal{M}, \mathcal{A})$-basis.*

▶ **Lemma 14.** *Let $e \in E$ be a colored uncertain element, and $f \in E - e$.*

(i) *Suppose e is blue. If f is blue (resp. red) in $(\mathcal{M}, \mathcal{A})$, then f is blue (resp. red) in $(\mathcal{M}/e, \mathcal{A}|_{E-e})$.*

(ii) *Suppose e is red. If f is blue (resp. red) in $(\mathcal{M}, \mathcal{A})$, then f is blue (resp. red) in $(\mathcal{M} \setminus e, \mathcal{A}|_{E-e})$.*

Proof of Theorem 12. We only need to prove the converse and we proceed by induction on the number of uncertain elements k. If $k = 0$ an $\mathcal{A}$-basis is simply a basis of minimum weight, which clearly exists. Suppose that $k > 0$, and let $e \in E$ be any uncertain element. If e is blue, we have that every uncertain element of $E - e$ is colored in $(\mathcal{M}/e, \mathcal{A}|_{E-e})$, as colors were preserved. By inductive hypothesis, we have an $(\mathcal{M}/e, \mathcal{A}|_{E-e})$-basis T and by Lemma 13 we have that $T + e$ is an $(\mathcal{M}, \mathcal{A})$-basis. If e is red, one can proceed similarly but deleting instead. ◀

▶ Remark 15. The previous theorem gives an algorithmic way to test if $(\mathcal{M}, \mathcal{A})$ admits an $\mathcal{A}$-basis: we simply check if every element is colored, using Lemma 8. Let us now consider the problem of finding one such base. It is worth noting that algorithms for this task are available for closed interval and open interval uncertainty areas. For closed interval uncertainty areas, one can find an $\mathcal{A}$-basis by using Kaspersky and Zielinsky's approach [10] for finding a zero maximal regret basis. For open intervals (or singletons), one can simply run the 2-competitive U-RED algorithm by Erlebach, Hoffmann and Kammer [4] for the online adaptive variant: if this algorithm does not perform any query, then it outputs a uniformly optimal basis. Otherwise, the algorithm finds a witness set, i.e., a set for which at least one element must be queried in order to find a solution. In what follows we provide a new algorithm that finds $\mathcal{A}$-bases for arbitrary uncertainty areas (not just intervals).

If uniformly optimal bases exist, then the elements are partitioned into blue uncertain, red uncertain and certain elements. After contraction of the set B of all blue uncertain elements and deletion of the set R of all red uncertain elements, we are left with a matroid that only has certain elements, we call such weighted matroid the *certain weighted matroid* $(\mathcal{M}^c, w^c)$, where $\mathcal{M}^c = \mathcal{M}/B \setminus R$, and $w^c = \mathcal{A}|_{E\setminus(R\cup B)}$. The following theorem shows that every uniformly optimal basis arises by extending some optimal basis on the certain weighted matroid.

▶ **Theorem 16.** *Let $(\mathcal{M}, \mathcal{A})$ be an uncertainty matroid for which an $\mathcal{A}$-basis exists and $(\mathcal{M}^c, w^c)$ its certain weighted matroid. Then T is an $\mathcal{A}$-basis if and only if:*

(i) *T contains every blue uncertain element,*
(ii) *T avoids each red uncertain element, and*
(iii) *The certain elements of T form a minimum weight basis of $(\mathcal{M}^c, w^c)$.*

Proof. $\mathcal{A}$-bases always contain every blue uncertain element and avoid each red uncertain element by Lemma 11. By Lemma 14, we can delete each red uncertain element while preserving colors. More so, from Lemma 13 we conclude that T is uniformly optimal after these deletions. A similar argument allows us to now contract each blue uncertain element while preserving colors in each contraction. We are only left with the certain elements of T and by Lemma 13 we conclude they form a minimum weight basis of $(\mathcal{M}^c, w^c)$.

We show the converse by induction on the number of uncertain elements k of E. If $k = 0$, using (iii) we get that T is a minimum weight basis of $(\mathcal{M}^c, w^c)$. Noting that $\mathcal{M} = \mathcal{M}^c$ and $\mathcal{A} = w^c$ we conclude that T is an $(\mathcal{M}, \mathcal{A})$-basis.

If $k > 0$ select any uncertain element $e \in E$. If e is blue, we have from (i) that $e \in T$. As colors are preserved when contracting e, it follows by inductive hypothesis that $T - e$ is an $(\mathcal{M}/e, \mathcal{A}|_{E-e})$-basis, and using Lemma 13 we conclude that T is an $(\mathcal{M}, \mathcal{A})$-basis. If e is red, then $e \notin T$ by (ii). We now proceed as before but deleting e instead. ◀

Theorems 12 and 16 allow for algorithmic implementation. We can decide if an $\mathcal{A}$-basis exists by checking if all elements are colored. If every element is colored, we contract every blue uncertain element, delete each red uncertain element, compute a minimum weight basis of the certain weighted matroid and output the optimal certain basis along with every blue uncertain element. We summarize this result and discuss it in more detail in the full version. We also use the previous theorem to characterize the matroid $\mathtt{mat}(\mathcal{M}, \mathcal{A})$.

▶ **Corollary 17.** *There is an algorithm that finds an $\mathcal{A}$-basis or decides that none exists in polynomial time.*

▶ **Corollary 18.** *If $(\mathcal{M}, \mathcal{A})$ admits an $\mathcal{A}$-basis, then the matroid $\mathtt{mat}(\mathcal{M}, \mathcal{A})$ of all $\mathcal{A}$-bases is a sum of minors of $\mathcal{M}$. In particular, if $\mathcal{M}$ belongs to some minor closed class of matroids (e.g., graphic, linear, gammoid) then so does $\mathtt{mat}(\mathcal{M}, \mathcal{A})$.*

4 Feasible Queries

▶ **Definition 19.** *A set $F \subseteq E$ is a feasible query (or simply feasible) if no matter its revelation it guarantees the existence of a uniformly optimal basis. That is, $\forall \mathcal{B} \in \mathcal{R}(F, \mathcal{A})$ there exists some $\mathcal{B}$-basis.*

Any superset of a feasible query is also feasible. Using this, one can show that feasible sets for a given uncertainty area function are also feasible for any revelation of a subset.

▶ **Lemma 20.** *Let $F \subseteq E$ be feasible in $(\mathcal{M}, \mathcal{A})$ and $X \subseteq E$. If $\mathcal{B} \in \mathcal{R}(X, \mathcal{A})$, then F is feasible for $(\mathcal{M}, \mathcal{B})$.*

Proof. Since F is feasible in $(\mathcal{M}, \mathcal{A})$, so is $X \cup F$. Consider any revelation $\mathcal{B}' \in \mathcal{R}(X, \mathcal{B})$. Since $\mathcal{B}' \in \mathcal{R}(X \cup F, \mathcal{A})$ and $X \cup F$ is feasible, we conclude that there exists a $\mathcal{B}'$-basis. ◀

Since revealing elements that are certain does not yield extra information, we also get that all minimal (for inclusion) queries only contain uncertain elements. A simple, yet strong result is that it never pays off to query a colored element. Due to space consideration, we defer the proof of the next lemma to the full version.

▶ **Lemma 21.** *Let $X \subseteq E$ a feasible query. If $e \in X$ is colored, then $X - e$ is a feasible query. In particular, if X is a feasible query minimal for inclusion then X consists only of uncertain uncolored elements.*

An exciting application of this lemma is that the set of all uncolored uncertain elements is always a feasible query set. To see this, start with all the uncertain elements (which are a feasible query) and repeatedly remove colored elements while applying Lemma 21.

By Theorem 12, a set F is feasible only if after its revelation all uncertain elements are colored. So, consider any uncertain element e before any revelation. Intuitively, its color depends on the possible relative position between its real weight w_e and the real weight of elements that could potentially span it (or cospan it). In particular, it is not hard too see that the *color* of e is unaffected if we reveal an element f whose uncertainty area is too low (say $U_f \leq L_e$), because the real value of f will be in every case at most that of e. A similar situation happens if the uncertainty area is too high (say $U_e \leq L_f$). A complicated thing occurs if $\mathcal{A}(f)$ intersects (L_e, U_e), because after revealing f we may still don't know the relative positions of w_e and w_f until we reveal e. Finally, if f is in none of the previous situation, then after revealing it we will know for sure the relative position of w_e, w_f even without revealing e. The previous discussion motivates the following definitions.

▶ **Definition 22.** *For each $e \in E$ define the sets $\mathtt{low}(e)$, $\mathtt{mid}(e)$, $\mathtt{high}(e)$ and $\mathtt{both}(e)$ by:*

$$\mathtt{low}(e) = \{f \in E - e : U_f \leq L_e\}, \qquad \mathtt{high}(e) = \{f \in E - e : U_e \leq L_f\},$$
$$\mathtt{mid}(e) = \{f \in E - e : \mathcal{A}(f) \cap (L_e, U_e) \neq \emptyset\},$$
$$\mathtt{both}(e) = \{f \in E \setminus \mathtt{mid}(e) - e : \mathcal{A}(f) \cap (-\infty, L_e] \neq \emptyset \ \wedge \ \mathcal{A}(f) \cap [U_e, \infty) \neq \emptyset\}.$$

Note that that for $e \in E$, $F(e) = (E - e) \setminus \mathtt{high}(e)$ and $F^*(e) = (E - e) \setminus \mathtt{low}(e)$. Furthermore if e is uncertain (i.e., $L_e < U_e$) then $E - e$ is partitioned into the sets $\mathtt{low}(e), \mathtt{high}(e), \mathtt{mid}(e)$ and $\mathtt{both}(e)$.

▶ **Definition 23.** *For each $e \in E$ denote $\mathcal{M}/\mathtt{low}(e) \setminus \mathtt{high}(e)$ by $\mathcal{M}'_e$*

▶ Remark 24. In this section we talk about different revelations simultaneously (for instance, $\mathcal{A}$ and $\mathcal{B}$). We differentiate the objects that arise this way by using superscript denoting these dependencies. For example $F^{\mathcal{A}}(e)$ denotes $F(e)$ with respect to the areas given by $\mathcal{A}$.

The next technical lemma formalizes the idea that the elements that influence the color of an uncertain element e are those in $\mathtt{mid}(e)$ and those in $\mathtt{both}(e)$ that are not queried.

▶ **Lemma 25.** *Let $X \subseteq E$ and $\mathcal{B} \in \mathcal{R}(E \setminus X, \mathcal{A})$ a revelation of its complement. If $e \in X$ is uncertain and uncolored in $\mathcal{B}$, then there exists a circuit C in $\mathcal{M}'_e$ such that $e \in C$ and $(C - e) \cap [\mathtt{mid}^{\mathcal{A}}(e) \cup (X \cap \mathtt{both}^{\mathcal{A}}(e))] \neq \emptyset$.*

Proof. Consider the revelation $\widetilde{\mathcal{B}} \in \mathcal{R}(\texttt{both}(e) \setminus X, \mathcal{A})$ such that $\widetilde{\mathcal{B}}(f) = \mathcal{B}(f) (= \{B_f\})$ if $f \in \texttt{both}(e) \setminus X$ and $\widetilde{\mathcal{B}}(f) = \mathcal{A}(f)$ otherwise and note that $\mathcal{B} \in \mathcal{R}(E \setminus X, \widetilde{\mathcal{B}})$. By Lemma 7, since e is uncolored in $(\mathcal{M}, \mathcal{B})$, we get that e is also uncolored in $(\mathcal{M}, \widetilde{\mathcal{B}})$. Define the auxiliary sets $Y = \{f \in \texttt{both}^{\mathcal{A}}(e) \setminus X : B_f \leq L_e\}$ and $\overline{Y} = \{f \in \texttt{both}^{\mathcal{A}}(e) \setminus X : B_f > L_e\}$. Note that $\texttt{low}^{\widetilde{\mathcal{B}}}(e) = \texttt{low}^{\mathcal{A}}(e) \cup Y; \texttt{high}^{\widetilde{\mathcal{B}}}(e) = \texttt{high}^{\mathcal{A}}(e) \cup \overline{Y}; \texttt{mid}^{\widetilde{\mathcal{B}}}(e) = \texttt{mid}^{\mathcal{A}}(e)$; and $\texttt{both}^{\widetilde{\mathcal{B}}}(e) = X \cap \texttt{both}^{\mathcal{A}}(e)$. Since e is uncertain and it is not blue nor red in $(\mathcal{M}, \widetilde{B})$ we get: $e \in \text{span}[\texttt{low}^{\widetilde{\mathcal{B}}}(e) \cup \texttt{mid}^{\widetilde{\mathcal{B}}}(e) \cup \texttt{both}^{\widetilde{\mathcal{B}}}(e)] = \text{span}[\texttt{low}^{\mathcal{A}}(e) \cup \texttt{mid}^{\mathcal{A}}(e) \cup Y \cup (X \cap \texttt{both}^{\mathcal{A}}(e))]$, and $e \in \text{cospan}[\texttt{high}^{\widetilde{\mathcal{B}}}(e) \cup \texttt{mid}^{\widetilde{\mathcal{B}}}(e) \cup \texttt{both}^{\widetilde{\mathcal{B}}}(e)] = \text{cospan}[\texttt{high}^{\mathcal{A}}(e) \cup \texttt{mid}^{\mathcal{A}}(e) \cup \overline{Y} \cup (X \cap \texttt{both}^{\mathcal{A}}(e))]$. Using that $e \in \text{cospan}[Q]$ implies $e \notin \text{span}[(E - e) \setminus Q]$ for any $Q \subseteq E$, we conclude:

$$e \in \text{span}[\texttt{low}^{\mathcal{A}}(e) \cup \texttt{mid}^{\mathcal{A}}(e) \cup Y \cup (X \cap \texttt{both}^{\mathcal{A}}(e))] \setminus \text{span}[\texttt{low}^{\mathcal{A}}(e) \cup Y]. \tag{1}$$

From (1), $e \in \text{span}[\texttt{low}^{\mathcal{A}}(e) \cup \texttt{mid}^{\mathcal{A}}(e) \cup Y \cup (X \cap \texttt{both}^{\mathcal{A}}(e))] \setminus \texttt{low}^{\mathcal{A}}(e) = \text{span}_{M'_e}[\texttt{mid}^{\mathcal{A}}(e) \cup Y \cup (X \cap \texttt{both}^{\mathcal{A}}(e))]$, where the equality follows from properties of contraction and deletion. Therefore there is a circuit C in $\mathcal{M}'_e$ such that $e \in C$ and $C - e \subseteq \texttt{mid}^{\mathcal{A}}(e) \cup Y \cup X \cap \texttt{both}^{\mathcal{A}}(e)$. If $(C - e) \cap [\texttt{mid}^{\mathcal{A}}(e) \cup (X \cap \texttt{both}^{\mathcal{A}}(e))] = \emptyset$, we would have that $C - e \subseteq Y$, implying that $e \in \text{span}_{\mathcal{M}'_e} Y = \text{span}[\texttt{low}^{\mathcal{A}}(e) \cup Y] \setminus \texttt{low}^{\mathcal{A}}(e)$ which contradicts (1). ◀

The previous lemma is useful to characterize the sets that intersect every feasible set.

▶ **Definition 26.** *A set $X \subseteq E$ is a witness set if it intersects every feasible set.*

Witness sets have been studied before in the context of online adaptive algorithms for MST and matroids. Since the definition of feasible is slightly different in that settings (they are feasible for the verification problem, in which one knows the real values a priori), these witness sets are also different from ours.

▶ **Lemma 27.** *Let $X \subseteq E$. The following statements are equivalent:*

(i) *X is a witness set.*

(ii) *There exists an uncertain element $e \in X$ and a circuit C in $\mathcal{M}'_e$ such that $e \in C$ and $C \cap [\texttt{mid}(e) \cup (X \cap \texttt{both}(e))] \neq \emptyset$.*

Proof. Let X be a witness set. Since $E \setminus X$ isn't feasible there is a revelation $\mathcal{B} \in \mathcal{R}(E \setminus X, \mathcal{A})$ such that there is no $(\mathcal{M}, \mathcal{B})$-basis and by Theorem 12 we must have an element e that is uncertain and uncolored in $(\mathcal{M}, \mathcal{B})$. Note that $e \in X$ as every element in $E \setminus X$ is certain in $(\mathcal{M}, \mathcal{B})$. We conclude by using Lemma 25 on e.

For the converse, set $Y = [\texttt{both}(e) \setminus X] \cap C$, $\overline{Y} = [\texttt{both}(e) \setminus X] \setminus C$ and note that $e \in \text{span}_{\mathcal{M}'_e}(C - e) = \text{span}_{\mathcal{M}'_e}[(C \cap \texttt{mid}(e)) \cup (C \cap (\texttt{both}(e) \setminus X)) \cup (C \cap \texttt{both}(e) \cap X)] \subseteq \text{span}_{\mathcal{M}'_e}[\texttt{mid}(e) \cup Y \cup (X \cap \texttt{both}(e))]$.

If $e \in \text{span}_{\mathcal{M}'_e}(Y)$ there would be a circuit D in $\mathcal{M}'_e$ such that $D - e \subseteq Y \subseteq C$, but since $C \cap [\texttt{mid}(e) \cup (X \cap \texttt{both}(e))] \neq \emptyset$ we get $D \subsetneq C$ which contradicts the minimality of C as circuit. Then, $e \in \text{span}_{\mathcal{M}'_e}[\texttt{mid}(e) \cup Y \cup (X \cap \texttt{both}(e))] \setminus \text{span}_{\mathcal{M}'_e} Y$, which is included in $\text{span}[\texttt{low}(e) \cup \texttt{mid}(e) \cup Y \cup (X \cap \texttt{both}(e))] \setminus \text{span}[\texttt{low}(e) \cup Y]$.

As $e \notin \texttt{high}(e) \cup \texttt{mid}(e) \cup \overline{Y} \cup (X \cap \texttt{both}(e))$ we conclude that $e \in \text{span}[\texttt{low}(e) \cup \texttt{mid}(e) \cup Y \cup (X \cap \texttt{both}(e))]$ and $e \in \text{cospan}[\texttt{high}(e) \cup \texttt{mid}(e) \cup \overline{Y} \cup (X \cap \texttt{both}(e))]$.

Choose two revelations $w^+, w^- \in \mathcal{R}(E, \mathcal{A})$ as follows:

$$w_f^+ \in \begin{cases} (L_e, U_e) \cap \mathcal{A}(f) & \text{if } f \in \texttt{mid}(e), \\ (-\infty, L_e] \cap \mathcal{A}(f) & \text{if } f \in Y \text{ or} \\ & f \in X \cap \texttt{both}(e), \\ [U_e, \infty) \cap \mathcal{A}(f) & \text{if } f \in \overline{Y}, \\ \mathcal{A}(e) \cap (L_e, U_e] & \text{if } f = e. \end{cases} \qquad w_f^- \in \begin{cases} [U_e, \infty) \cap \mathcal{A}(f) & \text{if } f \in X \cap \texttt{both}(e), \\ \mathcal{A}(e) \cap [L_e, U_e) & \text{if } f = e, \\ \{w_f^+\} & \text{otherwise.} \end{cases}$$

Note that w^+ and w^- only differ on $X \cap \texttt{both}(e)$ and e. As $e \in \text{span}[\texttt{low}(e) \cup \texttt{mid}(e) \cup Y \cup (X \cap \texttt{both}(e))]$ it is the unique heaviest element in a circuit in $(\mathcal{M}, w^+)$, therefore it is in no $(\mathcal{M}, w^+)$-basis. Similarly, since $e \in \text{cospan}[\texttt{high}(e) \cup \texttt{mid}(e) \cup \overline{Y} \cup (X \cap \texttt{both}(e))]$ it is the unique lightest element in a cocircuit in $(\mathcal{M}, w^-)$, hence it is in every $(\mathcal{M}, w^-)$-basis. To conclude suppose that there is a feasible query set F such that $X \cap F = \emptyset$, we then pick the revelation $\mathcal{B} \in \mathcal{R}(F, \mathcal{A})$ such that $\mathcal{B}(f) = \{w^+_f\} = \{w^-_f\}$ if $f \in F$, and $\mathcal{B}(f) = \mathcal{A}(f)$ otherwise.

Select any $\mathcal{B}$-basis T. As $w^+, w^- \in \mathcal{R}(E, \mathcal{B})$, T is both a w^+-basis and a w^--basis. By the previous paragraph, this implies that $e \notin T$ and $e \in T$ which is a contradiction. ◀

▶ **Lemma 28.** *The minimal feasible queries are the sets intersecting every witness set.*

Proof. Recall that a *clutter* is a family of sets such that no one is contained in another. The *blocker* of a clutter is the clutter of all minimal sets that intersect the first one. Thus, the minimal witness sets are the blocker of the minimal feasible queries. A basic result in packing and covering theory (see e.g., [15, Theorem 77.1]) states that the blocker of the blocker of a clutter is again the original clutter. The lemma follows from this fact. ◀

As we see below, minimal witness sets cannot be large: they can have at most 2 elements.

▶ **Corollary 29.** *Let X be a witness set such that $|X| \geq 2$. Then, there exists distinct $e, f \in X$ such that $\{e, f\}$ is a witness set.*

Proof. If $X \cap F \neq \emptyset$ for every feasible query set F then, by Lemma 27, there exists an uncertain $e \in X$, a circuit C in $\mathcal{M}'_e$ such that $e \in C$ and $C \cap [\texttt{mid}(e) \cup (X \cap \texttt{both}(e))] \neq \emptyset$.

If $C \cap (X \cap \texttt{both}(e)) = \emptyset$, then $C \cap \texttt{mid}(e) \neq \emptyset$ and by Lemma 27 we have that $\{e\} \cap F \neq \emptyset$ for every F feasible query set. Picking any $f \in X - e$ we conclude that $\{e, f\} \cap F \neq \emptyset$ for every feasible query set F. If $C \cap (X \cap \texttt{both}(e)) \neq \emptyset$, select any $g \in C \cap (X \cap \texttt{both}(e))$. Once again, Lemma 27 lets us conclude that $\{e, g\} \cap F \neq \emptyset$ for every F feasible query set. ◀

▶ **Definition 30.** *Let $\texttt{core} = \{e \in E \colon \{e\}$ is a witness set$\}$ and $\overline{\texttt{core}} = E \setminus \texttt{core}$. Define the graph $G^{wit} = (\overline{\texttt{core}}, E^{wit})$ where $ef \in E^{wit}$ if $\{e, f\}$ is a witness set.*

By Lemma 28, minimal feasible sets are exactly those sets containing all elements in `core` together with a vertex cover of G^{wit}. The following clean characterization of `core` follows directly from Lemma 27.

▶ **Lemma 31.** *Let e be an uncertain element. $e \in \texttt{core}$ if and only if there is a circuit C in $\mathcal{M}'_e$ such that $e \in C$ and $C \cap \texttt{mid}(e) \neq \emptyset$.*

We can turn the previous lemma into an algorithm that computes `core`. In order to do this we compute the connected component[4] of $\mathcal{M}'_e$ that contains e and check if it has non-empty intersection with $\texttt{mid}(e)$. We compute connected components with an algorithm due to Krogdhal [12] that takes polynomial time. Therefore, the previous procedure also takes polynomial time. In what follows we show that G^{wit} has a very nice structure.

▶ **Lemma 32.**

(i) *Let $e, f \in \overline{\texttt{core}}$ distinct. $ef \in E^{wit}$ if and only if $\mathcal{A}(e) = \mathcal{A}(f) = \{L_e, U_e\}$ with $L_e < U_e$ and there is a circuit C of $\mathcal{M}'_e$ such that $e, f \in C$.*

(ii) *The connected components of the graph G^{wit} are cliques.*

[4] Recall that e is connected to f in a matroid if and only if there is a circuit that contains e and f. A connected component is an equivalence class of this equivalence relation (see, eg. [14, Section 4.1]).

Proof.

(i) Suppose $ef \in E^{\text{wit}}$. Since $\{e, f\}$ is a witness set, $E \setminus \{e, f\}$ is not feasible. Let $\mathcal{B} \in \mathcal{R}(E \setminus \{e, f\}, \mathcal{A})$ be a revelation without $\mathcal{B}$-basis. Since $f \notin \texttt{core}$, we have that $E - f$ is feasible in $(\mathcal{M}, \mathcal{A})$ and by Lemma 20 it is also feasible in $(\mathcal{M}, \mathcal{B})$. If e was colored in $(\mathcal{M}, \mathcal{B})$ then, by Lemma 21 we would conclude that $(E - f) - e = E \setminus \{e, f\}$ is feasible in $(\mathcal{M}, \mathcal{B})$. But all elements in $E \setminus \{e, f\}$ are already certain in $(\mathcal{M}, \mathcal{B})$, from which we deduce that $\emptyset$ is feasible in $(\mathcal{M}, \mathcal{B})$, which is a contradiction.
We conclude that e is uncolored in $(\mathcal{M}, \mathcal{B})$. From this, we can use Lemma 25 to obtain a circuit C in $\mathcal{M}'_e$ such that $e \in C$ and $C \cap [\texttt{mid}^{\mathcal{A}}(e) \cup (\{e, f\} \cap \texttt{both}^{\mathcal{A}}(e))] \neq \emptyset$. Since $e \notin \texttt{core}$, we have by Lemma 31 that $C \cap \texttt{mid}(e) = \emptyset$. Then $C \cap \{e, f\} \cap \texttt{both}(e) \neq \emptyset$, consequently $e, f \in C$ and $f \in \texttt{both}(e)$. We can now use the same argument for f, concluding that $e \in \texttt{both}(f)$. The only way for $e \in \texttt{both}(f)$ and $f \in \texttt{both}(e)$ to occur at the same time is that $\mathcal{A}(e) = \mathcal{A}(f) = \{L_e, U_e\}$. Finally, since witness sets do not contain elements that are certain (this follows since minimal feasible sets only have uncertain elements, hence by removing certain elements from a witness set it would still intersect every feasible set) we must have $L_e < U_e$.
We now prove the converse. As e is an uncertain element such that there is a circuit C in $\mathcal{M}'_e$ and $f \in C \cap \{e, f\} \cap \texttt{both}(e)$ by Lemma 27 we conclude that $ef \in E^{\text{wit}}$.

(ii) We only need to show that whenever $ef, fg \in E^{\text{wit}}$ we also have $eg \in E^{\text{wit}}$. Suppose that $ef, fg \in E^{\text{wit}}$. By the previous item we have $\mathcal{A}(e) = \mathcal{A}(f) = \mathcal{A}(g) = \{L_e, U_e\}$, in particular $\mathcal{M}'_e = \mathcal{M}'_f = \mathcal{M}'_g \doteq \mathcal{M}'$. The previous item also allows us to conclude that there are two circuits C^1, C^2 in $\mathcal{M}'$ such that $e, f \in C^1$ and $f, g \in C^2$. Then e, f and g are in the same matroid connected component in $\mathcal{M}'$. Therefore there is a circuit C^3 in $\mathcal{M}'$ such that $e, g \in C^3$ and using the previous item we conclude that $eg \in E^{\text{wit}}$. ◀

We can test if $ef \in E^{\text{wit}}$ similarly to how we computed $\texttt{core}$: we start by considering elements with areas of size two (by checking if $(L_e, U_e) \cap \mathcal{A}(e) = \emptyset$ for every element e). If e and f have the same two-element uncertainty area, we check if they belong to the same connected component in $\mathcal{M}'_e = \mathcal{M}'_f$ using Krogdhal's algorithm [12].

▶ **Theorem 33.** *$X \subseteq E$ is a minimal feasible query if and only if $\texttt{core} \subseteq X$ and X intersects all but one element in each connected component of G^{wit}.*

Proof. By Lemma 28 and the definition of G^{wit} the minimal feasible queries X satisfies that $\texttt{core} \subseteq X$ and $X \cap \overline{\texttt{core}}$ is a minimal vertex cover of G^{wit}. Since every connected component is a clique, the minimal vertex covers of G^{wit} are exactly those sets containing all but one element in each connected component. ◀

▶ **Corollary 34.** *Let $c : E \to \mathbb{R}$ be any cost function. We can compute a minimum-cost feasible query in polynomial time.*

Proof. Computing $\texttt{core}$ and G^{wit} can be done in polynomial time and polynomial number of calls to the independence oracle of $\mathcal{M}$ or to minors of $\mathcal{M}$ (for example, $\mathcal{M}'_e$), since the oracle of independence of any minor of $\mathcal{M}$ can also be implemented using a polynomial number of calls to the oracle of $\mathcal{M}$. One can compute a minimum-cost minimal size feasible query F by simply returning a set containing $\texttt{core}$ and all but the most expensive element from each connected component of G^{wit}. If we allow negative costs, then the minimum-cost feasible query is F together with all the negative cost elements outside F. An efficient implementation is discussed in the full version of this paper. ◀

We finish our study with two special cases of the last theorem.

▶ **Corollary 35.** *Suppose that $\mathcal{A}(e)$ is an interval (of any type: open, closed, semiopen, trivial) for every element $e \in E$. Then the set S of all uncolored uncertain elements is the only minimum-size feasible query.*

▶ **Corollary 36.** *Let G be a connected graph such that for every $e \in E(G)$, $\mathcal{A}(e) = \{0, 1\}$. A set $F \subseteq E(G)$ is a minimal feasible query for the MST problem if and only if F contains all but one edge from each 2-connected component of G.*

References

1 Richard Bruce, Michael Hoffmann, Danny Krizanc, and Rajeev Raman. Efficient Update Strategies for Geometric Computing with Uncertainty. *Theory Comput. Syst.*, 38(4):411–423, 2005. doi:10.1007/s00224-004-1180-4.

2 George Charalambous and Michael Hoffmann. Verification Problem of Maximal Points under Uncertainty. In Thierry Lecroq and Laurent Mouchard, editors, *Combinatorial Algorithms - 24th International Workshop, IWOCA 2013, Rouen, France, July 10-12, 2013, Revised Selected Papers*, volume 8288 of *Lecture Notes in Computer Science*, pages 94–105. Springer, 2013. doi:10.1007/978-3-642-45278-9_9.

3 Thomas Erlebach and Michael Hoffmann. Query-competitive algorithms for computing with uncertainty. *Bulletin of EATCS*, 2(116), 2015.

4 Thomas Erlebach, Michael Hoffmann, and Frank Kammer. Query-competitive algorithms for cheapest set problems under uncertainty. *Theor. Comput. Sci.*, 613:51–64, 2016. doi:10.1016/j.tcs.2015.11.025.

5 Thomas Erlebach, Michael Hoffmann, Danny Krizanc, Matús Mihal'ák, and Rajeev Raman. Computing Minimum Spanning Trees with Uncertainty. In Susanne Albers and Pascal Weil, editors, *25th International Symposium on Theoretical Aspects of Computer Science*, volume 1 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 277–288, Dagstuhl, Germany, February 2008. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.STACS.2008.1358.

6 Tomás Feder, Rajeev Motwani, Liadan O'Callaghan, Chris Olston, and Rina Panigrahy. Computing shortest paths with uncertainty. *J. Algorithms*, 62(1):1–18, 2007. doi:10.1016/j.jalgor.2004.07.005.

7 Tomás Feder, Rajeev Motwani, Rina Panigrahy, Chris Olston, and Jennifer Widom. Computing the Median with Uncertainty. *SIAM J. Comput.*, 32(2):538–547, 2003. doi:10.1137/S0097539701395668.

8 Manoj Gupta, Yogish Sabharwal, and Sandeep Sen. The Update Complexity of Selection and Related Problems. *Theory Comput. Syst.*, 59(1):112–132, 2016. doi:10.1007/s00224-015-9664-y.

9 Simon Kahan. A Model for Data in Motion. In *Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing*, STOC '91, pages 265–277, New York, NY, USA, 1991. ACM. doi:10.1145/103418.103449.

10 Adam Kasperski and Pawel Zielinski. An approximation algorithm for interval data minmax regret combinatorial optimization problems. *Inf. Process. Lett.*, 97(5):177–180, 2006. doi:10.1016/j.ipl.2005.11.001.

11 Adam Kasperski and Pawel Zielinski. On combinatorial optimization problems on matroids with uncertain weights. *European Journal of Operational Research*, 177(2):851–864, 2007. doi:10.1016/j.ejor.2005.12.033.

12 Stein Krogdahl. The dependence graph for bases in matroids. *Discrete Mathematics*, 19(1):47–59, 1977. doi:10.1016/0012-365X(77)90118-2.

13 Nicole Megow, Julie Meißner, and Martin Skutella. Randomization Helps Computing a Minimum Spanning Tree under Uncertainty. *SIAM Journal on Computing*, 46(4):1217–1240, January 2017. doi:10.1137/16M1088375.

14 James G. Oxley. *Matroid theory*. Oxford University Press, 1992.

15 Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.

16 Hande Yaman, Oya Ekin Karasan, and Mustafa Ç. Pinar. The robust spanning tree problem with interval data. *Oper. Res. Lett.*, 29(1):31–40, 2001. doi:10.1016/S0167-6377(01)00078-5.

Short Proofs Are Hard to Find

Ian Mertz
University of Toronto, Canada
mertz@cs.toronto.edu

Toniann Pitassi
University of Toronto, Canada
Institute for Advanced Study, Princeton, NJ, USA
toni@cs.toronto.edu

Yuanhao Wei
Carnegie Mellon University, Pittsburgh, PA, USA
yuanhao1@cs.cmu.edu

Abstract

We obtain a streamlined proof of an important result by Alekhnovich and Razborov [2], showing that it is hard to automatize both tree-like and general Resolution. Under a different assumption than [2], our simplified proof gives improved bounds: we show under ETH that these proof systems are not automatizable in time $n^{f(n)}$, whenever $f(n) = o(\log^{1/7-\epsilon} \log n)$ for any $\epsilon > 0$. Previously non-automatizability was only known for $f(n) = O(1)$. Our proof also extends fairly straightforwardly to prove similar hardness results for PCR and Res(r).

2012 ACM Subject Classification Theory of computation → Proof complexity; Hardware → Theorem proving and SAT solving

Keywords and phrases automatizability, Resolution, SAT solvers, proof complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.84

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `http://www.cs.toronto.edu/~mertz/papers/mpw19.short_proofs_are_hard_to_find.pdf`.

Funding *Ian Mertz*: Research supported by NSERC.
Toniann Pitassi: Research supported by NSERC.
Yuanhao Wei: Work done while a student at University of Toronto. Research supported by NSERC.

Acknowledgements The authors thank Noah Fleming, Pravesh Kothari, Denis Pankratov, Robert Robere, Mika Göös, and Avi Wigderson for helpful conversations. We also thank Yijia Chen for providing more details on the construction in [14], and Pasin Manurangsi for giving feedback on the parameters in [13].

1 Introduction

Proof complexity first and foremost aims to understand, for a given propositional formula τ, how long of a proof is needed to verify that τ is unsatisfiable. To understand the expressiveness of a proof system, we need to understand what formulas can and cannot be efficiently proven in that system. However, for algorithmic applications where formulas often have fairly short proofs, what is perhaps more important than knowing the worst-case proof length of a given τ is actually *finding* proofs of τ. In particular, even if we're promised that τ has proofs of small size, say polynomial in the size of τ, can we hope to find one that's not too much larger?

This question, of finding optimal proofs in a given system, is known as *automatizability*, introduced by Bonet, Pitassi, and Raz [11]. A proof system $\mathcal{Q}$ is automatizable if there exists an algorithm which, given an unsatisfiable formula τ on n variables, returns a $\mathcal{Q}$-refutation of

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 84; pp. 84:1–84:16


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

τ in time $\mathsf{poly}(n, |\tau|, S)$ where $S := S_{\mathcal{Q}}(\tau)$ is the size of the shortest $\mathcal{Q}$-refutation of τ. Twenty years later no reasonable proof systems are known to be polynomially automatizable, and little is known even for the more general notion of f-automatizability, where the algorithm can run in time $f(n, |\tau|, S)$.

Understanding the automatizability of various proof systems is a major tool in algorithm design; two well known examples are SAT solvers, where the best algorithms are highly optimized version of the Resolution (Res) proof system (see e.g. [32]) and celebrated algorithmic versions of the Sum-of-Squares (SoS) proof system for approximation [37] and learning (see e.g. [38] for a survey on recent developments in this very active field of research). We especially draw attention to the question of automatizing Res. Resolution is a simple and fairly weak proof system, and yet Res proofs are the objects at the heart of the best known SAT solvers, with a long line of research connecting Res size to notions such as conflict driven clause learning and restarts [29, 30, 35, 42]. Automatizing Res is also key to the best known automated theorem provers for propositional and first order logics [17, 16]. Therefore, the tractability of finding short Res proofs lies at the heart of understanding the frontiers and limitations of SAT solving algorithms and automated theorem proving.

Despite the importance of automatizability for Res and other proof systems, our understanding of this question is limited at best. In terms of upper bounds, the best automatizing algorithm for Res runs in slightly subexponential time. In terms of lower bounds, until recently the main hardness result was the landmark paper of Alekhnovich and Razborov [2], who prove that under the assumption $\mathsf{FPT} \neq \mathsf{W[P]}$,[1] Res (as well as tree-like Res, denoted TreeRes) is not polynomially-automatizable. Using similar ideas, Galesi and Lauria [20] adapted Alekhnovich and Razborov's proof in order to obtain the same result for the Polynomial Calculus (PC) system, an extension of Res which is the proof complexity model for the Groebner basis algorithm [15].[2]

For all other well-studied practical systems almost nothing is known. To give a short list of other well-known proof systems used in algorithm design, we have Cutting Planes (CP), widely used for optimization algorithms (see e.g. [28]); Sherali-Adams (SA), which underlies a general family of linear programming algorithms [41]; and the aforementioned Sum-of-Squares (SoS)-based semi-definite programming algorithms. For these systems we have no extension of the argument of [2], and therefore no notable lower bounds on automatizability.

1.1 Our Contributions

Our motivation for this work is to adapt the techniques of [2], first to move past polynomial automatizability lower bounds for Res (and PC), and second to hopefully shed light on the automatizability of proof systems such as CP, SA, and SoS. The starting point of our contribution is in switching to the exponential time hypothesis (ETH) as opposed to the $\mathsf{FPT} \neq \mathsf{W[P]}$ assumption in [2, 20]. A central limitation in starting from the assumption that some problem has no FPT algorithm is that FPT algorithms run in time $f(k)n^{O(1)}$, and so the best lower bound one can get from such an assumption, without a careful analysis of f and the range of k, is $n^{\omega(1)}$. In the past decade a line of work by Chen and Lin [14] showed how to obtain fixed parameter lower bounds beyond $f(k)n^{O(1)}$ for gap versions of NP-hard

[1] The original result of [2] uses FPR, a randomized version of FPT, in place of FPT in the assumption. This was improved to the stated assumption by [19].

[2] While the most well-studied and widely used SAT solvers are based on Res, there have been some implementations that use the Groebner basis algorithm to utilize the more expressive power of PC, see e.g. [12].

problems, such as dominating set and hitting set, by starting not from an assumption about FPT but from ETH. Analyzing these reductions we can derive a hardness result for a *fixed* f and k, which allows us to go beyond the $n^{\omega(1)}$ barrier in [2, 20], albeit starting from the slightly stronger ETH assumption. We state our main theorem precisely now.

▶ **Theorem 1** (Main Theorem). *Let $\mathcal{Q} \in \{$Res, TreeRes, Nullsatz, PC, PCR$\}$. Assuming ETH holds $\mathcal{Q}$ is not n^f-automatizable for any $f = o(\log^{1/7-\varepsilon} \log n)$ (where $\varepsilon > 0$ is any constant).*

Equally important as extending the results of [2, 20] is our second goal, namely simplifying the presentation of the construction and proofs. Moving to the stronger ETH assumption allows us to change the central formula in a way that, while still using the core machinery of [2], leads to a conceptually simpler formula and proof. The basis of the formula in [2] is the monotone minimum circuit satisfying assignment (MMCSA) problem, which takes as input a poly-size monotone circuit. The natural encoding of their formula as a CNF formula requires extra variables to represent the internal gates of the monotone circuit, leading to many technicalities involved in proving a Res width lower bound, namely an indirect and highly redundant encoding of the circuit. Our proof starts from the hitting set problem, which is a special case of MMCSA where the circuit is a CNF. Since the formula is already a CNF, the input can be encoded directly as the formula rather than indirectly having variables for each of the gates, and as a result the upper and lower bound proofs in our paper are highly streamlined.

While we do start from a stronger assumption than [2, 20], there are few additional advantages to our new formula beyond presentation. First, going beyond superpolynomial hardness for Res allows us to obtain hardness results on the automatizability of Res(r), a proof system generalizing Res by allowing lines to be disjunctions of size r conjunctions. Prior to our paper nothing was known for Res(r) for any $r \geq 2$, and the formula from [2] would not be able to go past Res(r) for constant r.

▶ **Theorem 2** (Main Theorem for Res(r)). *Let $\mathcal{Q} =$ Res(r). Assuming ETH holds then for any $\varepsilon > 0$, $\mathcal{Q}$ is not $n^{f/\exp(r^2)}$-automatizable for any $f = o(\log^{\frac{1}{7}-\varepsilon} \log n)$ if $r \in O(\sqrt{\log f})$.*

Second, our technique has a direct, and in our view achievable, path to further improvement: if the reduction of [14] were to be improved to allow a lower bound against gap hitting set for larger parameters, it would immediately translate to a stronger non-automatizability result. We discuss this idea in detail in Section 6. Third, our results are also immediately strengthened if, instead of using ETH, one uses a slightly stronger assumption known as the *gap exponential time hypothesis* (GapETH), as introduced in [18, 27]. We formally define GapETH along with ETH in Section 2, but these results require no change in our formula nor our proofs. As with starting from [14] for our ETH results, the work required to use GapETH is analyzing a reduction of Chalermsook et al. [13], so we defer the results and analysis to to the full version of the paper.

1.2 Related Work

Table 1 lists the known results for Res and PC. An early result [1] shows that it is NP-hard to find proofs whose size is a constant factor of optimal, and this holds for *all* standard proof systems.

For stronger proof systems we have more lower bounds, although these bounds still only rule out polynomial automatizability and require cryptographic assumptions. Krajíček and Pudlák showed non-automatizability of the Extended Frege system under the hardness of

Proof system	Assumption	Result	Reference
all systems	$\mathsf{P} \neq \mathsf{NP}$	$\omega(1) \cdot n$	[1]
Res, TreeRes	$\mathsf{W[P]} \neq \mathsf{FPT}$	$n^{\omega(1)}$	[2]
Nullsatz, PC, PCR	$\mathsf{W[P]} \neq \mathsf{FPT}$	$n^{\omega(1)}$	[20]
Res, TreeRes, Nullsatz, PC, PCR	ETH	$n^{\Omega(\log^{1/7} \log n)}$	this work
Res(r)	ETH	$n^{\Omega(\log^{1/7} \log n / \exp(r^2))}$	this work

Figure 1 Lower bounds on automatizabillity of weak proof systems.

discrete log [25], with subsequent works proving the same lower bounds for Frege and AC^0-Frege under similar assumptions [10, 11]. Conceptually these more expressive classes should be harder to automatize because there exist many more short proofs than for say Res, but a nice upshot of these results is that they hold for a much weaker notion of automatizability, aptly named *weak automatizability*. Weak automatizability of a proof system $\mathcal{Q}$ only requires that the automatizing algorithm return a proof of τ in *some* proof system, so long as it's close in length to the shortest $\mathcal{Q}$-proof of τ.[3] Clearly hardness of weak automatizability implies hardness of automatizability, and hardness of weak automatizability is closely related to feasible interpolation [36], which was the tool used in the results listed above.

Turning to upper bounds, there are a class of *width/degree* based automatizability algorithms for Res, PC, SA, and SoS. The width of a Res refutation is the maximal number of literals appearing in any line of the refutation, and the width of a CNF formula τ, denoted $w(\tau)$, is the minimum width of any Res refutation refuting τ. It is not hard to see that exhaustive search allows us to find a Res refutation for τ in time $n^{O(w(\tau))}$ [8]. A non-trivial fact is that the same upper bound holds for PC (due to the Groebner basis paper of Clegg, Impagliazzo, and Edmonds [15]), SA [40], and SoS [34, 26], where the degree of the polynomials appearing in the proofs is used in place of width. These algorithms are known to be tight for width/degree based automatizability, as there exist tautologies τ with proof size $S(\tau) = n^{\Omega(d)}$ for Res, PC, SA, and SoS[4] [6].

A groundbreaking work of Ben-Sasson and Wigderson [9] showed that $w(\tau) \leq \log S(\tau)$ for the special case of TreeRes and $w(\tau) \leq \sqrt{n \log S(\tau)}$ for general Res. Combined with the $n^{O(w(\tau))}$ upper bound for both systems gives automatizability for TreeRes and Res in time $n^{O(\log S(\tau))}$ and $n^{O(\sqrt{n \log S(\tau)})}$, respectively. Perhaps even more surprisingly, a result of [15] gives the same degree/size tradeoff for PC as [9] gave for Res; $d(\tau) \leq \sqrt{n \log S(\tau)}$ for the case of PC, and $d(\tau) \leq \log S(\tau)$ for a static version of PC called *Nullstellensatz* (Nullsatz).[5] Combining these degree bounds with the degree based algorithms gives automatizability for Nullsatz and PC in time $n^{O(\log S(\tau))}$ and $n^{O(\sqrt{n \log S(\tau)})}$, respectively. While these upper bounds are very strong for TreeRes and Nullsatz, for Res and PC they are still weakly exponential, and the results of [9, 15] are tight. Thus non-width/degree based techniques are needed to improve these upper bounds.

[3] This can be seen as analogous to the two notions of of learning, proper versus nonproper, where the former is required to produce a hypothesis from the original concept class, whereas the latter may produce any hypothesis.

[4] The degree-automatizability of SoS is not established definitely due to the bit-complexity of the underlying polynomials, which can be exponential [33].

[5] This result of [15] actually preceded [9].

1.2.1 Recent Developments

In a recent breakthrough paper, Atserias and Müller [4] resolve the automatizability of (general) Resolution. In particular they show that it is NP-hard to distinguish whether τ has Res refutations of size $n^{1+\epsilon}$ or none of size $2^{n^{1/(2+\epsilon)}}$ for any $\epsilon > 0$, which implies that assuming ETH, Res is not $2^{n^{\delta}}$ automatizable for any $\delta < \frac{1}{2}$. Because of the quasipolynomial upper bounds on automatizability for the other systems that we study here (tree-like Resolution, and the Polynomial Calculus) our results as well as [2] are incomparable with [4]. Thus our results (and technique) are still at the frontier for all other systems discussed.

2 Preliminaries

Let $\tau = \{C_1, C_2, \ldots, C_m\}$ be an unsatisfiable CNF formula over $X = \{x_1 \ldots x_n\}$. We denote by $|\tau|$ the *size* of τ, and likewise for a proof π refuting τ let $|\pi|$ denote the size of π. For a proof system $\mathcal{Q}$ let $S := S_{\mathcal{Q}}(\tau)$ be the size of the shortest $\mathcal{Q}$-proof refuting τ. A proof system $\mathcal{Q}$ is said to be $f(n, |\tau|, S)$-*automatizable* if there exists an algorithm A such that for every unsatisfiable τ A runs in time $f(n, |\tau|, S)$ and outputs a valid $\mathcal{Q}$-proof refuting τ. A proof system $\mathcal{Q}'$ *p-simulates* $\mathcal{Q}$ if for every $\mathcal{Q}$-proof π refuting τ there is a corresponding $\mathcal{Q}'$-proof π' refuting τ such that $|\pi'| = |\pi|^{O(1)}$.

A *Resolution* (Res) refutation of τ is a sequence of clauses $\pi = \{D_1, D_2, \ldots, D_S\}$ such that $D_S = \emptyset$, and each line D_i is either some initial clause $C_j \in \tau$ or is derived from two previous lines using the *resolution rule*: from $(E \vee x)$, $(F \vee \overline{x})$ we derive $(E \vee F)$, where $x \in X$, E and F are clauses, and $E \vee F$ is their disjunction with repeated literals removed. We can view a Res proof π as a directed acyclic graph with a unique clause D_i at every vertex, with initial clauses $C_j \in \tau$ at the leaves, $\emptyset$ at the root, and having an edge from D_i to D_j if D_i was used to derive D_j. With this view, a TreeRes refutation requires that all non-leaf vertices of the underlying graph have outdegree 1 (so the underlying graph of any TreeRes proof is tree-like).

Given a Res or TreeRes refutation $\pi = \{D_1, D_2, \ldots, D_S\}$, the size of π is the number of lines in π, in this case S. The *width* of a clause D_i is the number of literals in it, and the width of π is the maximum width of a clause in the proof. We denote the width of a clause D_i or proof π by $w(D_i)$ and $w(\pi)$, respectively. Clearly Res can p-simulate TreeRes with respect to size and width, as every TreeRes-proof is also a Res-proof.

An *r-Resolution* (Res(r)) refutation[6] is similar to a Res refutation, but each line D_i is an r-DNF instead of a clause, and the resolution rule is adapted as follows: from $(E \vee (\vee_{j \in J} x_j)), (F \vee (\wedge_{j \in J} \overline{x_j}))$ we derive $(E \vee F)$, where $J \subseteq [n]$ such that $|J| \leq r$, E and F are r-DNFs, and $E \vee F$ is their disjunction with repeated conjunctions removed (note that $\vee_{j \in J} x_j$ is a DNF with $|J|$ terms while $\wedge_{j \in J} \overline{x_j}$ is a single term). Note that Res(1) = Res. The size of a Res(r) proof is the number of r-disjunctions in it. (See [39] for more details.)

An *algebraic proof system* for refuting CNF $\tau = \{C_1 \ldots C_{m'}\}$ over variable set X is a proof system where each of the clauses C_i is converted into a polynomial equality or inequality P_i over X, such that any assignment of all x_j to $\{0,1\}^n$ satisfies C_i iff it satisfies P_i. For this paper the conversion is done is by sending every positive literal x_j to $(1 - x_j)$ and every negative literal $\overline{x_j}$ to x_j, and P_i is satisfied if the product of all converted literals in C_i is 0. For example, the clause $C_i = x_1 \vee \overline{x_2} \vee x_3$ is converted to $P_i = (1 - x_1)(x_2)(1 - x_3) = 0$.

[6] This class is more commonly called k-Resolution, or Res(k), in proof complexity literature, but the parameter k already plays a central role in our paper.

In addition, we add the equations $x_j^2 - x_j = 0$ for all $j \leq n$. Let the resulting $m = m' + n$ equations corresponding to τ be denoted by $\mathcal{P} = \{P_1, \ldots, P_m\}$. Since every P_i is of the form $p_i = 0$ we use P_i to refer to p_i.

The *Nullstellensatz* (Nullsatz) refutation system [7] is an algebraic proof system that uses Hilbert's Nullstellensatz as a certificate of unsatisfiablility. A Nullsatz proof (over a field $\mathbb{F}$) of τ is a set of polynomials $Q_1, \ldots, Q_m$ such that $\sum_i P_i Q_i$ is the formal polynomial "1". Note that this contradicts the statement that there exists an assignment such that $P_i = 0$ for all i. The size of a Nullsatz refutation π is the sum over all $i \in [m]$ of the number of monomials in the expansion of the term $P_i Q_i$, while the *degree* of the refutation is the maximum degree $\deg(P_i Q_i)$ over all $i \in [m]$. It is known that Nullsatz p-simulates TreeRes.

The *Polynomial Calculus* (PC) system is a dynamic version of Nullsatz [15], where the lines of a PC proof π are all polynomials $Q_1, Q_2, \ldots, Q_S$. The lines Q_i can be any of the initial polynomial equations $\mathcal{P}$ or can be derived from previous lines by the following rules: (1) from Q_i we can derive $x_j Q_i$ or $(1 - x_j)Q_i$ for any variable x_j; (2) from Q_i, Q_j we can derive $aQ_i + bQ_j$ for any $a, b \in \mathbb{R}$. As with Nullsatz the final line Q_S is the formal polynomial "1". Similarly to Nullsatz the degree of a PC proof π is the maximal degree of any line Q_i, and the size of π is the total number of monomials in the refutation, where multiple occurrences of the same monomial are counted for each occurrence. PC trivially p-simulates Nullsatz and the simulation is degree-preserving.

The PCR system is a simple modification to the PC proof system so that it can p-simulate Res proofs with respect to size. For PCR, polynomials are allowed to use additional variables $\overline{x}_1, \ldots, \overline{x}_n$ and axioms of the form $1 - \overline{x}_j - x_j = 0$ for all $j \in [n]$. Furthermore all terms $(1 - x_j)$ in the input polynomials in $\mathcal{P}$ are replaced by the variables $\overline{x_j}$. Intuitively although the variables x_j and $\overline{x_j}$ are distinct they stand for the negations of one another, which is enforced by the new axiom corresponding to x_j. It is not hard to see that PCR can now p-simulate Res with respect to size.

Let $\mathcal{S} = \{S_1, \ldots, S_n\}$ be a collection of non-empty sets S_j over $[n]$. A *hitting set* $H \subseteq [n]$ is a set of elements such that $H \cap S_j \neq \emptyset$ for all $j \in [n]$. Let $\gamma(\mathcal{S})$ be the size of the smallest hitting set for $\mathcal{S}$. The *gap hitting set problem* is the task of distinguishing, on input $(\mathcal{S}, k, hk)$, the following two cases: (1) $\gamma(\mathcal{S}) \leq k$; (2) $\gamma(\mathcal{S}) > hk$.

▶ **Definition 3.** *The Exponential Time Hypothesis (ETH) states [23] that for sufficiently large m and n, no algorithm running in time $2^{o(n)}$ can decide, for given CNF τ with m clauses and n variables, whether all m clauses of τ are satisfiable or not. The Gap Exponential Time Hypothesis (GapETH) states [18, 27] that for sufficiently large m and n, no algorithm running in time $2^{o(n)}$ can decide, for given CNF τ with m clauses and n variables and any constant $\epsilon \in (0, 1)$, whether all m clauses of τ are satisfiable or if at most $(1 - \epsilon)m$ of the clauses are satisfiable.*

We state the following hardness results for the hitting set problem under ETH, which can be deduced from a construction by Chen and Lin [14]. The actual lemma we prove is slightly more technical but actually slightly stronger than the one we state here. A full discussion and proof of the lemma is included in the full version of the paper[7].

▶ **Lemma 4** (ETH-Hardness of Hitting Set)**.** *Assuming ETH, for sufficiently large n and $k = O(\log^{1/7-\epsilon} \log n)$ no algorithm can solve the gap hitting set problem $(\mathcal{S}, k, k^2)$ in time $n^{o(k)}$.*

[7] A similar result holds for GapETH, which can be deduced from recent work of Chalermsook et al [13]. See the full version for more details.

Consider a set $A \subseteq \{0,1\}^m$ of m-bit strings such that $|A| = m$. We say that A is (m,k)-*universal* if for every subset $J \subseteq [m]$ of up to k distinct positions in $[m]$, the projection $A|_J$ (restricting the strings in A to these positions) contains all possible $2^{|J|}$ binary strings of length $|J|$. Observe that we can take the dual of the set A in the following sense: if $A = \{a_1, \ldots, a_m\}$, and let $B \subseteq \{0,1\}^m$ be the set of all strings b_j for $j \in [m]$ such that the ith bit of b_j is the jth bit of a_i. Another way to think about B is taking the strings of A to be the columns of an $m \times m$ matrix and letting B be the columns of that matrix's transpose. We say A is (m,k)-*dual-universal* if B is (m,k)-universal. Equivalently A is (m,k)-dual-universal if for every ordered subset $I \subseteq A$ of up to k distinct strings in A and for every string $s \in \{0,1\}^{|I|}$, there exists some position $j \in [m]$ such that s is the string formed by concatenating the jth bit of all strings in I in order. The existence of efficiently constructible $(m, \log m/4)$-universal sets is known. It is also known that there exist efficiently constructible sets that are both $(m, \log m/4)$-universal *and* $(m, \log m/4)$-dual-universal. For a concrete example, [2] uses the *Paley graph* G_m on m vertices [8] For the rest of the paper we will fix an arbitrary A that is efficiently computable and is both $(m, \log m/4)$-universal and $(m, \log m/4)$-dual-universal.

3 Main reduction

We first state our main lemma from which Theorem 1 is easily proven.

▶ **Lemma 5.** *Let $\mathcal{Q} \in \{$Res, TreeRes, Nullsatz, PC, PCR$\}$. For sufficiently large n and $k = O(\log^{1/3} n)$, let $(\mathcal{S}, k, k^2)$ be an instance of the gap hitting set problem over $[n]$. Then there exists an unsatisfiable CNF $\tau_{\mathcal{S}}$ which can be computed in time $n^{O(1)}$ such that the following two properties hold*

(i) *if $\gamma(\mathcal{S}) \leq k$ then $S_{\mathcal{Q}}(\tau_{\mathcal{S}}) \leq n^{O(1)}$;*
(ii) *if $\gamma(\mathcal{S}) > k^2$ then $S_{\mathcal{Q}}(\tau_{\mathcal{S}}) \geq n^{\Omega(k)}$.*

Proof of Theorem 1. Assuming that $\mathcal{Q}$ is n^f automatizable for some $f(n) = o(\log^{1/7-\epsilon} \log n)$ for $\epsilon > 0$, we describe an efficient algorithm for the gap hitting set problem. Given an instance $(\mathcal{S}, k, k^2)$ of the gap hitting set problem over $[n]$, with n sufficiently large and $k = O(\log^{1/7-\epsilon} \log n)$, we generate the CNF $\tau_{\mathcal{S}}$, and simulate the automatizing algorithm on $\tau_{\mathcal{S}}$ for $n^{O(f)}$ timesteps. If the automatizing algorithm outputs a legal $\mathcal{Q}$ refutation of $\tau_{\mathcal{S}}$ within the allotted time, then we output "$\gamma(\mathcal{S}) \leq k$" and otherwise output "$\gamma(\mathcal{S}) > k^2$". Because $f = o(k)$ the correctness is guaranteed by Lemma 5. Thus we can decide the gap hitting set problem in time $n^{O(f)} = n^{o(k)}$, which by Lemma 4, contradicts ETH. ◀

The rest of the paper is devoted to the proof of Lemma 5. In this section we give the reduction $\tau_{\mathcal{S}}$, and prove the upper and lower bounds needed for the case of Res in Sections 4 and 5. This also gives the upper bound for PC and Res(r); the lower bounds are deferred to the full paper. We briefly note that the strength of the result in Theorem 1 relies solely on the largest value we can set k to. We choose $k = O(\log^{1/7-\epsilon} \log n)$ because this is the largest value we can use and still get a contradiction with Lemma 4, but for Lemma 5 to hold we can tolerate up to $k = O(\log^{1/3} \log n)$, meaning that if the reduction in [14] were

[8] Many examples of universal sets (including the Paley graph construction) are discussed in [24], as well as [31, 3]. Alternate constructions use properties such as k-wise independent sample spaces and linear codes, and counting arguments for different parameter regimes exist. Notably the Paley construction fulfills our four essential properties of being small (of size m), polytime constructible, $(m, \log m/4)$-universal, and $(m, \log m/4)$-dual-universal.

improved, a stronger version of Theorem 1 would immediately follow. Likewise, starting from the GapETH assumption we could use a stronger version of Lemma 4 and immediately get the stronger result claimed in Section 1 (see the full version of the paper).

Hereafter, fix $k = O(\log^{1/7-\epsilon} \log n)$ and define $m := n^{1/k}$. Observe that $k \log m = \log n$ and $k^2 < \log m$ for large enough n. In what follows we will abuse notation and x_i, y_j will denote a tuple of Boolean variables (rather than a single Boolean variable). The tuple size of x_i, y_j will be clear from context, but generally x_i will be a $O(\log m)$-tuple and y_j will be a $O(\log n)$-tuple. Additionally $\vec{x} = x_1, \ldots, x_n$, $\vec{y} = y_1, \ldots, y_m$ will denote vectors of the tuples x_i and y_j. α_i and β_j will denote a 0/1 assignment to the tuples x_i and y_j respectively, and $\vec{\alpha}, \vec{\beta}$ will each denote a 0/1 assignment to the vector of tuples $\vec{x}, \vec{y}$ respectively.

3.1 The Formula $\psi_{\mathcal{S}}$

Given a hitting set instance $\mathcal{S}$ we will define an unsatisfiable formula $\psi_{\mathcal{S}}$. Recall that A is a set of m-bit strings such that $|A| = m$ and A is both $(m, (\log m)/4)$-universal and $(m, (\log m)/4)$-dual-universal. We also define the *characteristic vector* of a set $S \subseteq [n]$ to be the binary vector $s \in \{0,1\}^n$ such that $s_i = 0$ for all $i \notin S$ and $s_i = 1$ for all $i \in S$.

The formula $\psi_{\mathcal{S}}$ will have variables $\vec{x}$ and $\vec{y}$ that will respectively encode n-by-m matrices M and N. The variables of $\vec{x}$ will define M such that each of the n rows of M is some vector in A, and the variables $\vec{y}$ will define N such that each of the m columns of N is the characteristic vector for some set S from the hitting set instance $\mathcal{S}$. In particular, x_i will indicate a vector in A to serve as the ith row of M, while y_j will indicate a set in $\mathcal{S}$ whose characteristic vector will serve as the jth column of N, with each x_i and y_j being chosen separately. For the remainder of the section, we restrict our attention to matrices M and N defined this way. We say that M and N *intersect* if $M[i,j] = N[i,j] = 1$ for some pair (i,j). $\psi_{\mathcal{S}}$ will be defined so that it is falsified whenever M and N intersect and satisfied otherwise.

Notice that when some column of M is the characteristic vector of a hitting set, $\psi_{\mathcal{S}}$ is falsified because there is no way to pick the corresponding column in N so that the two columns do not intersect. Conversely, if none of the columns in M represent a hitting set, then there is always a way to pick N so that $\psi_{\mathcal{S}}$ is satisfied (for each column we simply pick the set that was not hit). Therefore proving that $\psi_{\mathcal{S}}$ is unsatisfiable boils down to proving that for any choice of M, some column of M represents a hitting set.

▷ Claim 6. $\psi_{\mathcal{S}}$ is unsatisfiable when $\gamma(\mathcal{S}) \leq \frac{\log m}{4}$.

Proof sketch. Let H be any hitting set of size at most $\frac{\log m}{4}$, which we interpret of as a set of row indices into M. By the $(m, (\log m)/4)$-dual-universality of A, any set I of at most $(\log m)/4$ strings from A has a location such that all the strings in I contain a 1 at that location.[9] Since rows of M are strings in A, taking $I = H$ there must exist a column j^* such that $M[i, j^*] = 1$ for every $i \in H$. Because H is a hitting set and the jth column of N is the indicator vector of a set $S \in \mathcal{S}$, there must be some $i^* \in H$ such that $N[i^*, j^*] = 1$, and so M and N intersect at (i^*, j^*). ◁

Next, we define the formula more formally. The variables of $\psi_{\mathcal{S}}$ are $\vec{x} = \{x_i \mid i \in [n]\}$ where x_i is a tuple of $\log m$ boolean variables, and $\vec{y} = \{y_j \mid j \in [m]\}$ where y_j is a tuple of $\log n$ boolean variables. Given an assignment $\vec{\alpha} = \{\alpha_i \mid i \in [n]\}$ to the $\vec{x}$-variables, $\vec{\alpha}$

[9] We do not require that the rows of M are *distinct* rows of A, but because we are only looking for a location with a 1 for every row this does not pose an issue. In fact we only ever use the universal and dual universal properties to search for a location with either all 0 or all 1, where repetition doesn't break the universal properties we need.

encodes an n-by-m matrix $M_{\vec{\alpha}}$ where the i-th row of $M_{\vec{\alpha}}$ equals $a_{\alpha_i} \in A$ (interpreting α_i as an index in $[m]$). Similarly given an assignment $\vec{\beta} = \{\beta_j \mid j \in [m]\}$ to the $\vec{y}$-variables, $\vec{\beta}$ encodes an n-by-m matrix $N_{\vec{\beta}}$, where column j is the characteristic vector of the set $S_{\beta_j} \in \mathcal{S}$ (interpreting β_j as an index in $[n]$). We will sometimes write $M_{\vec{\alpha}}[i,j]$ as $M_{\alpha_i}[i,j]$ to stress that the ith row of $M_{\vec{\alpha}}$ is determined by α_i. Similarly, we will sometimes write $N_{\vec{\beta}}[i,j]$ as $N_{\beta_j}[i,j]$.

Lastly, we formally define the clauses in $\psi_{\mathcal{S}}$ so that it is falsified whenever $M_{\vec{\alpha}}$ and $N_{\vec{\beta}}$ intersect and satisfied otherwise.

▶ **Definition 7.** *For every $i \in [n]$ and $j \in [m]$, and for every pair of values $\alpha_i \in \{0,1\}^{\log m}$, $\beta_j \in \{0,1\}^{\log n}$ such that $M_{\alpha_i}[i,j] = 1$ and $N_{\beta_j}[i,j] = 1$, we have the clause $\overline{x_i^{\alpha_i} \wedge y_j^{\beta_j}}$ where $x_i^{\alpha_i} = \wedge_{t \in [n]} (x_i)_t^{(\alpha_i)_t}$ is the conjunction of all variables in x_i, each of which occurs positively when the corresponding bit of α_i is 1 and negatively when the corresponding bit of α_i is 0 (we define $y_j^{\beta_j}$ in the same way). This axiom is falsified iff x_i is assigned value α_i and y_j is assigned value β_j.*

This formula has the property we want because if $M_{\vec{\alpha}}$ and $N_{\vec{\beta}}$ intersect at some location i,j, then the axiom $\overline{x_i^{\alpha_i} \wedge y_j^{\beta_j}}$ exists in $\psi_{\mathcal{S}}$ and would be falsified. Conversely, if $\psi_{\mathcal{S}}$ is falsified, then some axiom $\overline{x_i^{\alpha_i} \wedge y_j^{\beta_j}}$ is falsified, which means $M_{\vec{\alpha}}[i,j] = N_{\vec{\beta}}[i,j] = 1$.

It is easy to check that the number of variables in $\psi_{\mathcal{S}}$ is $n \log m + m \log n$. The number of clauses is at most $n^2 m^2$, since for each $i \in [n]$ and $j \in [m]$, each of the mn possible assignments to (x_i, y_j) adds at most one clause to $\psi_{\mathcal{S}}$.

3.2 Redundantly Encoding $\psi_{\mathcal{S}}$

In order to prove our result we will need a way of proving both upper and lower bounds on $S_{\mathcal{Q}}(\psi_{\mathcal{S}})$, but it turns out that the lower bounds are difficult to prove if we use $\psi_{\mathcal{S}}$ as is. Thus, we will employ a standard trick in proof complexity, which is to redundantly encode the variables in the formula; more specifically we follow [2] and redundantly code blocks of variables, namely each row and column, using error-correcting codes. It is interesting to note that for our formulas, we are unable to prove even width lower bounds without the redundant encoding. In contrast, most proof complexity applications use this trick solely for the purpose of reducing size lower bounds to width lower bounds.

▶ **Definition 8.** *For $q, r, s \in \mathbb{N}$, a (q,r,s)-code is a total function f from $\{0,1\}^q$ to $\{0,1\}^r$ with the property that for any $\rho \in \{0,1,*\}^q$ such that ρ fixes at most s values to $\{0,1\}$, $f|_\rho$ is surjective on $\{0,1\}^r$. Efficiently computable constructions using linear codes are known for any $r, q = 6r, s = 2r$ (see e.g. [2]). We say that f is r-surjective.*

Let $f_x : \{0,1\}^{6 \log m} \to [m]$ be a $(6 \log m, \log m, 2 \log m)$-code and let $f_y : \{0,1\}^{6 \log n} \to [n]$ be a $(6 \log n, \log n, 2 \log n)$-code. We will have a vector $x_i \in \{0,1\}^{6 \log m}$ for each $i \in [n]$ and a vector $y_j \in \{0,1\}^{6 \log n}$ for each $j \in [m]$. Given an assignment $\vec{\alpha}$ to all of the $\vec{x}$-variables, we will associate with $\vec{\alpha}$ an n-by-m matrix $M_{\vec{\alpha}}$, where the ith row of $M_{\vec{\alpha}}$ will be the vector $a_{f_x(\alpha_i)} \in A$. Similarly given an assignment $\vec{\beta}$ to all of the $\vec{y}$-variables, we will associate with $\vec{\beta}$ an n-by-m matrix $N_{\vec{\beta}}$, where column j is the characteristic vector corresponding to the set $S_{f_y(\beta_j)} \in \mathcal{S}$ In other words, $N_{\vec{\beta}}[i,j]$ is 1 if and only if set $S_{f_y(\beta_j)}$ contains element i.

We now define our unsatisfiable CNF $\tau_{\mathcal{S}}$ in the same way as $\psi_{\mathcal{S}}$ using these redundant encodings. Note that it is unsatisfiable for exactly the same reason as stated before.

▶ **Definition 9.** *The clauses of $\tau_{\mathcal{S}}$ are defined as follows. For every $i \in [n], j \in [m]$ and for every pair of assignments (α_i, β_j) to (x_i, y_j) such that $M_{\alpha_i}[i,j] = 1$ and $N_{\beta_j}[i,j] = 1$, we have the clause $\overline{x_i^{\alpha_i} \wedge y_j^{\beta_j}}$.*

In the redundant encoding we have $n \cdot 6 \log m$ x-variables and $m \cdot 6 \log n$ y-variables, for a total of $O(n \log m)$ variables when $m = n^{1/k} \ll n$. The number of clauses in $\tau_{\mathcal{S}}$ is at most $n^7 m^7$, since for each $i \in [n]$ and $j \in [m]$, each of the $m^6 n^6$ possible assignments to (x_i, y_j) adds at most one clause to $\tau_{\mathcal{S}}$.

The following two lemmas, which will be the focus of the rest of the paper, give tight upper and lower bounds on $S_{\mathcal{Q}}(\tau_{\mathcal{S}})$ as a function of $\gamma(\mathcal{S})$. Since we can clearly construct $\tau_{\mathcal{S}}$ in time polynomial in n, proving these two lemmas is all we need to finish Lemma 5.

▶ **Lemma 10.** *For sufficiently large n and $k = O(\log^{1/3} n)$, let $(\mathcal{S}, k, k^2)$ be an instance of the gap hitting set problem over $[n]$ such that $\gamma(\mathcal{S}) \leq k$. Then $S_{\mathcal{Q}}(\tau_{\mathcal{S}}) \leq n^{O(1)}$ for any $\mathcal{Q} \in \{$Res, TreeRes, Nullsatz, PC, PCR$\}$.*

▶ **Lemma 11.** *For sufficiently large n and $k = O(\log^{1/3} n)$, let $(\mathcal{S}, k, k^2)$ be an instance of the gap hitting set problem over $[n]$ such that $\gamma(\mathcal{S}) > k^2$. Then $S_{\mathcal{Q}}(\tau_{\mathcal{S}}) \geq n^{\Omega(k)}$ for any $\mathcal{Q} \in \{$Res, TreeRes, Nullsatz, PC, PCR$\}$.*

It may be instructive to note that both the upper and lower bounds are exactly $n^{\Theta(\gamma(\mathcal{S})/k)} = m^{\Theta(\gamma(\mathcal{S}))}$, which is polynomial in the number of distinct assignments to $\alpha_1 \ldots \alpha_{\gamma(\mathcal{S})}$, assuming without loss of generality that the minimum hitting set of $\mathcal{S}$ is the first $\gamma(\mathcal{S})$ elements $\{1 \ldots \gamma(\mathcal{S})\} \subseteq [n]$. In Sections 4 and 5 we show how these assignments exactly characterize the shortest proof of τ.

4 Upper bound in TreeRes

In this section we prove Lemma 10. Note that it suffices to give an upper bound in TreeRes since all of the other proof systems can p-simulate TreeRes.

Proof of Lemma 10. The proof is just a formalization of the argument given in the proof of Claim 6. Using the well-known equivalence between TreeRes proofs and decision trees, it suffices to give a decision tree solving the search problem for $\tau_{\mathcal{S}}$; that is, a decision tree (over the underlying variables of $\tau_{\mathcal{S}}$), where every leaf l is labelled with a clause of $\tau_{\mathcal{S}}$ that is falsified by the partial assignment that labels the path to l.

We will first show that if $\gamma(\mathcal{S}) \leq k$, then there is a height $2 \log n$ decision tree (and therefore size n^2) for the unencoded formula $\psi_{\mathcal{S}}$. Since $\gamma(\mathcal{S}) \leq k$, assume without loss of generality that $H = \{1, \ldots, k\}$ is a valid hitting set for $\mathcal{S}$. The decision tree for $\psi_{\mathcal{S}}$ consists of two phases. First, the decision tree will branch on all of the Boolean variables in $x_1, \ldots, x_k$. This will result in a full binary tree, call it T, of depth $k \log m$. In the second phase, at each leaf vertex of T we will query all of the variables of some y_j variable, where the choice of y_j will be a function of the path taken in T.

Consider some path in T leading to leaf $l_{\vec{\alpha}}$, corresponding to the assignment $\vec{\alpha} = \alpha_1, \ldots \alpha_k$ for $x_1, \ldots, x_k$. The assignment $\vec{\alpha}$ corresponds to an ordered set of strings $I \subseteq A$, where $|I| \leq k$. Since $k \in O(\log^{1/3} n)$ and $m = n^{1/k}$, $k \leq \frac{\log m}{4}$ for large n. By the $(m, \log m/4)$-dual-universal property of A there is some $j \in [m]$ such that I restricted to position j is all 1's, and thus $M_{\vec{\alpha}}[i,j] = 1$ for all $i \in [k]$. In the second phase, at this leaf vertex $l_{\vec{\alpha}}$ of T we will then query all of the Boolean variables in y_j. Let β_j be one partial assignment to

these variables and consider the path labelled by $\vec{\alpha}\beta_j$ leading to the leaf vertex $l_{\vec{\alpha}\beta_j}$. Since $\{1,\dots,k\}$ is a hitting set for $\mathcal{S}$ we are guaranteed that $N_{\vec{\beta}_j}[i,j]=1$ for at least one $i\in[k]$, and since $M_{\vec{\alpha}}[i,j]=1$ for all $i\in[k]$, one of the clauses in $\tau_{\mathcal{S}}$ must be violated by the partial assignment $\vec{\alpha},\beta_j$, so we label $l_{\vec{\alpha}\beta_j}$ with any such clause. The resulting decision tree thus solves the search problem associated with $\psi_{\mathcal{S}}$ and has height $k\log m+\log n=2\log n$.

The decision tree for the redundant version $\tau_{\mathcal{S}}$ is essentially the same but instead we query the redundant encodings of the variables. First, we query $x_1,\dots,x_k$, resulting in a full binary tree of height $k\cdot 6\log m$, and then, we query a particular y_j (depending on the path taken in T), which is $6\log n$ variables, and thus the height is $k\cdot 6\log m+6\log n=12\log n$. ◀

5 Nonautomatizability of Res and TreeRes

In this section we prove Lemma 11 for the case of $\mathcal{Q}=$ Res, which implies the result for TreeRes as well. We begin by proving a *wide clause lemma* for $\tau_{\mathcal{S}}$, which alone is enough to prove lower bounds for TreeRes (using the size-width relationship for TreeRes due to Ben-Sasson and Wigderson [9]); for general Res, we apply a standard application of random restrictions to reduce to width.

Our notion of "wide" will be a bit richer than the usual definition. For a clause D, let $I_0(D)$ be the set of all $i\in[n]$ for which there are at least $\log m$ literals in D that correspond to variables from x_i. Likewise let $J_0(D)$ be the set of all $j\in[m]$ for which there are at least $\log n$ literals in D that correspond to variables from y_j.

▶ **Lemma 12** (Wide Clause Lemma). *For sufficiently large n, if $\gamma(\mathcal{S})>k^2$ and f_x (f_y) is $\log m$-surjective ($\log n$-surjective, respectively), then for any Res refutation π refuting $\tau_{\mathcal{S}}$ there exists a clause $D\in\pi$ such that $|I_0(D)|\geq k^2$ or $|J_0(D)|\geq k$.*

Proof. We follow the *prover-delayer game* of [36, 5] in the style of [6]. The width-w game on an unsatisfiable formula τ is played between a Delayer, who is asserting that she has a satisfying assignment for τ, and a Prover, who is trying to force the Delayer into a contradiction by asking her values of the underlying variables. However, the Prover has limited memory and can only remember the values of up to w of the variables at a time.

Both players know τ and the contents of the Prover's memory, which is initially empty. At the start of each round there are at most $w-1$ values in memory. The Prover asks the Delayer the value of some variable whose value is not currently in memory. The Delayer responds with an answer (either 0 or 1), and upon receiving the answer, the Prover adds this assignment to his memory (increasing the number of stored values by 1). He can then erase (forget) any existing values from memory, possibly decreasing the number of stored values. The Prover declares victory if at some point, the partial assignment written in his memory falsifies one of the clauses of τ. The Delayer has a winning strategy for the width-w game on τ if no matter how the Prover plays the game, he cannot win. It was shown [36, 5] that the Delayer has a winning strategy for the width-w game if and only if the Res width of τ is at least $w-1$.

For our formula $\tau_{\mathcal{S}}$, the game proceeds as above, but now let D be the set of literals in the Prover's memory, and we demand instead of only holding w variables total in memory that $|I_0(D)|\leq k^2$ and $|J_0(D)|\leq k$. By the transformation from [36], the Prover has a winning strategy for this game if there is a Res refutation such that $|I_0(D)|\leq k^2-1$ and $|J_0(D)|\leq k-1$ for every clause D. Therefore the Delayer has a winning strategy for this game if and only if the lemma holds. The Delayer's winning strategy is as follows.

- If the Prover asks about a variable in x_i:
  - If $i \notin I_0(D)$ and after adding this bit there are still less than $\log m$ variables from x_i in memory, the Delayer can answer with either 0 or 1 arbitrarily.
  - If $i \notin I_0(D)$ but after adding this bit to memory there are now $\log m$ variables from x_i in memory, the Delayer uses the fact that $|J_0(D)| \leq k \leq \log m/4$ and the $(m, \log m/4)$-universal property of A to find a string $a_0 \in A$ such that $a_0|_{J_0(D)}$ is the all-zeros string, and uses the surjective property of f_x to find an assignment α_i consistent with the assignment to the x_i variables in memory such that $f_x(\alpha_i) = a_0$. The Delayer will remember the assignment α_i for x_i from now on, and note that $I_0(D)$ now contains i.
  - Finally if $i \in I_0(D)$ then the Delayer is maintaining an assignment α_i for x_i, so she answers according to α_i.
- If the Prover asks about a variable in y_j:
  - If $j \notin J_0(D)$ and after adding this bit there are still less than $\log n$ variables from y_j in memory, the Delayer can answer with either 0 or 1 arbitrarily.
  - If $j \notin J_0(D)$ but there are now $\log n$ variables from y_j in memory, the Delayer uses the fact that $|I_0(D)| \leq k^2 < \gamma(\mathcal{S})$ and finds a set S_0 that doesn't contain any element $i \in I_0(D)$, and uses the surjective property of f_y to find an assignment β_j consistent with the assignment to the y_j variables in memory such that $f_y(\beta_j) = S_0$. The Delayer will remember the assignment β_j for x_j, and note that $J_0(D)$ now contains j.
  - Finally if $j \in J_0(D)$ then the Delayer is already maintaining an assignment β_j for y_j, so she answers according to β_j.
- Whenever the Prover erases a variable from x_i from his memory, if $i \in I_0$ and now there are less than $\log m$ variables from x_i in memory, the Delayer forgets α_i. (note that i is no longer in I_0) Similarly, whenever the Prover erases a variable from y_j from his memory, if $j \in J_0$ and now there are less than $\log n$ variables from y_j in memory, the Delayer removes β_j from J_0. (note that j is no longer in J_0)

Assume for contradiction the game ends with the Prover winning. Consider when the game ends, and say the Prover claims the axiom $\overline{x_i^{\alpha_i} \wedge y_j^{\beta_j}}$ was falsified, and thus that $M_{\vec{\alpha}}[i,j] = N_{\vec{\beta}}[i,j] = 1$. First, consider the case when either $i \notin I_0$ or $j \notin J_0$. In either case there are is at least one variable in the axiom that is not in memory, which means that it has not been falsified, which is a contradiction. So assume that $i \in I_0$ and $j \in J_0$, and consider the last time that i was added to I_0 and the last time that j was added to J_0. Assume that i was added after j. Since j was in J_0 at the time we defined α_i, $M_{\alpha_i}[i,j] = 0$ by our choice of α_i, which is a contradiction. Finally assume that j was added after i. Then since i was in I_0 at the time we defined β_j, $f_y(\beta_j)$ does not contain i, and so $N_{\beta_j}[i,j] = 0$, which is also a contradiction. ◀

Before proceeding on to the proof of Lemma 11, we need to change Lemma 12 slightly, in order to be able to apply a restriction argument to turn width lower bounds into size lower bounds for $\tau_{\mathcal{S}}$. We use the notation $f|_\rho$ to denote the *restriction* of the function f over $x_1 \dots x_s$ by $\rho \in \{0,1,*\}^s$, which is the function f over the variables x_i for all $i \in \rho^{-1}(*)$ obtained by setting all other variables x_j to $\rho(j)$. Likewise we use the notation $\tau|_\rho$ to denote the restriction of the tautology τ by ρ.

▶ **Definition 13.** *Let $\rho_{x_i} \in \{0,1,*\}^{x_i}$ and let $\rho_{y_j} \in \{0,1,*\}^{y_j}$. Furthermore, let $\mathcal{R}$ be the set of all $\vec{\rho} = \{\rho_{x_1} \dots \rho_{x_n}, \rho_{y_1} \dots \rho_{y_m}\}$, such that for all $i \in [n]$ and $j \in [m]$, $|\rho_{x_i}^{-1}(*)| = 5 \log m$ and $|\rho_{y_j}^{-1}(*)| = 5 \log n$. Let f_x^i be the function f_x on the variables $\rho_{x_i}^{-1}(*)$ after restricting all other inputs to ρ_{x_i}, and likewise for f_y^j.*

▶ **Lemma 14** (Wide Clause Lemma under restrictions). *For sufficiently large n and $\rho \in \mathcal{R}$, if $\gamma(\mathcal{S}) > k^2$ then for any Res refutation π refuting $\tau_{\mathcal{S}}|_{\vec{\rho}}$ there exists a clause $D \in \pi$ such that $|I_0(D)| \geq k^2$ or $|J_0(D)| \geq k$.*

We omit the proof of Lemma 14, as it is essentially identical to Lemma 12. The only difference is that in each row i the Delayer chooses α_i based on f_x^i instead of f_x, and likewise for the columns. Note that f_x was $2 \log m$ surjective before the restriction, and since only $\log m$ variables are fixed in every row f_x^i is still $\log m$ surjective (and similarly for f_y^j).

Proof of Lemma 11. Let π be a Res refutation of $\tau_{\mathcal{S}}$ and assume for contradiction that $|\pi| < n^{k/16}$. First, consider a clause $D \in \pi$ such that $|I_0(D)| \geq k^2$. For each $i \in I_0(D)$, the chance that a randomly chosen $\vec{\rho} \in \mathcal{R}$ doesn't set one of the x_i literals in D to 1 is less than $(1 - (\frac{1}{6} \cdot \frac{1}{2}))^{\log m}$. Thus the probability that no $i \in I_0(D)$ sets D to 1 is at most $(\frac{11}{12})^{k^2 \log m} = (\frac{11}{12})^{k \log n} < \frac{1}{n^{k/8}}$. By a union bound the probability that some clause D in π satisfying $|I_0(D)| \geq k^2$ is not set to 1 is less than $\frac{n^{k/16}}{n^{k/8}} = \frac{1}{n^{k/16}}$, using the fact that $|\pi| < n^{k/16}$.

Similarly the probability that some clause $D \in \pi$ satisfying $|J_0(D)| \geq k$ is not set to 1 is at most $\frac{1}{n^{k/16}}$. Thus with probability at least $1 - \frac{2}{n^{k/16}}$, all clauses D satisfying $|I_0(D)| \geq k^2$ or $|J_0(D)| \geq k$ are set to 1 by a random restriction, and thus there exists a restriction $\vec{\rho} = \{\rho_{x_1} \dots \rho_{x_n}, \rho_{y_1} \dots \rho_{y_m}\}$ setting all such clauses to 1. However, this contradicts Lemma 14, as $\tau_{\mathcal{S}}|_{\vec{\rho}}$ must still have at least one such clause. Thus $S_{\mathcal{Q}}(\tau_{\mathcal{S}}) \geq n^{c_l k}$ for $c_l = \frac{1}{16}$. ◀

6 Conclusions

In terms of optimality of our results, the constructions in [14, 13] are not known to be optimal, and any hardness results against approximating the gap hitting set problem in time $n^{o(k)}$ for a larger value of k immediately gives a lower bound of $n^{o(k)}$ against automatizability. While their results are "optimal" in terms of fixed-parameter tractability guarantees, there is nothing limiting a different reduction from getting the same (or even a weaker) result that works for larger values of the fixed parameter.[10]

On the flip side, all of our hardness results also work for TreeRes and Nullsatz, and therefore this reduction is limited to quasipolynomial hardness. This is in line with the details of the reduction; by the crucial fact that $k^2 \leq \frac{\log m}{4} = \frac{\log n}{4k}$, this technique can't be strengthened past the $k = o(\log^{1/3} n)$ threshold.[11] Thus, the upper limit of improving the reductions of [14, 13] coincides almost exactly with the upper limit of our argument, and by extension any argument using the machinery of [2].

A central motivation of this work was to make the techniques clear and simple in hopes that they can be made to work for stronger systems such as SA and SoS, where no lower bounds are known. A degree lower bound matching our results for Res and PC would shed light on the limitations of our current approximation algorithms. Similarly it's possible

[10] Classically the hitting set problem has no $o(\log n)$ approximations; the obstacle to using this classical hardness is that it only rules out algorithms that get $o(\log n)$ approximation for *all* hitting set sizes, whereas [14] rules out algorithms for any *fixed* hitting set size. Nevertheless it's believed that $\Omega(\log n)$ hardness holds even for fixed hitting set sizes, and getting a reduction that achieves this result would strengthen our argument.

[11] If we allow the formula to be satisfiable in the case where $\gamma(\mathcal{S}) > k^2$ we only need $k \leq \frac{\log m}{4}$ since we only ever allow the proof to query k columns. This can also be made to work in the base setting where the formula must always be unsatisfiable by standard tricks. However this still yields a barrier of $k = o(\log^{1/2} n)$.

that this proof can be made to work for the case of TreeCP or CP, where instead of arguing lower bounds directly we can hope to leverage the power of lifting theorems [22, 21]; in particular a constant-sized lifting gadget would immediately give results for TreeCP matching our other results.

References

1 Michael Alekhnovich, Samuel R. Buss, Shlomo Moran, and Toniann Pitassi. Minimum Propositional Proof Length Is NP-Hard to Linearly Approximate. *J. Symb. Log.*, 66(1):171–191, 2001.

2 Michael Alekhnovich and Alexander A. Razborov. Resolution Is Not Automatizable Unless W[P] Is Tractable. *SIAM J. Comput.*, 38(4):1347–1363, 2008.

3 Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple Construction of Almost k-wise Independent Random Variables. *Random Struct. Algorithms*, 3(3):289–304, 1992.

4 Albart Atserias and Moritz Müller. Automating Resolution is NP-Hard. *CoRR*, abs/1904.02991, 2019. `arXiv:1904.02991`.

5 Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. *J. Comput. Syst. Sci.*, 74(3):323–334, 2008.

6 Albert Atserias, Massimo Lauria, and Jakob Nordström. Narrow Proofs May Be Maximally Long. *ACM Trans. Comput. Log.*, 17(3):19:1–19:30, 2016.

7 Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, and Pavel Pudlák. Lower Bound on Hilbert's Nullstellensatz and propositional proofs. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 794–806, 1994.

8 Paul Beame and Toniann Pitassi. Simplified and Improved Resolution Lower Bounds. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 274–282, 1996.

9 Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow - resolution made simple. *J. ACM*, 48(2):149–169, 2001.

10 Maria Luisa Bonet, Carlos Domingo, Ricard Gavaldà, Alexis Maciel, and Toniann Pitassi. Non-Automatizability of Bounded-Depth Frege Proofs. *Computational Complexity*, 13(1-2):47–68, 2004.

11 Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. On Interpolation and Automatization for Frege Systems. *SIAM J. Comput.*, 29(6):1939–1967, 2000.

12 Michael Brickenstein and Alexander Dreyer. PolyBoRi: A Framework for GröBner-basis Computations with Boolean Polynomials. *J. Symb. Comput.*, 44(9):1326–1345, 2009.

13 Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. From Gap-ETH to FPT-Inapproximability: Clique, Dominating Set, and More. *CoRR*, abs/1708.04218, 2017. `arXiv:1708.04218`.

14 Yijia Chen and Bingkai Lin. The Constant Inapproximability of the Parameterized Dominating Set Problem. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 505–514, 2016.

15 Matthew Clegg, Jeff Edmonds, and Russell Impagliazzo. Using the Groebner Basis Algorithm to Find Proofs of Unsatisfiability. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 174–183, 1996.

16 Martin Davis, George Logemann, and Donald Loveland. A Machine Program for Theorem-Proving. *J. ACM*, 5(7):394–397, 1961.

17 Martin Davis and Hilary Putnam. A Computing Procedure for Quantification Theory. *J. ACM*, 7(3):201–215, 1960.

18 Irit Dinur. Mildly exponential reduction from gap 3SAT to polynomial-gap label-cover. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:128, 2016.

19 K. Eickmeyer, M. Grohe, and M. Gruber. Approximation of natural W[P]-complete minimisation problems is hard. In *23rd Annual IEEE Conference on Computational Complexity, CCC*, pages 8–18, 2008.

20 Nicola Galesi and Massimo Lauria. On the Automatizability of Polynomial Calculus. *Theory Comput. Syst.*, 47(2):491–506, 2010.

21 Ankit Garg, Mika Göös, Pritish Kamath, and Dmitry Sokolov. Monotone Circuit Lower Bounds from Resolution. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 902–911, 2018.

22 Mika Göös, Toniann Pitassi, and Thomas Watson. Query-to-Communication Lifting for BPP. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA*, pages 132–143. IEEE Computer Society, 2017.

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

24 Stasys Jukna. *Extremal Combinatorics: With Applications in Computer Science*. Springer Publishing Company, Incorporated, 1st edition, 2010.

25 Jan Krajíček and Pavel Pudlák. Some Consequences of Cryptographical Conjectures for S^1_2 and EF. *Inf. Comput.*, 140(1):82–94, 1998.

26 J. Lasserre. Global Optimization With Polynomials and the Problem of Moments. *SIAM J. Optimization*, 11(3):796–817, 2001.

27 Pasin Manurangsi and Prasad Raghavendra. A Birthday Repetition Theorem and Complexity of Approximating Dense CSPs. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 78:1–78:15, 2017.

28 Hugues Marchand, Alexander Martin, Robert Weismantel, and Laurence Wolsey. Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics*, 123(1):397–446, 2002.

29 Joao Marques-Silva and Karem A. Sakallah. GRASP: A Search Algorithm for Propositional Satisfiability. *IEEE Trans. Computers*, 48:506–521, 1999.

30 Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an Efficient SAT Solver. In *DAC*, 2001.

31 Joseph Naor and Moni Naor. Small-Bias Probability Spaces: Efficient Constructions and Applications. *SIAM J. Comput.*, 22(4):838–856, 1993.

32 Jakob Nordström. On the Interplay Between Proof Complexity and SAT Solving. *ACM SIGLOG News*, 2(3):19–44, 2015.

33 Ryan O'Donnell. SOS is not obviously automatizable, even approximately. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:141, 2016.

34 Pablo Parrilo. Structured Semidefinite Programs and Semialgebraic Geometry. *Methods in Robustness and Optimization*, 2000.

35 K. Pipatsrisawat and A. Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence*, 175(2):512–525, 2011.

36 Pavel Pudlák. Proofs as Games. *The American Mathematical Monthly*, 107(6):541–550, 2000.

37 Prasad Raghavendra. Optimal Algorithms and Inapproximability Results for Every CSP? In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 245–254, 2008.

38 Prasad Raghavendra, Tselil Schramm, and David Steurer. High-dimensional estimation via sum-of-squares proofs. *CoRR*, 2018. `arXiv:1807.11419`.

39 Nathan Segerlind, Samuel R. Buss, and Russell Impagliazzo. A Switching Lemma for Small Restrictions and Lower Bounds for k-DNF Resolution. *SIAM J. Comput.*, 33(5):1171–1200, 2004.

40 H Sherali and W Adams. A Hierarchy of Relaxations Between the Continuous and Convex Hull Representations for Zero-One Programming Problems. *SIAM J. Disc. Math*, 3(3):411–430, 1990.

41 Johan Thapper and Stanislav Zivny. The power of Sherali-Adams relaxations for general-valued CSPs. *CoRR*, 2016. `arXiv:1606.02577`.

42 M. Vinyals, J. Elffers, J. Giráldez-Cru, S. Gocht, and J Nordström. On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence*, 175(2):512–525, 2011.

A Tight Approximation for Submodular Maximization with Mixed Packing and Covering Constraints

Eyal Mizrachi
Computer Science Department, Technion, Haifa 32000, Israel
eyalmiz@cs.technion.ac.il

Roy Schwartz
Computer Science Department, Technion, Haifa 32000, Israel
schwartz@cs.technion.ac.il

Joachim Spoerhase
Department of Computer Science, Aalto University, Espoo, Finland
joachim.spoerhase@aalto.fi

Sumedha Uniyal
Department of Computer Science, Aalto University, Espoo, Finland
`https://users.aalto.fi/~uniyals1/`
sumedha.uniyal@aalto.fi

Abstract

Motivated by applications in machine learning, such as subset selection and data summarization, we consider the problem of maximizing a monotone submodular function subject to mixed packing and covering constraints. We present a tight approximation algorithm that for any constant $\varepsilon > 0$ achieves a guarantee of $1 - 1/e - \varepsilon$ while violating only the covering constraints by a multiplicative factor of $1 - \varepsilon$. Our algorithm is based on a novel enumeration method, which unlike previously known enumeration techniques, can handle both packing and covering constraints. We extend the above main result by additionally handling a matroid independence constraint as well as finding (approximate) pareto set optimal solutions when multiple submodular objectives are present. Finally, we propose a novel and purely combinatorial dynamic programming approach. While this approach does not give tight bounds it yields *deterministic* and in some special cases also considerably faster algorithms. For example, for the well-studied special case of only packing constraints (Kulik et al. [Math. Oper. Res. '13] and Chekuri et al. [FOCS '10]), we are able to present the first deterministic non-trivial approximation algorithm. We believe our new combinatorial approach might be of independent interest.

2012 ACM Subject Classification Theory of computation → Submodular optimization and polymatroids; Theory of computation → Approximation algorithms analysis

Keywords and phrases submodular function, approximation algorithm, covering, packing

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.85

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1804.10947`.

Funding *Roy Schwartz*: Supported by ISF grant 1336/16 and NSF-BSF grant number 2016742.
Joachim Spoerhase: Supported by European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant number 759557) and by Academy of Finland (grant number 310415).
Sumedha Uniyal: Partially supported by Academy of Finland under the grant agreement number 314284.

Acknowledgements Joachim Spoerhase and Sumedha Uniyal thank an anonymous reviewer for pointing them to the fact that Theorem 6 also applies to polytopes that are not down-closed, which makes it possible to apply a randomized rounding approach.

1 Introduction

The study of combinatorial optimization problems with a submodular objective has attracted much attention in the last decade. A set function $f : 2^{\mathcal{N}} \to \mathbb{R}_+$ over a ground set $\mathcal{N}$ is called *submodular* if it has the *diminishing returns* property: $f(A \cup \{i\}) - f(A) \geq f(B \cup \{i\}) - f(B)$ for every $A \subseteq B \subseteq \mathcal{N}$ and $i \in \mathcal{N} \setminus B$.[1] Submodular functions capture the principle of economy of scale, prevalent in both theory and real world applications. Thus, it is no surprise that combinatorial optimization problems with a submodular objective arise in numerous disciplines, *e.g.*, machine learning and data mining [4, 5], algorithmic game theory and social networks [13, 22, 25, 27, 39], and economics [2]. Additionally, many classical problems in combinatorial optimization are in fact submodular in nature, *e.g.*, maximum cut and maximum directed cut [20, 21, 24, 26, 28], maximum coverage [15, 29], generalized assignment problem [8, 10, 14, 16], maximum bisection [3, 17], and facility location [1, 11, 12].

In this paper we consider the problem of maximizing a monotone[2] submodular function given mixed packing and covering constraints. In addition to being a natural problem in its own right, it has further real world applications.

As a motivating example consider the subset selection task in machine learning [18, 19, 30] (also refer to Kulesza and Taskar [31] for a thorough survey). In the subset selection task the goal is to select a diverse subset of elements from a given collection. One of the prototypical applications of this task is the document summarization problem [30, 34, 35]: given textual units the objective is to construct a short summary by selecting a subset of the textual units that is both representative and diverse. The former requirement, representativeness, is commonly achieved by maximizing a submodular objective function, *e.g.*, graph based [34, 35] or log subdeterminant [30]. The latter requirement, diversity, is typically tackled by penalizing the submodular objective for choosing similar textual units (this is the case for both of the above two mentioned submodular objectives). However, such an approach results in a submodular objective which is not necessarily non-negative thus making it extremely hard to cope with. As opposed to penalizing the objective, a remarkably simple and natural approach to tackle the diversity requirement is by the introduction of covering constraints. For example, one can require that for each topic that needs to appear in the summary, a sufficient number of textual units that refer to it are chosen. Unfortunately, to the best of our knowledge there is no previous work in the area of submodular maximization that incorporates general covering constraints.[3]

Let us now formally define the main problem considered in this paper. We are given a monotone submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}_+$ over a ground set $\mathcal{N} = \{1, 2, \ldots, n\}$. Additionally, there are p packing constraints given by $\mathbf{P} \in \mathbb{R}_+^{p \times n}$, and c covering constraints given by $\mathbf{C} \in \mathbb{R}_+^{c \times n}$ (all entries of $\mathbf{P}$ and $\mathbf{C}$ are non-negative). Our goal is to find a subset $S \subseteq \mathcal{N}$ that satisfies all packing and covering constraints that maximizes the value of f:

$$\max \{f(S) : S \subseteq \mathcal{N}, \mathbf{P}\mathbf{1}_S \leq \mathbf{1}_p, \mathbf{C}\mathbf{1}_S \geq \mathbf{1}_c\}. \tag{1}$$

In the above $\mathbf{1}_S \in \mathbb{R}^n$ is the indicator vector for $S \subseteq \mathcal{N}$ and $\mathbf{1}_k \in \mathbb{R}^k$ is a vector of dimension k whose coordinates are all 1. We denote this problem as Packing-Covering Submodular Maximization (PCSM). It is assumed we are given a feasible instance, *i.e.*, there exists $S \subseteq \mathcal{N}$ such that $\mathbf{P}\mathbf{1}_S \leq \mathbf{1}_p$ and $\mathbf{C}\mathbf{1}_S \geq \mathbf{1}_c$.

---

[1] An equivalent definition is: $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$ for every $A, B \in \mathcal{N}$.
[2] f is monotone if $f(S) \leq f(T)$ for every $S \subseteq T \subseteq \mathcal{N}$.
[3] There are works on exact cardinality constraints for non-monotone submodular functions, which implies a special, uniform covering constraint [6, 33, 41].

As previously mentioned, (PCSM) captures several well known problems as a special case when only a single packing constraint is present ($p = 1$ and $c = 0$): maximum coverage [29], and maximization of a monotone submodular function given a knapsack constraint [40, 42] or a cardinality constraint [37]. For all of these special cases an approximation of $(1 - 1/e)$ is achievable and known to be tight [38] (even for the special case of a coverage function [15]). When a constant number of knapsack constraints is given ($p = O(1)$ and $c = 0$) Kulik et al. [32] presented a tight $(1 - 1/e - \varepsilon)$-approximation for any constant $\varepsilon > 0$. An alternative algorithm with the same guarantee was given by Chekuri et al. [9].

Our Results. We present a tight approximation guarantee for (PCSM) when the number of constraints is constant. Recall that we assume we are given a feasible instance, *i.e.*, there exists $S \subseteq \mathcal{N}$ such that $\mathbf{P1}_S \leq \mathbf{1}_p$ and $\mathbf{C1}_S \geq \mathbf{1}_c$. The following theorem summarizes our main result. From this point onwards we denote by O some fixed optimal solution to the problem at hand.

▶ **Theorem 1.** *For every constant $\varepsilon > 0$, assuming p and c are constants, there exists a randomized polynomial time algorithm for* (PCSM) *running in time $n^{poly(1/\varepsilon)}$ that outputs a solution $S \subseteq \mathcal{N}$ that satisfies:* (1) $f(S) \geq (1 - 1/e - \varepsilon)\, f(O)$*; and* (2) $\mathbf{P1}_S \leq \mathbf{1}_p$ *and* $\mathbf{C1}_S \geq (1 - \varepsilon)\mathbf{1}_c$.

We note four important remarks regarding the tightness of Theorem 1:

1. The loss of $1-1/e$ in the approximation cannot be avoided, implying that our approximation guarantee is (virtually) tight. The reason is that no approximation better than $1 - 1/e$ can be achieved even for the case where only a single packing constraint is present [38].
2. The assumption that the number of constraints is constant is unavoidable. The reason is that if the number of constraints is not assumed to be constant, then even with a linear objective (PCSM) captures the maximum independent set problem. Hence, no approximation better than $n^{-(1-\varepsilon)}$, for any constant $\varepsilon > 0$, is possible [23].[4]
3. No *true* approximation with a finite approximation guarantee is possible, *i.e.*, finding a solution $S \subseteq \mathcal{N}$ such that $\mathbf{P1}_S \leq \mathbf{1}_p$ and $\mathbf{C1}_S \geq \mathbf{1}_c$ with no violation of the constraints. The reason is that one can easily encode the subset sum problem using a single packing and a single covering constraint. Thus, just deciding whether a feasible solution exists, regardless of its cost, is already NP-hard.
4. Guaranteeing one-sided feasibility, *i.e.*, finding a solution which does not violate the packing constraints and a violates the covering constraint only by a factor of $1 - \varepsilon$, cannot be achieved in time $n^{o(1/\varepsilon)}$ unless the exponential time hypothesis fails (see [36] for details).

Therefore, we can conclude that our main result (Theorem 1) provides the best possible guarantee for the (PCSM) problem. We also note that all previous work on the special case of only packing constraints [9, 32] have the same running time of $n^{\text{poly}(1/\varepsilon)}$.

We present additional extensions of the above main result. The first extension deals with (PCSM) where we are also required that the output is an independent set in a given matroid $\mathcal{M} = (\mathcal{N}, \mathcal{I})$. We denote this problem by Matroid Packing-Covering Submodular Maximization (MatroidPCSM), and it is defined as follows:

[4] If the number of packing constraints p is super-constant then approximations are known only for special cases with "loose" packing constraints, *i.e.*, $P_{i,\ell} \leq O(\varepsilon^2 / \ln p)$ (see, *e.g.*, [9]).

$\max\{f(S) : S \subseteq \mathcal{N}, \mathbf{P1}_S \leq \mathbf{1}_p, \mathbf{C1}_S \geq \mathbf{1}_c, S \in \mathcal{I}\}$. As in (PCSM), we assume we are given a feasible instance, *i.e.*, there exists $S \subseteq \mathcal{N}$ such that $\mathbf{P1}_S \leq \mathbf{1}_p$, $\mathbf{C1}_S \geq \mathbf{1}_c$, and $S \in \mathcal{I}$. Our result is summarized in the following theorem.

▶ **Theorem 2.** *For every constant $\varepsilon > 0$, assuming p and c are constants, there exists a randomized polynomial time algorithm for* (MatroidPCSM) *that outputs a solution $S \in \mathcal{I}$ that satisfies:* (1) $f(S) \geq (1 - 1/e - \varepsilon)\, f(O)$*; and* (2) $\mathbf{P1}_S \leq \mathbf{1}_p$ *and* $\mathbf{C1}_S \geq (1-\varepsilon)\mathbf{1}_c$.

The second extension deals with the multi-objective variant of (PCSM) where we wish to optimize over several monotone submodular objectives. We denote this problem by Packing-Covering Multiple Submodular Maximization (MultiPCSM). Its input is identical to that of (PCSM) except that instead of a single objective f we are given t monotone submodular functions $f_1, \ldots, f_t : 2^{\mathcal{N}} \to \mathbb{R}_+$. As before, we assume we are given a feasible instance, *i.e.*, there exists $S \subseteq \mathcal{N}$ such that $\mathbf{P1}_S \leq \mathbf{1}_p$ and $\mathbf{C1}_S \geq \mathbf{1}_c$. Our goal is to find pareto set solutions considering the t objectives. To this end we prove the following theorem.

▶ **Theorem 3.** *For every constant $\varepsilon > 0$, assuming p, c and t are constants, there exists a randomized polynomial time algorithm for* (MultiPCSM) *that for every target values $v_1, \ldots, v_t$ either:* (1) *finds a solution $S \subseteq \mathcal{N}$ where $\mathbf{P1}_S \leq \mathbf{1}_p$ and $\mathbf{C1}_S \geq (1-\varepsilon)\mathbf{1}_c$ such that for every $1 \leq i \leq t$: $f_i(S) \geq (1 - 1/e - \varepsilon)\, v_i$; or* (2) *returns a certificate that there is no solution $S \subseteq \mathcal{N}$, where $\mathbf{P1}_S \leq \mathbf{1}_p$ and $\mathbf{C1}_S \geq \mathbf{1}_c$ such that for every $1 \leq i \leq t$: $f_i(S) \geq v_i$.*

We also note that Theorems 2 and 3 can be combined such that we can handle (MultiPCSM) where a matroid independence constraint is present, in addition to the given packing and covering constraints, achieving the same guarantees as in Theorem 3.

All our previously mentioned results employ a continuous approach and are based on the multilinear relaxation, and thus are inherently *randomized*.[5] We present a new combinatorial greedy-based dynamic programming approach for submodular maximization that enables us, for several well studied special cases of (PCSM), to obtain deterministic and considerably faster algorithms. Perhaps the most notable result is the first deterministic non-trivial algorithm for maximizing a monotone submodular function subject to a constant number of packing constraints (previous works [9, 32] are randomized).

▶ **Theorem 4.** *For every constants $\varepsilon > 0$ and $p \in \mathbb{N}$, there exists a deterministic algorithm for maximizing a monotone submodular function subject to p packing constraints, that runs in time $O(n^{\mathrm{poly}(1/\varepsilon)})$ and achieves an approximation of $1/e - \varepsilon$.*

The interesting special case of (PCSM) is when a single packing and a single covering constraints are present ($p = c = 1$) is summarized in the following theorem.

▶ **Theorem 5.** *For every constant $\varepsilon > 0$ and $p = c = 1$, there exists a deterministic algorithm for* (PCSM) *running in time $O(n^{1/\varepsilon})$ that outputs a solution $S \subseteq \mathcal{N}$ that satisfies:* (1) $f(S) \geq 0.353 f(O)$*; and* (2) $\mathbf{P1}_S \leq (1+\varepsilon)\mathbf{1}_p$ *and* $\mathbf{C1}_S \geq (1-\varepsilon)\mathbf{1}_c$*. For the case when the packing constraint is a cardinality constraint,* i.e.*, $\mathbf{P} = \mathbf{1}_n^\intercal/k$, we can further guarantee that $\mathbf{P1}_S \leq \mathbf{1}_p$ and a running time of $O(n^4/\varepsilon)$.*

Our Techniques. Our main result is based on a continuous approach: first a continuous relaxation is formulated, second it is (approximately) solved, and finally the fractional solution is rounded into an integral solution. Similarly to the previous works of [9, 32], which

[5] Known techniques to efficiently evaluate the multilinear extension are randomized, *e.g.*, [7].

focus on the special case of only packing constraints, the heart of the algorithm lies in an enumeration preprocessing phase that chooses and discards some of the elements prior to formulating the relaxation. The enumeration preprocessing step of [9, 32] is remarkably simple and elegant. It enumerates over all possible collections of large elements the optimal solution chooses, *i.e.*, elements whose size exceeds some fixed constant in at least one of the packing constraints and are chosen by the optimal solution.[6] All remaining large elements not in the guessed collection are discarded. This enumeration terminates in polynomial time and ensures that no large elements are left in any of the packing constraints. Thus, once no large elements remain concentration bounds can be applied. For the correct guess, any of the several known randomized rounding techniques can be employed (alongside a simple rescaling) to obtain an approximation of $1 - 1/e - \varepsilon$ (here $\varepsilon > 0$ is a constant that is used to determine which elements are considered large). Unfortunately, this approach fails in the presence of covering constraints since an optimal solution can choose *many* large elements in any given covering constraint. One can naturally adapt the above known preprocessing by enumerating over all possible collections of covering constraints that the optimal solution O covers using only large elements. However, this leads to an approximation of $1 - 1/e - \varepsilon$ while *both* packing and covering constraints are violated by a multiplicative factor of $1 \pm \varepsilon$. We aim to obtain one sided violation of the constraints, *i.e.*, only the covering constraints are violated by a factor of $1 - \varepsilon$ whereas the packing constraints are fully satisfied.

Avoiding constraint violation is possible in the presence of pure packing constraints [9, 32]. Known approaches for the latter are crucially based on *removing* elements in a pre-processing and post-processing step in order to guarantee that concentration bounds hold. For mixed constraints, these known removal operations may, however, arbitrarily violate the covering constraints. Our approach aims at pre-processing the input instance via partial enumeration so as to avoid discarding elements by ensuring that the remaining elements are "locally" small relatively to the *residual* constraints. If this property would hold scaling down the solution by a factor $1/(1+\varepsilon)$ would be sufficient to avoid violation of the packing constraints. Unfortunately, we cannot guarantee this to hold for all constraints. Rather, for some *critical* constraints locally large elements may still be present. We introduce a novel enumeration process that detects these critical constraints, *i.e.*, constraints that are prone to violation. Such constraints are given special attention as the randomized rounding might cause them to significantly deviate from the target value. Unlike the previously known preprocessing method, our enumeration process handles covering constraints with much care and it takes into account the *actual* coverage of the optimal solution O of each of the covering constraints. Combining the above, alongside a postprocessing phase that discards large elements from critical packing constraints, suffices to yield the desired result.

We also independently present a novel purely combinatorial greedy-based dynamic programming approach that yields deterministic and in some special cases considerably faster algorithms. Previously, greedy algorithms were known for one cardinality constraint [37] and one packing constraint [40]. But in the presence multiple constraints, it is not clear how to design a rule to greedily pick the next element. In fact, we tried several natural greedy strategies but all of them failed. This holds even when some oracle gives us the set of vectors of packing and covering values from an optimum solution and the algorithm follows any *fixed* sequence of these values.

[6] An additional part of the preprocessing involves enumerating over collections of elements whose marginal value is large with respect to the objective f, however this part of the enumeration is not affected by the presence of covering constraints and thus is ignored in the current discussion.

In our approach we maintain a table that contains greedy approximate solutions for all possible packing and covering values. Using this table we extend the simple greedy process by populating each table entry with the most profitable extension of the previous table entries. In this way we are able to simulate (in a certain sense) *all* possible sequences of packing and covering values for the greedy algorithm, ultimately leading to a good feasible solution. Our result implies that there exists one sequence (depending on the instance) where greedy performs well. To estimate the approximation factor we employ a factor-revealing linear program. To the best of our knowledge, this is the first time a dynamic programming based approach is used for submodular optimization. We believe our new combinatorial dynamic programming approach is of independent interest.

2 Preliminaries

In this paper we assume the standard *value oracle* model, where the algorithm can only access the given submodular function f with queries of the form: what is $f(S)$ for a given S? The running time of the algorithm is measured by the total number of value oracle queries and arithmetic operations it performs. Additionally, let us define $f_A(S) \triangleq f(A \cup S) - f(A)$ for any subsets $A, S \subseteq \mathcal{N}$. Furthermore, let $f_A(\ell) \triangleq f_A(\{\ell\})$.

The multiliear extension $F : [0,1]^{\mathcal{N}} \to \mathbb{R}_+$ of a given set function $f : 2^{\mathcal{N}} \to \mathbb{R}_+$ is:

$$F(\mathbf{x}) \triangleq \sum_{R \subseteq \mathcal{N}} f(R) \prod_{\ell \in R} x_\ell \prod_{\ell \notin R} (1 - x_\ell) \qquad \forall \mathbf{x} \in [0,1]^{\mathcal{N}}.$$

Additionally, we make use of the following theorem that provides the guarantees of the continuous greedy algorithm of [7].[7]

▶ **Theorem 6** (Chekuri et al. [7]). *We are given a ground set $\mathcal{N}$, a monotone submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}_+$, and a polytope $\mathcal{P} \subseteq [0,1]^{\mathcal{N}}$. If $\mathcal{P} \neq \emptyset$ and one can solve in polynomial time $\operatorname{argmax}\{\mathbf{w}^T\mathbf{x} : \mathbf{x} \in \mathcal{P}\}$ for any $\mathbf{w} \in \mathbb{R}^{\mathcal{N}}$, then there exists a polynomial time algorithm that finds $\mathbf{x} \in \mathcal{P}$ where $F(\mathbf{x}) \geq (1 - 1/e)\, F(\mathbf{x}^*)$. Here $\mathbf{x}^*$ is an optimal solution to the problem:* $\max\{F(\mathbf{y}) : \mathbf{y} \in \mathcal{P}\}$.

3 Algorithms for the (PCSM) Problem

Preprocessing – Enumeration with Mixed Constraints. We define a *guess* D to be a triplet $(E_0, E_1, \mathbf{c}')$, where $E_0 \subseteq \mathcal{N}$ denotes elements that are discarded, $E_1 \subseteq \mathcal{N}$ denotes elements that are chosen, and $\mathbf{c}' \in \mathbb{R}_+^c$ represents a rough estimate (up to a factor of $1 + \varepsilon$) of how much an optimal solution O covers each of the covering constraints, *i.e.*, $\mathbf{C}\mathbf{1}_O$. Let us denote by $\tilde{\mathcal{N}} \triangleq \mathcal{N} \setminus (E_0 \cup E_1)$ the remaining undetermined elements with respect to guess D.

We would like to define when a given fixed guess $D = (E_0, E_1, \mathbf{c}')$ is *consistent*, and to this end we introduce the notion of *critical* constraints. For the i^{th} packing constraint the residual value that can still be packed is: $(\mathbf{r}_D)_i \triangleq 1 - \sum_{\ell \in E_1} \mathbf{P}_{i,\ell}$, where $\mathbf{r}_D \in \mathbb{R}^p$. For the j^{th} covering constraint the residual value that still needs to be covered is: $(\mathbf{s}_D)_j \triangleq \max\left\{0, \mathbf{c}'_j - \sum_{\ell \in E_1} \mathbf{C}_{j,\ell}\right\}$, where $\mathbf{s}_D \in \mathbb{R}^c$. A packing constraint i is called *critical* if $(\mathbf{r}_D)_i \leq \delta$, and a covering constraint j is called *critical* if $(\mathbf{s}_D)_j \leq \delta \mathbf{c}'_j$ ($\delta \in (0,1)$ is

[7] We note that the actual guarantee of the continuous greedy algorithm is $(1 - 1/e - o(1))$. However, for simplicity of presentation, we can ignore the $o(1)$ term due to the existence of a loss of ε (for any constant ε) in all of our theorems.

a parameter to be chosen later). Thus, the collections of critical packing and covering constraints, for a given guess D, are given by: $Y_D \triangleq \{i = 1, \ldots, p : (\mathbf{r}_D)_i \leq \delta\}$ and $Z_D \triangleq \{j = 1, \ldots, c : (\mathbf{s}_D)_j \leq \delta \mathbf{c}'_j\}$. Moreover, elements are considered *large* if their size is at least some factor α of the residual value of some non-critical constraint ($\alpha \in (0, 1)$ is a parameter to be chosen later). Formally, the collection of large elements with respect to the packing constraints is defined as $P_D \triangleq \{\ell \in \tilde{\mathcal{N}} : \exists i \notin Y_D \text{ s.t. } \mathbf{P}_{i,\ell} \geq \alpha(\mathbf{r}_D)_i\}$, and the collection of large elements with respect to the covering constraints is defined as $C_D = \{\ell \in \tilde{\mathcal{N}} : \exists j \notin Z_D \text{ s.t. } \mathbf{C}_{j,\ell} \geq \alpha(\mathbf{s}_D)_j\}$. It is important to note, as previously mentioned, that the notion of a large element is with respect to the residual constraint, as opposed to previous works [9, 32] where the definition is with respect to the original constraint. Let us now formally define when a guess D is called *consistent*.

▶ **Definition 1.** *A guess $D = (E_0, E_1, \mathbf{c}')$ is* consistent *if:* (1) $E_0 \cap E_1 = \emptyset$; (2) $\mathbf{c}' \geq \mathbf{1}_c$; (3) $\mathbf{P}\mathbf{1}_{E_1} \leq \mathbf{1}_p$; *and* (4) $P_D = C_D = \emptyset$.

Intuitively, requirement (1) states that a variable cannot be both chosen and discarded, (2) states that the each covering constraint is satisfied by an optimal solution O, (3) states the chosen elements E_1 do not violate the packing constraints, and (4) states that no large elements remain in any non-critical constraint.

Finally, we need to define when a consistent guess is *correct*. Assume without loss of generality that $O = \{o_1, \ldots, o_k\}$ and the elements of O are ordered greedily: $f_{\{o_1,\ldots,o_i\}}(o_{i+1}) \leq f_{\{o_1,\ldots,o_{i-1}\}}(o_i)$ for every $i = 1, \ldots, k-1$. In the following definition γ is a parameter to be chosen later.

▶ **Definition 2.** *A consistent guess $D = (E_0, E_1, \mathbf{c}')$ is called* correct *with respect to O if:* (1) $E_1 \subseteq O$; (2) $E_0 \subseteq \bar{O}$; (3) $\{o_1, \ldots, o_\gamma\} \subseteq E_1$; *and* (4) $\mathbf{c}' \leq \mathbf{C}\mathbf{1}_O \leq (1+\delta)\mathbf{c}'$.

Intuitively, requirement (1) states that the chosen elements E_1 are indeed elements of O, (2) states that no element of O is discarded, (3) states that the γ elements of largest marginal value are all chosen, and (4) states that $\mathbf{c}'$ represents (up to a factor of $1+\delta$) how much O actually covers each of the covering constraints.

We are now ready to present our preprocessing algorithm (Algorithm 1), which produces a list $\mathcal{L}$ of consistent guesses that is guaranteed to contain at least one guess that is also correct with respect to O. Lemma 7 summarizes this, its proof appears in [36].

Algorithm 1: Preprocessing.

```
L ← ∅
foreach j_1, ..., j_c ∈ {0, 1, ..., ⌈log_{1+δ} n⌉} do
    Let c' = ((1+δ)^{j_1}, ..., (1+δ)^{j_c})
    foreach E_1 ⊆ N such that |E_1| ≤ γ + (p+c)/(αδ) do
        Let H = (∅, E_1, c')
        Let E_0 = {ℓ ∈ N \ E_1 : f_{E_1}(ℓ) > (γ^{-1}) f(E_1)} ∪ P_H ∪ C_H
        Set D = (E_0, E_1, c')
        If D is consistent according to Definition 1 add it to L.
Output L.
```

▶ **Lemma 7.** *The output $\mathcal{L}$ of Algorithm 1 contains at least one guess D that is correct with respect to some optimal solution O.*

Proof. Fix any optimal solution O. At least one of the vectors $\mathbf{c}'$ enumerated by Algorithm 1 satisfies property (4) in Definition 2 with respect to O. Let us fix an iteration in which such a $\mathbf{c}'$ is enumerated. Define the "large" elements O has with respect to this $\mathbf{c}'$:

$$O_L \triangleq \{\ell \in O : \exists i \text{ s.t. } \mathbf{P}_{i,\ell} \geq \alpha\delta\} \cup \left\{\ell \in O : \exists j \text{ s.t. } \mathbf{C}_{j,\ell} \geq \alpha\delta\mathbf{c}'_j\right\}. \tag{2}$$

Denote by $O_\gamma \triangleq \{o_1, \ldots, o_\gamma\}$ the γ elements of O with the largest marginal (recall the ordering of O satisfies: $f_{\{o_1,\ldots,o_i\}}(o_{i+1}) \leq f_{\{o_1,\ldots,o_{i-1}\}}(o_i)$). Let us fix $E_1 \triangleq O_\gamma \cup O_L$ and choose $H \triangleq (\emptyset, E_1, \mathbf{c}')$. Clearly, $|E_1| \leq \gamma + {}^{(p+c)}/_{(\alpha\delta)}$ since $|O_\gamma| = \gamma$ and $|O_L| \leq {}^{(p+c)}/_{(\alpha\delta)}$. Hence, we can conclude that H is considered by Algorithm 1.

We fix the iteration in which the above H is considered and show that the resulting $D = (E_0, E_1, \mathbf{c}')$ of this iteration is correct and consistent (recall that Algorithm 1 chooses $E_0 = \{\ell \in \mathcal{N} \setminus E_1 : f_{E_1}(\ell) > (\gamma^{-1})f(E_1)\} \cup P_H \cup C_H$). The following two observations suffice to complete the proof:

Observation 1: $\forall \ell \in O \cup (\mathcal{N} \setminus E_0)$: $f_{E_1}(\ell) \leq \gamma^{-1} f(E_1)$.

Observation 2: $O \cap P_H = \emptyset$ and $O \cap C_H = \emptyset$.

Clearly properties (1) and (3) of Definition 2 are satisfied by construction of E_1, H, and subsequently D. Property (2) of Definition 2 requires the above two observations, which together imply that no element of O is added to E_0 by Algorithm 1. Thus, all four properties of Definition 2 are satisfied, and we focus on showing that the above D is consistent according to Definition 1. Property (1) of Definition 1 follows from properties (1) and (2) of Definition 2. Property (2) of Definition 1 follows from the choice of $\mathbf{c}'$. Property (3) of Definition 1 follows from the feasibility of O and property (1) of Definition 2. Lastly, property (4) of Definition 1 follows from the fact that $P_D \subseteq P_H$ and that $P_H \subseteq E_0$, implying that $P_D = \emptyset$ (the same argument applies to C_D). We are left with proving the above two observations.

We start with proving the first observation. Let $\ell \in O \cup (\mathcal{N} \setminus E_0)$. If $\ell \in \mathcal{N} \setminus E_0$ then the observation follows by the construction of E_0 in Algorithm 1. Otherwise, $\ell \in O$. If $\ell \in O_\gamma$ then we have that $f_{E_1}(\ell) = 0$ since $O_\gamma \subseteq E_1$. Otherwise $\ell \in O \setminus O_\gamma$. Note:

$$f_{E_1}(\ell) \leq f_{O_\gamma}(\ell) \leq \gamma^{-1} f(O_\gamma) \leq \gamma^{-1} f(E_1).$$

The first inequality follows from diminishing returns and $O_\gamma \subseteq E_1$. The third and last inequality follows from the monotonicity of f and $O_\gamma \subseteq E_1$. Let us focus on the second inequality, and denote $O = \{o_1, \ldots, o_k\}$ and the sequence $a_i \triangleq f_{\{o_1,\ldots,o_{i-1}\}}(o_i)$. The sequence of a_is is monotone non-increasing by the ordering of O and the monotonicity of f implies that all a_is are non-negative. Note that $a_1 + \ldots + a_\gamma = f(O_\gamma)$, thus implying that $f_{O_\gamma}(\ell) \leq \gamma^{-1} f(O_\gamma)$ for every $\ell \in \{o_{\gamma+1}, \ldots, o_k\}$ (otherwise $a_1 + \ldots + a_\gamma > f(O_\gamma)$). The second inequality above, *i.e.*, $f_{O_\gamma}(\ell) \leq \gamma^{-1} f(O_\gamma)$, now follows since $\ell \in O \setminus O_\gamma = \{o_{\gamma+1}, \ldots, o_k\}$.

Let us now focus on proving the second observation. Let us assume on the contrary that there is an element ℓ such that $\ell \in O \cap P_H$. Recall that $P_H = \{\ell \in \mathcal{N} \setminus E_1 : \exists i \notin Y_H \text{ s.t. } \mathbf{P}_{i,\ell} \geq \alpha(\mathbf{r}_H)_i\}$ where $Y_H = \{i : (\mathbf{r}_H)_i \leq \delta\}$. This implies that $\ell \in O \setminus E_1$, namely that $\ell \notin O_L$, from which we derive that for *all* packing constraint i we have that $\mathbf{P}_{i,\ell} < \alpha\delta$. Since $\ell \in P_H$ we conclude that there exists a packing constraint i for which $(\mathbf{r}_H)_i \leq \mathbf{P}_{i,\ell}/\alpha$. Combining the last two bounds we conclude that $(\mathbf{r}_H)_i < \delta$, which implies that the i^{th} packing constraint is critical, *i.e.*, $i \in Y_H$. This is a contradiction, and hence $O \cap P_H = \emptyset$. A similar proof applies to C_H and the covering constraints. □ ◀

Randomized Rounding. Before presenting our main rounding algorithm, let us define the residual problem we are required to solve given a consistent guess D. First, the residual objective $g : 2^{\tilde{\mathcal{N}}} \to \mathbb{R}_+$ is defined as: $g(S) \triangleq f(S \cup E_1) - f(E_1)$ for every $S \subseteq \tilde{\mathcal{N}}$. Clearly, g

is submodular, non-negative, and monotone. Second, let us focus on the feasible domain and denote by $\tilde{\mathbf{P}}$ ($\tilde{\mathbf{C}}$) the submatrix of $\mathbf{P}$ ($\mathbf{C}$) obtained by choosing all the columns in $\tilde{\mathcal{N}}$. Hence, given $D = (E_0, E_1, \mathbf{c}')$ the residual problem is:

$$\max\{g(S) + f(E_1) : S \subseteq \tilde{\mathcal{N}}, \tilde{\mathbf{P}}\mathbf{1}_S \leq \mathbf{r}_D, \tilde{\mathbf{C}}\mathbf{1}_S \geq \mathbf{s}_D\}. \tag{3}$$

In order to formulate the multilinear relaxation of (3), consider the following two polytopes: $\mathcal{P} \triangleq \{\mathbf{x} \in [0,1]^{\tilde{\mathcal{N}}} : \tilde{\mathbf{P}}\mathbf{x} \leq \mathbf{r}_D\}$ and $\mathcal{C} \triangleq \{\mathbf{x} \in [0,1]^{\tilde{\mathcal{N}}} : \tilde{\mathbf{P}}\mathbf{x} \geq \mathbf{s}_D\}$. Let $G : [0,1]^{\tilde{\mathcal{N}}} \to \mathbb{R}_+$ be the multilinear extension of g. Thus, the continuous multilinear relaxation of (3) is:

$$\max\left\{f(E_1) + G(\mathbf{x}) : \mathbf{x} \in [0,1]^{\tilde{\mathcal{N}}}, \mathbf{x} \in \mathcal{P} \cap \mathcal{C}\right\}. \tag{4}$$

Our algorithm performs randomized rounding of a fractional solution to the above relaxation (4). However, this is not enough to obtain our main result and an additional post-processing step is required in which additional elements are discarded. Since covering constraints are present, one needs to perform the post-processing step in great care. To this end we denote by L_D the collection of large elements with respect to some critical packing constraint: $L_D \triangleq \{\ell \in \tilde{\mathcal{N}} : \exists i \in Y_D \text{ s.t. } \mathbf{P}_{i,\ell} \geq \beta \mathbf{r}_D\}$ ($\beta \in (0,1)$ is a parameter to be chosen later). Intuitively, we would like to discard elements in L_D since choosing any one of those will incur a violation of a packing constraint. We are now ready to present our rounding algorithm (Algorithm 2).

Algorithm 2: $(f, \mathcal{N}, \mathbf{P}, \mathbf{C})$.

1 Use Algorithm 1 to obtain a list of guesses $\mathcal{L}$.
2 **foreach** $D = (E_0, E_1, \mathbf{c}') \in \mathcal{L}$ **do**
3 Use Theorem 6 to compute an approximate solution $\mathbf{x}^*$ to problem (4).
4 Scale down $\mathbf{x}^*$ to $\bar{\mathbf{x}} = \mathbf{x}^*/(1+\delta)$
5 Let R_D be such that for every $\ell \in \tilde{\mathcal{N}}$ independently: $\Pr[\ell \in R_D] = \bar{\mathbf{x}}_\ell$.
6 Let $R'_D = R_D \setminus L_D$.
7 $S_D \leftarrow E_1 \cup R'_D$.
8 $S_{alg} \leftarrow \operatorname{argmax}\{f(S_D) : D \in \mathcal{L}, \mathbf{P} \cdot \mathbf{1}_{S_D} \leq \mathbf{1}_p, \mathbf{C} \cdot \mathbf{1}_{S_D} \geq (1-\varepsilon)\mathbf{1}_c\}$

We note that Line 6 of Algorithm 2 is the post-processing step where all elements of L_D are discarded. Our analysis of Algorithm 2 shows that in an iteration a correct guess D is examined, with a constant probability, S_D satisfies the packing constraints, violates the covering constraint by only a fraction of ε, and $f(S_D)$ is sufficiently high.

The following lemma gives a lower bound on the value of the fractional solution $\bar{\mathbf{x}}$ computed by Algorithm 2 (for a full proof refer [36]).

▶ **Lemma 8.** *If $D \in \mathcal{L}$ is correct then in the iteration of Algorithm 2 it is examined the resulting $\bar{\mathbf{x}}$ satisfies: $G(\bar{\mathbf{x}}) \geq (1 - 1/e - \delta)f(O) - f(E_1)$.*

Let us now fix an iteration of Algorithm 2 for which D is not only consistent but also correct (the existence of such an iteration is guaranteed by Lemma 7). Intuitively, Algorithm 2 performs a straightforward randomized rounding where each element $\ell \in \tilde{\mathcal{N}}$ is independently chosen with a probability that corresponds to its fractional value in the solution of the multilinear relaxation (4). However, two key ingredients in Algorithm 2 are required in order to achieve an ε violation of the covering constraints and no violation of the packing constraints: (1) *scaling*: prior to the randomized rounding $\mathbf{x}^*$ is scaled down by a factor $(1+\delta)$ (line 4 in Algorithm 2); and (2) *post-processing*: after the randomized rounding all chosen large elements in a critical packing constraint are discarded (line 6 in Algorithm 2).

The first ingredient above (scaling of $\mathbf{x}^*$) allows us to prove using standard concentration bounds that with good probability all non-critical packing constraints are not violated. However, when considering critical packing constraints this does not suffice and the second ingredient above (discarding L_D) is required to show that with good probability even the critical packing constraints are not violated. While discarding L_D is beneficial when considering packing constraints, it might have a destructive effect on both the covering constraints and the value of the objective. To remedy this we argue that with high probability only few elements in L_D are actually discarded, *i.e.*, $|R_D \cap L_D|$ is sufficiently small. Combining the latter fact with the assumption that the current guess D is not only consistent but also correct, according to Definition 2, allows us to prove the following lemma (for a full proof refer to [36]).

▶ **Lemma 9.** *For any constant $\varepsilon > 0$, choose constants $\alpha = \delta^3$, $\beta = \delta^2/(3b)$, $\gamma = 1/\delta^3$, and $\delta < \min\{1/(15(p+c)), \varepsilon/(2+30(p+c)^2)\}$. With a probability of at least $1/2$ Algorithm 2 outputs a solution S_{alg} satisfying: (1) $\mathbf{P}\mathbf{1}_{S_{alg}} \leq \mathbf{1}_p$; (2) $\mathbf{C}\mathbf{1}_{S_{alg}} \geq (1-\varepsilon)\mathbf{1}_c$; and (3) $f(S_{alg}) \geq (1 - 1/e - \varepsilon)f(O)$.*

The above lemma suffices to prove Theorem 1, as it immediately implies it.

4 Greedy Dynamic Programming

In this section, we present a novel algorithmic approach for submodular maximization that leads to *deterministic* and considerably faster approximation algorithms in several settings. Perhaps the most notable application of our approach is Theorem 4. To the best of our knowledge, it provides the first *deterministic* non-trivial approximation algorithm for maximizing a monotone submodular function subject to packing constraints. To highlight the core idea of our approach, we first present a vanilla version of the greedy dynamic programming approach applied to (PCSM) that gives a constant-factor approximation and satisfies the packing constraints, but violates the covering constraints by a factor of 2 and works in pseudo-polynomial time.

Vanilla Greedy Dynamic Programming. Let us start with a sketch of the algorithm's definition and analysis. For simplicity of presentation, we assume in the current discussion relating to pseudo-polynomial time algorithms that $\mathbf{C} \in \mathbb{N}_+^{c \times n}$ and $\mathbf{P} \in \mathbb{N}_+^{p \times n}$. Let $\mathbf{p} \in \mathbb{N}_+^p$ and $\mathbf{c} \in \mathbb{N}_+^c$ be the packing and covering requirements, respectively. A solution $S \subseteq \mathcal{N}$ is feasible if and only if $\mathbf{C} \cdot \mathbf{1}_S \geq \mathbf{c}$ and $\mathbf{P} \cdot \mathbf{1}_S \leq \mathbf{p}$. We also use the following notations: $c_{\max} = \|\mathbf{c}\|_\infty$, $p_{\max} = \|\mathbf{p}\|_\infty$, and $[s]_0 = \{0, \ldots, s\}$ for every integer s.

We define our dynamic programming as follows: for every $q \in [n]_0$, $\mathbf{c}' \in [n \cdot c_{\max}]_0^c$, and $\mathbf{p}' \in [p_{\max}]_0^p$ a table entry $T[q, \mathbf{c}', \mathbf{p}']$ is defined and it stores an approximate solution S of cardinality q with $\mathbf{C} \cdot \mathbf{1}_S = \mathbf{c}'$ and $\mathbf{P} \cdot \mathbf{1}_S = \mathbf{p}'$. [8] For the base case, we set $T[0, \mathbf{0}_c, \mathbf{0}_p] \leftarrow \varnothing$. For populating $T[q, \mathbf{c}', \mathbf{p}']$ when $q > 0$, we examine every set of the form $T[q-1, \mathbf{c}' - \mathbf{C}_\ell, \mathbf{p}' - \mathbf{P}_\ell] \cup \{\ell\}$, where ℓ satisfies $\ell \in \mathcal{N} \setminus T[q-1, \mathbf{c}' - \mathbf{C}_\ell, \mathbf{p}' - \mathbf{P}_\ell]$, $\mathbf{c}' - \mathbf{C}_\ell \geq 0$, and $\mathbf{p}' - \mathbf{P}_\ell \geq 0$. Out of all these sets, we assign the most valuable one to $T[q, \mathbf{c}', \mathbf{p}']$. Note that this operation stores a *greedy approximate* solution in the table entry $T[q, \mathbf{c}', \mathbf{p}']$. The output of our algorithm is the best of the solutions $T[q, \mathbf{c}', \mathbf{p}']$, for $1 \leq q \leq n$, $\mathbf{c}' \geq \mathbf{c}/2$ and $\mathbf{p}' \leq \mathbf{p}$. See Algorithm 3 for pseudo code.

[8] We introduce a dummy solution $\perp$ for denoting undefined table entries, and initialize the entire table with $\perp$. For the exact details we refer to [36].

Algorithm 3: Vanilla Greedy Dynamic Program.

create a table $T\colon [n]_0 \times [n \cdot c_{\max}]_0^c \times [p_{\max}]_0^p \to 2^{\mathcal{N}}$ initialized with entries $\perp$
$T[0, \mathbf{0}_c, \mathbf{0}_p] \leftarrow \varnothing$
for $q = 0$ **to** n **do**
  foreach $\mathbf{c}' \in [n \cdot c_{\max}]_0^c$ *and* $\mathbf{p}' \in [p_{\max}]_0^p$ **do**
    foreach $\ell \in \mathcal{N} \setminus T[q, \mathbf{c}', \mathbf{p}']$ **do**
      $\mathbf{c}'' \leftarrow \mathbf{c}' + \mathbf{C}_\ell$, $\mathbf{p}'' \leftarrow \mathbf{p}' + \mathbf{P}_\ell$
      $T[q+1, \mathbf{c}'', \mathbf{p}''] \leftarrow \arg\max\{f(T[q+1, \mathbf{c}'', \mathbf{p}'']), f(T[q, \mathbf{c}', \mathbf{p}'] \cup \{\ell\})\}$
Output $\operatorname{argmax}_{q, \mathbf{c}' \geq \mathbf{c}/2, \mathbf{p}' \leq \mathbf{p}} f(T[q, \mathbf{c}', \mathbf{p}'])$.

Let us now sketch the analysis of the above algorithm. Let O be an optimal set solution. We consider an arbitrary permutation of O, say $\{o^1, o^2, \ldots, o^k\}$. Let $O^i = \{o^1, \ldots, o^i\}$ be the set of the first i elements in this permutation and let $O^0 = \emptyset$. We introduce the function $g\colon O \to \mathbb{R}_+$ for denoting the *marginal* value of the elements in O. More precisely, let $g(o^i) = f_{O^{i-1}}(o^i)$. Note that $f(O) = \sum_{\ell \in O} g(\ell)$. Let for any subset $S \subseteq O$, $g(S) = \sum_{\ell \in S} g(\ell)$. We then inductively construct an order $o_1, \ldots, o_k$ of O with the intention of upper bounding for every prefix $O_q = \{o_1, \ldots, o_q\}$ the value $g(O_q)$ in terms of the value $f(S_q)$ of the table entry $S_q := T[q, \mathbf{C1}_{O_q}, \mathbf{P1}_{O_q}]$ *corresponding to* O_q. The construction of the sequence $o_1, \ldots, o_k$ divides $[k]$ into m phases where m is a positive integer parameter. A (possibly empty) phase $i \in [m]$ is characterized by the following property. Consider a prefix O_q and its corresponding table entry S_q. If q is in phase i then there exists an element $o_{q+1} \in O \setminus O_q$ such that adding o_{q+1} to S_q increases f by at least an amount of $(1 - i/m)g(o_{q+1})$. We set $O_{q+1} = O_q \cup \{o_{q+1}\}$. Thus, in earlier phases we make more progress in the corresponding dynamic programming solution S_q relative to $g(O_q)$ than in later phases. Additionally, we can prove a complementing inequality. At the end of phase $i \in [m]$ all elements in $O \setminus O_q$ increase f by no more than $(1 - i/m)g(o_{q+1})$. We prove that this implies that $f(S_q)$ is at least $i/m \cdot g(O \setminus O_q)$ and thus large relatively to the *complement* of O_q. We set up a factor-revealing linear program that constructs the worst distribution of the marginal values over the phases that satisfy the above inequalities. For the purpose of analysis, by scaling, we assume that $f(O) = \sum_{o \in O} g(o) = 1$. The following lemma formalizes the above sketch. It is also the basis for the *factor-revealing* LP below (for its proof refer to [36]).

▶ **Lemma 10.** *Let $m \geq 1$ be an integral parameter. We can pick for each $i \in [m]$ a set $O_i = \{o_1^i, o_2^i, \ldots, o_{q_i}^i\} \subseteq O$ (possibly empty) such that the following holds. For $i \neq j$, we have that $O_i \cap O_j = \varnothing$. Let $L_i = \sum_{j=1}^i q_j$, $Q_i = \cup_{j=1}^i O_j$, $\mathbf{c}_i = \mathbf{C1}_{Q_i}$, $\mathbf{p}_i = \mathbf{P1}_{Q_i}$ and let $A_i := T[L_i, \mathbf{c}_i, \mathbf{p}_i]$ be the corresponding DP cell. Then $\mathbf{C1}_{A_m} \geq \mathbf{c}/2$ and the following inequalities hold.*

1. $f(A_0) = g(O_0)$ *where* $A_0 = O_0 = \varnothing$,
2. $f(A_i) \geq f(A_{i-1}) + (1 - i/m)g(O_i)$ $\forall i \in [m]$ *and*
3. $f(A_i) \geq \frac{i}{m}\left(1 - \sum_{j \leq i} g(O_j)\right)$ $\forall i \in \{0\} \cup [m]$.

Below we describe a *factor-revealing* LP that captures the above-described multi-phase analysis for the greedy DP algorithm. The idea is to introduce variables for the quantities in the inequalities in the previous lemma and determining the minimum ratio that can be

guaranteed by these inequalities.

$$\begin{aligned}
&\min\ a_m && \text{s.t.} && && \text{(LP)}\\
&\quad a_1 \geq \left(1-\tfrac{1}{m}\right) o_1; && && && (5)\\
&\quad a_i \geq a_{i-1} + \left(1-\tfrac{i}{m}\right) o_i && \forall i \in [m]\setminus\{1\}; && (6)\\
&\quad a_i \geq \tfrac{i}{m}\left(1-\textstyle\sum_{j\leq i} o_j\right) && \forall i \in [m]; && (7)\\
&\quad a_i \geq 0,\ o_i \geq 0 && \forall i \in [m]. && (8)
\end{aligned}$$

The variable o_i corresponds to the marginal value $g(O_i)$ for the set O_i in our analysis. Variables a_i correspond to the quantities $f(A_i)$ for the approximate solution A_i for each phase $i = 1, 2, \ldots, m$. We add all the inequalities we proved in Lemma 10 as the constraints for this LP. Note that since $f(O) = 1$, the minimum possible value of a_m will correspond to a lower bound on the approximation ratio of our algorithm.

The following is the dual for the above LP.

$$\begin{aligned}
&\max\ \textstyle\sum_{i=1}^{m} \frac{i}{m} y_i && \text{s.t.} && && \text{(DP)}\\
&\quad x_i + y_i - x_{i+1} \leq 0 && && \forall i \in [m-1]; && (9)\\
&\quad x_m + y_m \leq 1; && && && (10)\\
&\quad \textstyle\sum_{j\geq i} \frac{j}{m} y_j - (1-\frac{i}{m}) x_i \leq 0 && \forall i \in [m]; && (11)\\
&\quad x_i \geq 0,\ y_i \geq 0 && \forall i \in [m]. && (12)
\end{aligned}$$

This linear program gives for every m a lower bound on the approximation ratio. Analytically, we can show that if m tends to infinity the optimum value of the LP converges to $1/e$. This leads to the following lemma (for its proof refer to [36]).

▶ **Lemma 11.** *Assuming p and c are constants, the vanilla greedy dynamic programming algorithm for* (PCSM) *runs in pseudo-polynomial time $O(n^2 p_{\max} c_{\max})$ and outputs a solution $S \subseteq \mathcal{N}$ that satisfies:* (1) $f(S) \geq (1/e) \cdot f(O)$, (2) $\mathbf{P1}_S \leq \mathbf{p}$ *and* $\mathbf{C1}_S \geq 1/2 \cdot \mathbf{c}$.

Applications and Extensions of Greedy Dynamic Programming Approach

We briefly explain the applications of the approach to the various specific settings and the required tailored algorithmic extensions to the vanilla version of the algorithm.

Scaling, guessing and post-processing for packing constraints. An immediate consequence of Lemma 11 is a *deterministic* $(1/e)$-approximation for the case of constantly many packing constraints that runs in pseudo-polynomial time. We can apply standard scaling techniques to achieve truly polynomial time. This may, however, introduce a violation of the constraints within a factor of $(1 + \varepsilon)$. To avoid this violation, we can apply a pre-processing and post-processing by Kulik et al. [32] to achieve Theorem 4.

Forbidden sets for a single packing and a single covering constraints. In this setting we are able to ensure a $(1 - \varepsilon)$-violation of the covering constraints by using the concept of *forbidden sets*. Intuitively, we exclude the elements of these set from being included to the dynamic programming table in order to be able to complete the table entries to solutions with only small violation.

Fix some $\varepsilon > 0$. By guessing we assume that we know the set G of all, at most $1/\varepsilon$ elements ℓ from the optimum solution with $\mathbf{P}_\ell > \varepsilon \cdot \mathbf{p}$. We can guess G using brute force in $n^{O(1/\varepsilon)}$ time. This allows us to remove all elements with $\mathbf{P}_\ell \geq \varepsilon \cdot \mathbf{p}$ from the instance. Let $\mathcal{N}'$ be the rest of the elements. (For consistency reasons, we use bold-face vector notation here also for dimension one.)

Fix an order of $\mathcal{N}'$ in which the elements are sorted in a non-increasing order of $\mathbf{C}_\ell/\mathbf{P}_\ell$ values, breaking ties arbitrarily. Let $\mathcal{N}_i$ be the set of the first i elements in this order. For any $\mathbf{p}' \leq \mathbf{p}$, let $F_{\mathbf{p}'}$ be the smallest set $\mathcal{N}_i$ with $\mathbf{P}\mathbf{1}_{\mathcal{N}_i} \geq \mathbf{p} - \mathbf{p}'$. Note that the profit of $F_{\mathbf{p}'}$ is at least the profit of any subset of $\mathcal{N}'$ with packing value at most $\mathbf{p} - \mathbf{p}'$ and that the packing value of $F_{\mathbf{p}'}$ is no larger than $(1+\varepsilon)\mathbf{p} - \mathbf{p}'$. Also note that for any $0 \leq \mathbf{p}' \leq \mathbf{p}'' \leq \mathbf{p}$, it holds that $F_{\mathbf{p}''} \subseteq F_{\mathbf{p}'}$.

Now we explain the modified Greedy-DP that incorporates the guessing and the forbidden sets ideas. Let G be the set of the guessed big elements as described above. For the base case, we set $T[\mathbf{C}\mathbf{1}_G, \mathbf{P}\mathbf{1}_G] = G$ and $T[\mathbf{c}', \mathbf{p}'] = \perp$ for all table entries with $\mathbf{c}' \neq \mathbf{C}\mathbf{1}_G$ or $\mathbf{p}' \neq \mathbf{P}\mathbf{1}_G$.

In order to compute $T[\mathbf{c}', \mathbf{p}']$, we look at every set of the form $T[\mathbf{c}' - \mathbf{C}_\ell, \mathbf{p}' - \mathbf{P}_\ell] \cup \{\ell\}$, where $\ell \in \mathcal{N} \setminus (T[\mathbf{c}' - \mathbf{C}_\ell, \mathbf{p}' - \mathbf{P}_\ell] \cup F_{\mathbf{p}'})$, $\mathbf{c}' - \mathbf{C}_\ell \geq 0$, and $\mathbf{p}' - \mathbf{P}_\ell \geq 0$. Notice that we forbid elements belonging to $F_{\mathbf{p}'}$ to be included in any table entry of the form $T[\mathbf{c}', \mathbf{p}']$. Now out of all these sets, we assign the most valuable set to $T[\mathbf{c}', \mathbf{p}']$. The output of our algorithm is the best of the solutions $T[\mathbf{c}', \mathbf{p}'] \cup F_{\mathbf{p}'}$, such that $\mathbf{c}' + \mathbf{C}\mathbf{1}_{F_{\mathbf{p}'}} \geq \mathbf{c}$.

By means of a more sophisticated factor-revealing LP, we obtain Theorem 5. Finally, if the packing constraint is actually a cardinality constraint we can assume that $\varepsilon < 1/\mathbf{p}$. Hence, there will be no violation of the cardinality constraint and also guessing can be avoided.

5 Extensions: Matroid Independence and Multi-Objective

Refer to [36] for the extensions that deal with a matroid independence constraint and with multiple objectives.

References

1 A. A. Ageev and M. I. Sviridenko. An 0.828 Approximation Algorithm for the Uncapacitated Facility Location Problem. *Discrete Appl. Math.*, 93(2-3):149–156, July 1999.

2 Shabbir Ahmed and Alper Atamtürk. Maximizing a class of submodular utility functions. *Mathematical Programming*, 128(1):149–169, June 2011.

3 Per Austrin, Siavosh Benabbas, and Konstantinos Georgiou. Better Balance by Being Biased: A 0.8776-Approximation for Max Bisection. *ACM Trans. Algorithms*, 13(1):2:1–2:27, 2016.

4 Francis Bach. *Learning with Submodular Functions: A Convex Optimization Perspective*. Now Publishers Inc., Hanover, MA, USA, 2013.

5 L. Bordeaux, Y. Hamadi, and P. Kohli. *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, 2014.

6 Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular Maximization with Cardinality Constraints. In *Proc. 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*, pages 1433–1452, 2014.

7 Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a Monotone Submodular Function Subject to a Matroid Constraint. *SIAM J. Comput.*, 40(6):1740–1766, December 2011.

8 Chandra Chekuri and Sanjeev Khanna. A Polynomial Time Approximation Scheme for the Multiple Knapsack Problem. *SIAM Journal on Computing*, 35(3):713–728, 2005.

9 Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Dependent Randomized Rounding via Exchange Properties of Combinatorial Structures. In *Proc. 51th Annual IEEE Symposium on Foundations of Computer Science (FOCS'10)*, pages 575–584, 2010.

10 Reuven Cohen, Liran Katzir, and Danny Raz. An Efficient Approximation for the Generalized Assignment Problem. *Inf. Process. Lett.*, 100(4):162–166, November 2006.

11 Gerard Cornuejols, Marshall Fisher, and George L. Nemhauser. On the Uncapacitated Location Problem. In *Studies in Integer Programming*, volume 1 of *Annals of Discrete Mathematics*, pages 163–177. Elsevier, 1977.

12 Gerard Cornuejols, Marshall L. Fisher, and George L. Nemhauser. Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms. *Management Science*, 23(8):789–810, 1977.

13 Shaddin Dughmi, Tim Roughgarden, and Mukund Sundararajan. Revenue Submodularity. *Theory of Computing*, 8(1):95–119, 2012.

14 U. Feige and J. Vondrak. Approximation algorithms for allocation problems: Improving the factor of 1 - 1/e. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 667–676, 2006.

15 Uriel Feige. A Threshold of Ln N for Approximating Set Cover. *J. ACM*, 45(4):634–652, July 1998.

16 Lisa Fleischer, Michel X Goemans, Vahab S Mirrokni, and Maxim Sviridenko. Tight approximation algorithms for maximum general assignment problems. In *Proc. 17th annual ACM-SIAM Symposium on Discrete Algorithm, (SODA'06)*, pages 611–620. SIAM, 2006.

17 Alan Frieze and Mark Jerrum. Improved approximation algorithms for MAX k-CUT and MAX BISECTION. In Egon Balas and Jens Clausen, editors, *Integer Programming and Combinatorial Optimization*, pages 1–13. Springer Berlin Heidelberg, 1995.

18 J. Gillenwater. *Approximate Inference for Determinantal Point Processes*. PhD thesis, University of Pennsylvania, 2014.

19 Jennifer Gillenwater, Alex Kulesza, and Ben Taskar. Near-Optimal MAP Inference for Determinantal Point Processes. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2735–2743. Curran Associates, Inc., 2012.

20 Michel X. Goemans and David P. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. ACM*, 42(6):1115–1145, November 1995.

21 Eran Halperin and Uri Zwick. Combinatorial Approximation Algorithms for the Maximum Directed Cut Problem. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, pages 1–7, 2001.

22 Jason Hartline, Vahab Mirrokni, and Mukund Sundararajan. Optimal Marketing Strategies over Social Networks. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 189–198, 2008.

23 Johan Håstad. Clique is hard to approximate within $n^{(1-\varepsilon)}$. In *Acta Mathematica*, pages 627–636, 1996.

24 Johan Håstad. Some Optimal Inapproximability Results. *J. ACM*, 48(4):798–859, July 2001.

25 Xinran He and David Kempe. Stability of Influence Maximization. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1256–1265, 2014.

26 Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, 1972.

27 David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the Spread of Influence through a Social Network. *Theory of Computing*, 11(4):105–147, 2015.

28 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal Inapproximability Results for MAX-CUT and Other 2-Variable CSPs? *SIAM Journal on Computing*, 37(1):319–357, 2007.

29 Samir Khuller, Anna Moss, and Joseph Seffi Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

30 Alex Kulesza and Ben Taskar. Learning Determinantal Point Processes. In *UAI 2011, Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, Barcelona, Spain, July 14-17, 2011*, pages 419–427, 2011.

31 Alex Kulesza and Ben Taskar. Determinantal Point Processes for Machine Learning. *Foundations and Trends in Machine Learning*, 5(2-3):123–286, 2012.

32 Ariel Kulik, Hadas Shachnai, and Tami Tamir. Approximations for Monotone and Non-monotone Submodular Maximization with Knapsack Constraints. *Mathematics of Operations Research*, 38(4):729–739, 2013. preliminary version appeared in SODA'09.

33 Jon Lee, Vahab S Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In *Proc. 41st Annual ACM Symposium on Theory Of Computing, (STOC'09)*, pages 323–332. ACM, 2009.

34 Hui Lin and Jeff Bilmes. Multi-document Summarization via Budgeted Maximization of Submodular Functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 912–920, 2010.

35 Hui Lin and Jeff Bilmes. A Class of Submodular Functions for Document Summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 510–520, 2011.

36 Eyal Mizrachi, Roy Schwartz, Joachim Spoerhase, and Sumedha Uniyal. A Tight Approximation for Submodular Maximization with Mixed Packing and Covering Constraints. *CoRR*, abs/1804.10947, 2018. `arXiv:1804.10947`.

37 George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Math. Program.*, 14(1):265–294, 1978.

38 George L Nemhauser and Leonard A Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operations research*, 3(3):177–188, 1978.

39 Andreas S. Schulz and Nelson A. Uhan. Approximating the least core value and least core of cooperative games with supermodular costs. *Discrete Optimization*, 10(2):163–180, 2013.

40 Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Oper. Res. Lett.*, 32(1):41–43, 2004.

41 Jan Vondrák. Symmetry and Approximability of Submodular Maximization Problems. In *Proc. 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS'09)*, pages 651–670. IEEE, 2009.

42 Laurence A. Wolsey. Maximising Real-Valued Submodular Functions: Primal and Dual Heuristics for Location Problems. *Mathematics of Operations Research*, 7(3):410–425, 1982.

Scheduling to Approximate Minimization Objectives on Identical Machines

Benjamin Moseley
Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA, USA
Relational AI, Berkeley, CA, USA
moseleyb@andrew.cmu.edu

Abstract

This paper considers scheduling on identical machines. The scheduling objective considered in this paper generalizes most scheduling minimization problems. In the problem, there are n jobs and each job j is associated with a monotonically increasing function g_j. The goal is to design a schedule that minimizes $\sum_{j\in[n]} g_j(C_j)$ where C_j is the completion time of job j in the schedule. An $O(1)$-approximation is known for the single machine case. On multiple machines, this paper shows that if the scheduler is required to be either non-migratory or non-preemptive then any algorithm has an unbounded approximation ratio. Using preemption and migration, this paper gives a $O(\log\log nP)$-approximation on multiple machines, the *first* result on multiple machines. These results imply the first non-trivial positive results for several special cases of the problem considered, such as throughput minimization and tardiness.

Natural linear programs known for the problem have a poor integrality gap. The results are obtained by strengthening a natural linear program for the problem with a set of covering inequalities we call *job cover inequalities*. This linear program is rounded to an integral solution by building on quasi-uniform sampling and rounding techniques.

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

Keywords and phrases Scheduling, LP rounding, Approximation Algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.86

Category Track A: Algorithms, Complexity and Games

Funding *Benjamin Moseley*: Supported in part by a Google Research Award, a Infor Award and NSF Grants CCF-1824303, CCF-1733873 and CCF-1845146.

1 Introduction

A common optimization challenge is scheduling a set of n jobs on m identical machines to optimize the quality of service delivered to the jobs. The quality of service objective could be: a delay based objective, such as minimizing the average waiting time; a fairness objective ensuring resources are shared fairly between jobs, such as the ℓ_2-norm of the waiting time; or a real-time objective such as ensuring a small number of jobs are not completed by their deadline.

Scheduling Model. This paper develops an algorithm that has strong guarantees for most reasonable objectives. This work considers the *identical* machines setting where all jobs are available at the same time. Each job j has a processing time p_j. The job can be processed on m identical machines where the processing time of the job is the same on all machines. This work assumes that *preemption* and *migration* are allowed. That is jobs can be stopped and resumed at a later time, possibly on a different machine.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 86; pp. 86:1–86:14


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

This paper initiates the study of the general scheduling problem (GSP) on identical machines. In this problem, each job j has a function $g_j(t) : \mathbb{R}^+ \to \mathbb{R}^+$. The value of $g_j(t)$ specifies the cost of completing job j at time t. The goal is to design an algorithm that completes each job j at time C_j to minimize $\sum_{j\in[n]} g_j(C_j)$. No assumptions on the functions are made except that they are positive and non-decreasing, so there never is an incentive to have a job wait longer to be completed. Note that each job has its own, *individual*, cost function. In several systems, it is the case that jobs can be associated with distinct cost functions [15, 16, 17].

The problem generalizes many scheduling objectives. Examples include the following. In the following descriptions, each job j has a positive weight w_j denoting its priority.

- **Weighted Completion Time:** A job's cost is its weight multiplied by its completion time. The completion time is how long the job waits in the system and this objective focuses on minimizing the priority scaled average waiting time. This objective is captured by setting $g_j(t) = w_j \cdot t$.
- **Weighted kth Norm of Completion Time:** This objective focuses on minimizing $\sqrt[k]{\sum_{j\in[n]} w_j C_j^k}$ or, by removing the outer kth root, $\sum_{j\in[n]} w_j C_j^k$. This objective is captured by setting $g_j(t) = w_j \cdot t^k$. This is used to enforce fairness in the schedule and typically $k \in \{2, 3\}$.
- **Weighted Throughput Minimization:** The goal is to minimize the weighted number of jobs that miss their deadline. Each job j has a deadline d_j. Setting $g_j(t) = 0$ for $t \leq d_j$ and w_j otherwise gives this objective.
- **Weighted Tardiness:** Each job has no cost if completed before its deadline and otherwise the job pays its weighted waiting time after is deadline. Each job has a deadline d_j and weight w_j. This objective is obtained by setting $g_j(t) = 0$ for $t \leq d_j$ and $w_j(t - d_j)$ otherwise.
- **Exponential Completion Time:** In this objective a job's cost grows exponentially with its completion time. The objective is captured by setting $g_j(t) = w_j \cdot \exp(t)$.

These problems have been challenging to understand. The problem considered is NP-Hard, even on a single machine and the cost functions are piecewise linear [10]. It is known that Smith's rule is optimal for minimizing the total weighted completion time [21]. Bansal and Pruhs in a breakthrough result introduced a $O(1)$-approximation algorithm for the general scheduling problem on a single machine [2]. Cheung et al. improved this to show a $(4 + \epsilon)$ approximation [6, 7, 18]. Antoniadis et al. gave a quasi-polynomial time approximation scheme on a single machine [1, 11].

The next step in this line of work is to generalize these techniques to multiple machine environments, but there is a clear barrier when generalizing past approaches to multiple machines. Prior work introduced a strong linear program that uses a polynomial number of knapsack cover inequalities. See [3] for details on knapsack cover inequalities. The inequalities in [2, 7, 18] are weak in multiple machine environments and result in linear programs with an unbounded integrality gap.

An open question is if there exists a linear program with a small integrality gap for multiple machines. Further, are there good approximation algorithms for the GSP on multiple machines.

Results. This paper studies the GSP in the identical machine environment. The paper shows the following theorem. The technical contributions that result in this theorem are the derivation of valid strong linear program inequalities that are used to strengthen a natural linear program relaxation of the problem and an iterative rounding technique that builds on quasi-uniform sampling [22].

▶ **Theorem 1.** *There is a randomized algorithm that achieves a $O(\log \log nP)$ approximation in expectation and runs in expected polynomial time for the GSP on multiple identical machines with preemption and migration where P is the ratio of the maximum to minimum job size.*

A natural question is if preemption and migration are necessary for an algorithm to have a good approximation ratio. This paper shows that they are by establishing that any scheduler required to be non-preemptive or non-migratory has an unbounded approximation ratio unless $P = NP$. The proof is omitted

▶ **Theorem 2.** *The approximation ratio of any algorithm for GSP is unbounded unless P=NP if either the algorithm is required to be non-migratory or non-preemptive on m identical machines.*

Overview of Technical Contributions. The main result is enabled by a set of strengthening inequalities added to a natural linear program (LP) for the problem. The paper calls these inequalities, **job cover inequalities**. See Section 4. These inequalities are needed because without them the LP introduced in this paper has an unbounded integrality gap even if all jobs arrive at the same time on a single machine[1]. Other natural LP relaxations, such as a time indexed LP, also have an unbounded gap even on a single machine [2, 7].

Prior work on a single machine also used a set of covering inequalities to strengthen a linear program. These inequalities consider every interval I and the set of jobs that arrive during the interval S_I. A constraint states that the total processing time of jobs in S_I that are completed after I ends must be greater than the total processing time of jobs in S_I minus the length of I [2, 7]. This is a covering constraint ensuring that the jobs arriving during I that complete during I have total size at most the length of I. These covering constraints are strengthened using knapsack cover inequalities. If such constraints are satisfied integrally then the Earliest-Deadline-First algorithm can be used to construct a schedule of the same cost as the LP.

A natural idea to extend this to identical machines is to use the same constraint, but the total work completed after I ends must be greater than the size of jobs in S_I minus m times the length of I. This generalization takes into account that each machine can be busy during I. Then the natural next step is to use knapsack cover inequalities to strengthen this new set of constraints. Unfortunately, it is easy to show such inequalities are insufficient and result in an LP with a large integrality gap. There are several issues and they are all rooted in the fact that this does not take into account that a job can only be processed on one machine at any point in time. To overcome this shortcoming, this paper considers covering constraints used to strengthen a minimum cut constraint arises from a natural bipartite flow problem.

This paper proceeds by first reducing the scheduling problem to the problem of finding completion times for each of the jobs, without committing to a schedule. Once a feasible set of completion times is discovered, the scheduling of jobs can easily be obtained by solving a bipartite flow problem. See Section 3 for details. Feasible solutions to the bipartite flow problem have a one-to-one correspondence to the original scheduling problem. We note that the reduction to this flow problem is a well-known scheduling technique.

The bipartite graph in the flow problem is used to derive the job cover inequalities. First an LP is written based on the flow problem. To ensure a feasible flow is possible, a set of constraints is added that ensure the minimum cut in the graph is sufficiently large. Then this

[1] Without strengthening inequalities and when jobs arrive at the same time on a single machine the LP introduced in this paper can be reduced to an LP used in prior work where the gap is known [2, 7]

set of covering constraints are strengthened. While these inequalities are used to strengthen constraints that arise from a flow graph, they are different than previously studied flow cover inequalities [19, 20, 14]. The key to defining the improved constraints is leveraging the structure of the minimum cuts in the bipartite graph resulting from the scheduling problem.

The algorithm solves the strong LP and rounds the solution. The idea is to use iterative randomized rounding, but this results in a large approximation ratio. We remark that standard randomized rounding techniques can be used to obtain a $O(\log n)$ approximation by over sampling variables by a $O(\log n)$ factor and then union bounding over constraints to show they are satisfied with high probability. However, there are issues with reducing the approximation ratio below this factor; the most challenging is showing that the constraints are satisfied if variables are sampled by a smaller factor, which is what would be needed to reduce the approximation ratio.

Instead, the algorithm uses an iterative scheme to round the solution. In each iteration, the algorithm over samples variables by a small factor and modifies the linear program. The modification ensures that (1) in expectation no variable, over all iterations, is sampled by more than a $O(\log \log nP)$ factor than how much it is selected by the optimal LP solution and (2) the relaxation remains feasible. The scheme builds on techniques of quasi-uniform sampling [4, 22] and quasi-uniform iterative rounding [12].

Other Related Work

Single Machine. It is known the Smith's rule is optimal for minimizing the total weighted completion time [21]. The tardiness problem has been challenging to understand. Lawer [13] gave a polynomial approximation scheme if jobs have unit size. Before the work on the GSP in the single machine environment, the previously best known approximation for arbitrary sized jobs was a $(n-1)$-approximation [5]. For the GSP a $(4+\epsilon)$-approximation is known [6] and a quasi-polynomial time approximation scheme has been established [1].

Identical Machines. Smith's rule is optimal for minimizing the total weighted completion time [21]. It is not difficult to see that there is an optimal algorithm for makespan[2] if jobs can be preempted and migrated across machines. An PTAS is known if migration is disallowed [9]. As far as the author is aware, there are no non-trivial results known for exponential completion time and tardiness on multiple machines.

2 Preliminaries

In the general scheduling problem (GSP), there is a set J of n jobs. Each job i has integer processing time p_i. Let P denote the maximum job size and assume the minumum job size is one. The jobs are to be scheduled on a set M of m identical machines that can schedule one job at any point in time. It is assumed that time is slotted and job i must be scheduled for p_i time units over all machines. Jobs can be preempted and migrated across machines. Jobs cannot be scheduled on more than one machine simultaneously. Every job i is associated with a function $g_i : \mathbb{R}^+ \to \mathbb{R}^+$ where $g_i(t)$ specifies the cost of completing job i at time t. Without loss of generality, assume that $g_i(0) = 0$ for all jobs i. The only assumption on $g_i(t)$ is that it is a non-negative non-decreasing function. Under a given schedule, job i is completed at time C_i. The goal is for the scheduler to minimize $\sum_{i \in J} g_i(C_i)$. Note that it can be assumed that all jobs are completed by time nP.

[2] Makespan is equivalent to minimizing the maximum completion.

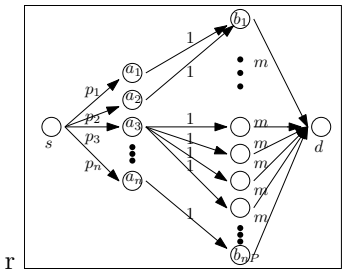


Figure 1 The graph G. A job i needs to assign p_i units of flow (processing time) to machines before C_i. Machines can process up to m units at each time.

3 Scheduling Jobs with Deadlines

This section shows that if the completion times of the jobs are fixed then there is a method to determine how to schedule the jobs at or before their completion times or determine if such a schedule is not possible. Notice that if such a schedule is feasible then this ensures the objective is either the same or smaller in the computed schedule than if all jobs are completed at exactly their given completion times. Let job i have a given completion time C_i. The completion time C_i is interpreted as job i's deadline.

The method to construct a schedule for the jobs is to setup a flow problem. Setting up a flow graph to determine if a set of jobs can be feasibly scheduled is a standard scheduling technique (e.g. [8]), but is presented here so that later this graph can be used in a linear program formulation. Consider creating a bipartite flow graph $G = (\{s, d\} \cup A \cup B, E)$ where A contains a node a_i for every job i and B contains a node b_t for every time step t. There is additionally a source s with an outgoing edge to each node a_i in A with capacity p_i. There is a sink node d that has an incoming edge from each node in B with capacity m. Finally there is an edge from $a_i \in A$ to $b_t \in B$ of capacity 1 if and only if $t \leq C_i$. See Figure 1.

A set of completion times are feasible if and only if there is a feasible flow in this network of value $\sum_{i \in [n]} p_i$. This is because a job can be scheduled for a unit at each time during $[0, C_i]$ and must be scheduled for p_i units total. Further, every time step can schedule up to m jobs.

There are two messages to takeaway from this. One is that the problem can be solved by only knowing completion times for the jobs. The other is that this flow graph can be used to determine if a set of completion times can be associated with a valid schedule. A set of completion times are said to be **valid** if there is a schedule that completes each job only earlier than the given completion time.

The following theorem follows from the construction of G.

▶ **Theorem 3.** *A set of completion times is valid if and only if there is a feasible maximum flow of value $\sum_{i=1}^{n} p_i$ in the flow graph G.*

Given a set of valid completion times, one can construct a feasible schedule using the flow graph. That is, an assignment of jobs to machine at each time step. Unfortunately, the graph has size $\Omega(nP)$ and could be exponential in size. Recall that P is the ratio of the maximum to minimum job size. There is a polynomial time algorithm that constructs a feasible schedule in time polynomial in n and $\log W$ given a set of valid completion times. Here W denotes the maximum value of $g_j(t)$ for $t \leq nP$. This algorithm is omitted due to space. It can be obtained by rounding possible completion times to geometrically increasing times.

4 Strengthened Linear Program with Job Cover Inequalities

Consider the following natural integer program. The constraints in the program come from the flow graph of the prior section. Let $x_{j,t}$ be 1 if job j is not completed at time t and 0 otherwise.[3] This implies that $x_{j,t}$ is continuously 1 for t less than j's completion time and then 0 for times t thereafter in an integer solution. Let T be the set of all time slots and J the set of all jobs. By assuming $g_j(0) = 0$ for all j, the objective function is a telescoping summation for each job j whose value will be $g_j(t)$ if t is the last time j is not fully processed. The first set of constraints says that if job j is completed at time $t-1$ then it is also completed at time t. The second set of constraints are more involved. One can think of the latest time t that $x_{j,t} = 1$ as being the completion time of j. Given this, the constraint says that every cut in the flow network from Section 3 has value at least $\sum_{i \in J} p_i$. This is a valid constraint by Theorem 3 and the maximum-flow minimum-cut theorem. Note that J' corresponds to jobs whose nodes are on the side of the cut with the source s. Similarly, T' corresponds to time steps whose nodes are on the side of the cut with the sink d.

$$\min \sum_{j \in J} \sum_{t \in T} x_{j,t}(g_j(t) - g_j(t-1)) \tag{1}$$

$$\text{s.t.} \quad x_{j,t} \leq x_{j,t-1} \qquad \forall j \in J, t \in T \tag{2}$$

$$\sum_{j \in J \setminus J'} p_j + \sum_{j \in J'} \sum_{t \in T'} x_{j,t} + \sum_{t \in T \setminus T'} m \geq \sum_{j \in J} p_j \qquad \forall J' \subseteq J, T' \subseteq T \tag{3}$$

$$x_{j,t} \in \{0,1\} \qquad \forall j \in J, t \in T$$

The goal is to derive a set of valid strengthening inequalities for the IP (1). These inequalities are used to strengthen the minimum cut constraints. These inequalities are needed because without them the LP has an unbounded integrality gap even if all jobs arrive at the same time on a single machine [7]. Other natural LP relaxations, such as a time indexed LP also have an unbounded gap even on a single machine.

We can derive a set of strengthening inequalities for this linear program that replace the set of constraints (3). The proof establishing validity of the following constraints is omitted due to space. The strengthening focuses on the constraints in (3) for $J' = J$. In the end, the derived constraints are strictly stronger than the above and one need not consider the other sets J'.

Fix $J' = J$ and consider the constraints $\sum_{j \in J} \sum_{t \in T'} x_{j,t} + \sum_{t \in T \setminus T'} m \geq \sum_{j \in J} p_j$ for all $T' \subseteq T$. For a job j and a collection of time steps T' let $E(T', j)$ be the set of up to $\min\{p_j, \sum_{i \in J} p_i - m|T \setminus T'|\}$ **earliest** time steps in T'. The full analysis first shows that we can strengthen this to

$$\sum_{j \in J} \sum_{t \in E(j,T')} x_{j,t} + \sum_{t \in T \setminus T'} m \geq \sum_{j \in J} p_j \text{ for all } T' \subseteq T.$$

Notice the first summation now only considers time steps in $E(T', j)$ for each job j.

The proof further improves these inequalities by taking inspiration from knapsack cover inequalities. Let D be a **vector** where D_j is a time corresponding to job j. Intuitively, D_j is a lower bound on the completion time for job j. The constraints below say that even if all jobs j are set to have completion times **at least** D_j then the constraints should still be satisfied.

[3] Note that this is *not* the standard time indexed LP where the variables represent the amount j is processed at time t.

Fix $T' \subseteq T$ and a vector D of completion times for every job. Consider the constraint for T' in the above set of inequalities. If job j is given a completion time of at least D_j then j will contribute at least $\sum_{t \in [0,D_j] \cap E(T',j)} 1$ to the left side of the inequality. Let $V(T',D) = \sum_{i \in J} p_i - \sum_{t \in T \setminus T'} m - \sum_{j \in J} \sum_{t \in [0,D_j] \cap E(T',j)} 1$. Let $E(T',D,j)$ be the **earliest** $V(T',D)$ time steps in $E(T',j)$ later than D_j. The new constraints are as follows. These are the **job cover inequalities**.

$$\sum_{j \in J} \sum_{t \in E(T',D,j)} x_{j,t} \geq V(T',D) \quad \forall T' \subseteq T, \forall D, V(T',D) > 0 \tag{4}$$

The validity of these inequalities is not difficult to show, but technical. Notice that the number of inequalities is large. We can somewhat reduce the number of constraints as follows. It can be established that any *integer* solution satisfies all the constraints in (4) if and only if the following are satisfied. The proof is omitted.

$$\sum_{j \in J} \sum_{t \in E(T',D,j)} x_{j,t} \geq V(T',D) \quad \forall b \in [0,\infty], T' = [b,\infty], \forall D, V(T',D) > 0 \tag{5}$$

These constraints state that if the subset of constraints in (4) are satisfied for any D and all sets T' that consist of a continuous set of time steps from some time b to time ∞ then all of the constraints in (4) are satisfied (for any T'). Due to this, we will only need to use the constraints in (5) that restricts the sets T'.

The constraints (3) in the IP are replaced by the constraints in (5). Throughout the paper these constraints are discussed and it is said that a fixed constraint is defined by the set $T' = [b,\infty]$ and a vector D.

Note on Solving the LP. The IP is relaxed to a LP. The LP is solved and then subsequently rounded to an integer solution. There are an exponential number of constraints. To solve the LP, the ellipsoid method is used. The author does not know of an efficient separation oracle for the set of constraints in (4). The reduced set of constraints in (5) are the only constraints needed for the analysis. For this set of constraints, an efficient dynamic programming algorithm can be used as a separation oracle. The separation oracle has been omitted due to space constraints.

5 The Rounding Algorithm

In this section the algorithm for rounding a fractional LP solution to an integral solution is described. Recall in Section 3 it was shown how to assign jobs to machines and time slots if a valid set of completion times have been established. The remaining goal is to design an algorithm that rounds a fractional LP solution to an integral solution, giving the completion times for the jobs.

The algorithm takes as input a fractional solution to the LP x'. The input solution is for the LP given in the previous section with constraints (3) replaced with the derived constraints (5). The algorithm rounds the x' solution to an *integral* solution x^*. If this solution is feasible, then the algorithm terminates. Otherwise x^* is modified to obtain a new feasible fractional solution $\tilde{x}^*$ that the algorithm recurses on.

Informal Algorithm Description and Intuition. The algorithm runs in phases. During a phase the algorithm finds a completion time for each job. The completion times are pushed back to later times in each phase. In a fixed phase the algorithm runs a randomized rounding

procedure to sample completion times for the jobs. Roughly, each completion time will be over sampled by a $\Theta(\log \log nP)$ factor over the fractional part of the LP. After sampling an integer solution x^* is created and variables are set corresponding to the sampled completion times of the jobs. Some of the constraints in the LP will be satisfied. These constraints will always remain satisfied because each job's completion time is only pushed back to a later time in subsequent iterations. The algorithm needs to satisfy the remaining constraints, which it does by recursing on a fractional LP solution $\tilde{x}^*$ to make completion times later. The fractional solution $\tilde{x}^*$ is constructed by increasing variables in the integral solution x^*.

In each iteration, the cost of the sampling is bounded by a $\Theta(\log \log nP)$ factor more than the *fractional* portion of the linear program objective in expectation. Ideally, one can use standard iterative rounding for the recursion. However, naive approaches could have large cost as it is difficult to bound the number of times the algorithm recurses and therefore difficult to bound how many times a variable in the LP is sampled. A standard approach will result in a $\Theta(\log nP)$ approximation.

The idea is to only include some fractional variables in the linear program solution that the algorithm recurses on. The variables that are included can be slightly larger than their original value. The essential properties are that (1) the linear program is feasible and (2) each fractional variable is set to 0 with constant probability. Using (2) it will be shown that the expected value of each variable drops by a constant (e.g. $\frac{1}{2}$) factor of its value at the beginning of the iteration. If this is established, then the probability a completion time is sampled decreases geometrically over the phases and we can bound the algorithm's objective by the cost of the sampling in the first phase, a $\Theta(\log \log nP)$ factor within the original LP object in expectation.

The recursion will determine fractional variables to set to satisfy all constraints. One can think of the fractional variables as **fractional completion times** for the jobs. The idea is to associate each constraint with a set of fractional completion times that are *critical* for satisfying the constraint in the solution x'. This will not be all completion times used to satisfy the constraint, but an essential subset of them. By slightly increasing the linear program variables in the integral solution x^* to get a solution $\tilde{x}^*$ it is the case that only the critical completion times are needed to satisfy their corresponding unsatisfied constraints.

If a constraint is unsatisfied, then the solution $\tilde{x}^*$ will set positive fractional values for all completion times critical for this constraint to satisfy it. The analysis will show that each fractional completion time is 0 (not increased) in $\tilde{x}^*$ with constant probability. For a completion time to be 0 in the recursion we need to ensure all constraints are satisfied where the completion time is critical by the integer solution x^*. Unfortunately, a completion time could be critical for many constraints. Due to this, it is insufficient to show each constraint is satisfied with good probability and then union bound over all constraints.

Fix a fractional completion time. The proof establishes that with good probability *every* constraint that the completion time is critical for is satisfied in x^*. This ensures that the the completion is 0 with constant probability. This will be used to show that the expected value of a fractional variable in the LP decreases geometrically over the iterations. A completion time is over-sampled by at most a $\Theta(\log \log nP)$ factor as compared to the original LP solution in expectation over all iterations. The cost is as if only one iteration of uniform sampling occurred. This is similar to the analysis approach used in quasi-uniform sampling [22] and rounding [12].

5.1 Formal Algorithm Description

The ith phase of the algorithm is the following. The algorithm recurses on the following until all constraints are satisfied. Let the input to the first phase be x, the optimal fractional solution to the LP. The algorithm utilizes randomized rounding parameterized by $c \leq 1$. The value of c will be set to be $\frac{1}{\Theta(\log\log nP)}$.

Phase i of the Algorithm. The algorithm first uses **randomized rounding**. Let x' be a feasible fractional solution to the LP at ith phase of the algorithm. For each job j, the algorithm chooses a value $\alpha_j \in [0,1]$ uniformly at random and independently. Let $C_{j,\alpha}$ be the latest time t where $x'_{j,t} \geq c\alpha_j$. Let β'_j be the latest time t where $x'_{j,t} = 1$. Let $\text{LP}'_{\text{int}} = \sum_j g_j(\beta'_j)$ be the total **integral cost** of the LP solution x' and let $\text{LP}'_{\text{frac}} = \sum_{j\in J}\sum_{t>\beta'_j} x'_{j,t}(g_j(t) - g_j(t-1))$ be the total **fractional cost** of the LP solution x'.

The algorithm modifies the LP solution x' to get a new (possibly infeasible) solution x^*. This is further modified to get a feasible solution $\tilde{x}^*$ that the algorithm recurses on. The algorithm sets $x^*_{j,t} = 1$ for all j and t where $t \leq C_{j,\alpha}$ and 0 otherwise. If all constraints are satisfied in the LP, then the algorithm sets $C^*_j = C_{j,\alpha}$ and returns this set of completion times as the final solution. If not, then the algorithm further modifies x^* as described below and recurses on phase $i+1$. Let x^* denote the current integral solution and $\tilde{x}^*$ a fractional solution resulting from the following modification to x^*.

Consider any constraint in (5) defined by a set of time steps T' and a vector of completion times D' that is not satisfied by x^*. Assume D' is chosen so that $D'_j \geq C_{j,\alpha}$ for all j. We only need to consider these constraints because $x_{j,t} = 1$ for $t \leq C_{j,\alpha}$. Recall that we may assume T' contains a continuous set of time steps beginning with some time $t_{T'}$ and going to ∞. The algorithm identifies a set of pairs of jobs and completion times that are fractionally chosen in x' that are "critical" for satisfying this constraint. Intuitively, we will need to include these same fractional completion times in $\tilde{x}^*$.

Let $J_{T',D'}$ be the set of jobs j where $\sum_{t\in E(T',D',j)} x'_{j,t} > 0$. These are jobs used to satisfy the constraint in x'. The following two sets of jobs can be thought of as *not* critical for the constraint.

1. Order the jobs j in $J_{T',D'}$ as $1,2,3,\ldots|J_{T',D'}|$ in decreasing order of D'_j. Let $M_{T',D'} \subseteq J_{T',D'}$ be the smallest prefix of jobs $1,2,\ldots k$ from this order such that $\sum_{j=1}^{k}\sum_{t\in E(T',D',j)} x'_{j,t} \geq \frac{1}{10}V(T',D')$.
2. Order the jobs j in $J_{T',D'}$ as $1,2,3,\ldots,|J_{T',D'}|$ in increasing order of D'_j. Let $L_{T',D'} \subseteq J_{T',D'}$ be the smallest prefix of jobs $1,2,\ldots k$ in this order such that $\sum_{j=1}^{k}\sum_{t\in E(T',D',j)} x'_{j,t} \geq \frac{1}{10}V(T',D')$.

We will say that the jobs in $\mathcal{J}_{T',D'} = J_{T',D'} \setminus (L_{T',D'} \cup M_{T',D'})$ are **critical** for satisfying the constraint T', D'. Let $\mathcal{J}^* = \{j \mid \exists T'D' \text{ s.t. the constraint for } T' \text{ and } D' \text{ is unsatisfied in } x^* \text{ and } j \in \mathcal{J}_{T',D'}\}$. This is the set of all critical jobs for constraints that are not satisfied by x^*.

For each job $j \in \mathcal{J}^*$ set $\tilde{x}^*_{j,t'}$ to be $10x'_{j,t'}$ for all $t' \geq C_{j,\alpha}$. Note that if $c < \frac{1}{10}$ and $C_{j,\alpha} \leq t'$ it is the case that $\tilde{x}^*_{j,t'} \leq c \leq 1/10$, so $\tilde{x}^*$ is in $[0,1]$ as desired. After performing all updates, recurse.

Terminating Condition. The above description states that the algorithm terminates when all constraints are satisfied by the integer solution x^*. The analysis will show that this occurs in polynomial time in expectation.

6 Bounding the Cost and Feasibility of the Algorithm

In this section, the correctness of the algorithm is established and the total cost of the algorithm is bounded as well as the running time. The analysis has several intermediate goals. One is to show that the resulting solution $\tilde{x}^*$ is feasible. Another is to show that the increase in cost of the randomized rounding is bounded by the cost of the fractional part of the LP in one iteration. The final goal is showing that the expected value of the objective for the fractional part of the LP solution decreases significantly in each iteration. After establishing these facts Theorem 3 will prove the main theorem.

Feasibility of the Algorithm. We now establish that $\tilde{x}^*$ is feasible for the LP. This ensures the algorithm constructs a feasible solution. The proof is omitted due to space. The proof follows by the fact that each unsatisfied constraint is associated with variables in the solution x' which are within a constraint multiplicative factor of satisfying the constraint. In the recursion, these variables are included in $\tilde{x}^*$ with their fractional valuesincreased by a factor 10 over x' ensuring their corresponding constraint is satisfied.

▶ **Lemma 4.** *In any iteration of the algorithm, if x' is a feasible LP solution then the algorithm constructs a feasible solution $\tilde{x}^*$ at the end of the iteration for any $c \leq 1/10$.*

Bounding the Cost and Running Time of the Algorithm. This section bounds the cost of the LP solution $\tilde{x}^*$ and the running time of the algorithm. First the cost of the randomized rounding is bounded. This bounds the cost of the *intermediate* integral solution x^*. The following lemma shows that the expected cost *increase* of the LP solution x^* over x' is bounded by $\frac{1}{c}\mathrm{LP}'_{\mathrm{frac}}$. Recall that x^* is the solution obtained by only the randomized rounding part of the algorithm and it is a possibly infeasible integer solution. After this lemma, the cost of the solution $\tilde{x}^*$ is bounded. This is the solution the algorithm recurses on. Combining the cost over the entire algorithm is bounded.

The following lemma bounds the cost of the solution x^*. The proof is omitted. The proof follows by a standard analysis of randomized rounding.

▶ **Lemma 5.** *The total expected different in cost of x^* and x' is at most $\frac{1}{c}\mathrm{LP}'_{\mathrm{frac}}$. That is, $\mathrm{E}[\sum_{j\in J}\sum_t (x^*_{j,t} - x'_{j,t})(g_j(t) - g_j(t-1))] \leq \frac{1}{c}\sum_{j\in J}\sum_t \mathrm{LP}'_{\mathrm{frac}}$.*

Let $\mathrm{LP}^*_{\mathrm{frac}} := \sum_{j\in J}\sum_{t>C_{j,\alpha}} \tilde{x}^*_{j,t}(g_j(t) - g_j(t-1))$ be the fractional cost of $\tilde{x}^*$ and $\mathrm{LP}^*_{\mathrm{int}} = \sum_{j\in J} g(C_{j,\alpha})$ be the integral cost. Note that $\mathrm{LP}^*_{\mathrm{int}}$ is precisely the objective of the integral solution x^*. The key to bounding the cost of the algorithm is to show that the fractional cost of the LP solution decreases by a constant factor in each iteration. This is stated in the following lemma. The proof is deferred due to space. This lemma is the most interesting part of the analysis and the proof is presented in Section 6.1.

▶ **Lemma 6.** *In any iteration of the algorithm, $\mathrm{E}[\mathrm{LP}^*_{\mathrm{frac}}] \leq \frac{1}{4}\mathrm{LP}'_{\mathrm{frac}}$.*

Using this lemma, the total cost of the algorithm can be bounded.

▶ **Lemma 7.** *Let OPT be the optimal feasible objective to the LP. It is the case that the algorithm's total cost is at most $\frac{2}{c}$OPT in expectation.*

Proof. Let $\mathrm{LP}^{\mathrm{i}}_{\mathrm{frac}}$ denote the fractional part of the objective in the LP solution at the beginning of the ith iteration of the algorithm. Note that $\mathrm{LP}^1_{\mathrm{frac}} \leq \mathrm{OPT}$. Inductively, Lemma 6 gives that $\mathrm{E}[\mathrm{LP}^{\mathrm{i}}_{\mathrm{frac}}] \leq \frac{1}{4^i}\mathrm{OPT}$.

Lemma 5 ensures that the integral portion of the LP objective increases by at most $\frac{1}{c}\mathrm{LP}^{\mathrm{i}}_{\mathrm{frac}}$ in each iteration. Thus the total cost can be bounded by $\frac{1}{c}\sum_{i=1}^{\infty}\frac{1}{4^i}\mathrm{OPT} \leq \frac{2}{c}\mathrm{OPT}$. ◀

The following proposition bounds the run time of the algorithm and the proof is omitted. This follows because the previous lemma will ensure the variables in the LP converge to 0 after $O(\log nP)$ iterations.

▶ **Proposition 8.** *The algorithm runs in polynomial time in expectation.*

Lemma 7 bounds the objective of the algorithm. Lemma 4 ensures that the algorithm constructs a feasible solution. Finally, Propostion 8 shows the algorithm runs in polynomial time. Together, this proves the main result, Theorem 1.

6.1 Proof of Lemma 6: Expected Decrease in the Fractional Objective

This section proves Lemma 6 for a fixed iteration of the algorithm. Fix a job j and the fractional solution x' that is input to the rounding algorithm in this iteration. The goal is to show that the probability that j is in $\mathcal{J}^*$ is small and therefore the algorithm only includes integral variables $\tilde{x}^*_{j,t}$ for job j when it recurses. In particular, the goal is to show the following lemma. In the following, n and P are assumed to be sufficiently large.

▶ **Lemma 9.** *Fix any job j^*. With probability at most $\frac{2}{\log^2 nP} \leq 1/40$ it is the case that there is a constraint T', D' unsatisfied by x^* and $j^* \in \mathcal{J}_{T',D'}$ when $c \leq \frac{1}{1000 \log\log nP}$.*

This lemma implies Lemma 6.

Proof of Lemma 6. Fix any job j and an iteration of the algorithm. If there is a constraint T', D' unsatisfied, $j \in \mathcal{J}_{T',D'}$, $x'_{j,t}$ is fractional, $t \geq C_{j,\alpha}$ and $t \in E(T', D', j)$ then the algorithm sets $\tilde{x}^*_{j,t}$ to be $10x'_{j,t}$.

This event increases the cost of $\tilde{x}^*$ by at most $10\sum_{t>C_{j,\alpha}} x'_{j,t}(g_j(t) - g_j(t-1))$ and it happens for some T', D' with probability at most $1/40$. The total expected cost $\mathrm{LP}^*_{\mathrm{frac}}$ is at most the following.

$$10\sum_{j\in J}\sum_{t>C_{j,\alpha}} x'_{j,t}(g_j(t) - g_j(t-1))\Pr[\exists T', D' |\, j \in \mathcal{J}_{T',D'} \text{ and constraint } T', D' \text{ unsatisfied by } x^*]$$

$$\leq 10\sum_{j\in J}\sum_{t>C_{j,\alpha}} x'_{j,t}(g_j(t) - g_j(t-1))\frac{1}{40} \leq \frac{1}{4}\sum_{j\in J}\sum_{t>C_{j,\alpha}} x'_{j,t}(g_j(t) - g_j(t-1)) \quad \text{[Lemma 9]}$$

Thus, the expected cost of $\mathrm{LP}^*_{\mathrm{frac}}$ decreases by a factor $1/4$ over $\mathrm{LP}'_{\mathrm{frac}}$. ◀

The remaining goal of this section is to prove Lemma 9. The proof begins by observing that since the solution x^* is integral if a constraint T' and D' is satisfied for a particular set D' then the constraints for T' and all sets D'' are satisfied. This will allow us to focus on constraints for one special set D'. The proof is omitted.

▶ **Proposition 10.** *Let $t_{T'}$ be some time step and $T' = [t_{T'}, \infty]$. Let D' be set such that D'_j is the latest time t where $x'_{j,t} \geq c$ for all jobs j. If the constraint for T' and D' is satisfied in the solution x^* then all constraints for T' and any set D'' are satisfied in the soultion x^*.*

For the remainder of the proof, fix D' to be as described in the prior lemma. Now we establish some basic propositions on which jobs contribute to a constraint. This will be useful for identifying critical jobs. Note that the two propositions below are different depending on the ordering of the times considered. The proofs are omitted.

▶ **Proposition 11.** *Fix any job j and two sets $T = [t_T, \infty]$ and $T' = [t_{T'}, \infty]$ where $D'_j \leq t_T < t_{T'}$. For any fractional LP solution x satisfying constraints (2) if $V(T, D') \geq \frac{1}{2}V(T', D')$ then it is the case that $\sum_{t \in E(T', D', j)} x_{j,t} \leq 2 \sum_{t \in E(T, D', j)} x_{j,t}$.*

▶ **Proposition 12.** *Fix any job j and two sets $T = [t_T, \infty]$ and $T' = [t_{T'}, \infty]$ and $D'_j > t_T > t_{T'}$. For any fractional LP solution x satisfying constraints (2) if $V(T, D') \geq \frac{1}{2}V(T', D')$ then $\sum_{t \in E(T', D', j)} x_{j,t} \leq 2 \sum_{t \in E(T, D', j)} x_{j,t}$.*

The next lemma establishes that for any fixed constraint T' and D', it is the case that the constraint is satisfied with good probability in the integral solution x^*. Further, the constraint is satisfied if only the jobs in $M_{T', D'}$ (or $L_{T', D'}$) are considered in the summation in the left hand side of the constraint. Due to space, the proof is ommited.

▶ **Lemma 13.** *Fix any $T' = [t_{T'}, \infty]$ and let the vector D' contain the time D'_j that is the latest time t where $x'_{j,t} \geq c$ for all jobs j. With probability at least $1 - \frac{1}{\log^{10} nP}$ it is the case that $\sum_{j \in L_{T', D''}} \sum_{t \in E(T', D', j)} x^*_{j,t} \geq 10V(T', D')$ and $\sum_{j \in M_{T', D'}} \sum_{t \in E(T', D', j)} x^*_{j,t} \geq 10V(T', D')$ when $c \leq \frac{1}{1000 \log \log nP}$.*

Fix any job j^*. Group constraints into two classes for job j^*. The first class are those constraints $T' = [t_{T'}, \infty]$ where $t_{T'} > D'_{j^*}$ and the second class are the remaining constraints. It will be shown separately for both groups of constraints that they are all unsatisfied with small probability.

In the following lemma, the first class of constraints are considered. This lemma heavily relies on Proposition 11.

▶ **Lemma 14.** *Fix any job j^*. With probability at most $1/\log^2 nP$ it is the case that there exists a constraint T', D'' unsatisfied by x^*, $j^* \in \mathcal{J}_{T', D''}$ and $t_{T'} > D'_{j^*}$ when $c \leq \frac{1}{1000 \log \log nP}$.*

Proof. Fix any job j^*. Let D' be set such that D'_j is the latest time t where $x'_{j,t} \geq c$ for all jobs j. The proof will establish that with probability greater $1 - \frac{1}{\log^2 nP}$ it is the case that the constraints for D' and any $T' = [t_{T'}, \infty]$ with $t_{T'} > D'_{j^*}$ are satisfied by x^*. Applying Proposition 10 this implies that x^* satisfies the same allowing for any constraint D'', proving the lemma.

Geometrically group constraints based on the value of $V(T', D')$. Let $\mathcal{C}_k$ contain the set $T' = [t_{T'}, \infty]$ if $j^* \in \mathcal{J}_{T', D'}$, $2^k \leq V(T', D') < 2^{k+1}$ and $t_{T'} > D'_{j^*}$ for any integer $0 \leq k \leq \log nP$.

Fix k and the set $T' \in \mathcal{C}_k$ such that $t_{T'}$ is as late as possible. Let $L_{T', D'}$ be as described in the algorithm definition. We will establish that if $\sum_{j \in L_{T', D'}} \sum_{t \in E(T', D', j)} x^*_{j,t} \geq 10V(T', D')$ then all constraints $V(T'', D')$ for any $T'' \in \mathcal{C}_k$ are satisfied. Once this is established, this will complete the proof as follows. We apply Lemma 13 stating that $\sum_{j \in L_{T', D'}} \sum_{t \in E(T', D', j)} x^*_{j,t} \geq 10V(T', D')$ occurs with probability at least $1 - \frac{1}{\log^{10} nP}$. By union bounding for all $\log nP$ values for k the lemma follows.

Say that $\sum_{j \in L_{T', D'}} \sum_{t \in E(T', D', j)} x^*_{j,t} \geq 10V(T', D')$. Consider any set $T'' \in \mathcal{C}_k$. By definition of the set $L_{T', D'}$ it is the case that $D'_j \leq D'_{j^*}$ for all $j \in L_{T', D'}$. Hence, $D'_j \leq D'_{j^*} \leq t_{T''} \leq t_{T'}$. Thus, Proposition 11 and the geometric grouping of constraints gives that $\sum_{j \in L_{T', D'}} \sum_{t \in E(T'', D', j)} x^*_{j,t} \geq \frac{1}{2} \sum_{j \in L_{T', D'}} \sum_{t \in E(T', D', j)} x^*_{j,t} \geq 5V(T', D') \geq V(T'', D')$. Thus the constraint for T'' and D' is satisfied, proving the lemma. ◀

Similar to the previous lemma, in the following lemma it is shown that all of the second class of constraints are satisfied with good probability. This lemma heavily relies on Proposition 12. This proof is similar to the prior lemma and is omitted.

▶ **Lemma 15.** *Fix any job j^*. With probability at most $1/\log^2 nP$ it is the case that there exists a constraint T', D'' unsatisfied by x^*, $j^* \in \mathcal{J}_{T',D'}$ and $t_{T'} < D'_{j^*}$ when $c \leq \frac{1}{1000 \log\log nP}$.*

For sufficiently large n and P, a union bound and Lemmas 14 and 15 prove Lemma 9.

7 Conclusion

This paper introduced a new set of strong inequalities for scheduling problems on multiple identical machines. Using these inequalities, the paper showed an iterative algorithm that rounds a fractional LP solution to an integral solution which achieves an $O(\log\log nP)$ approximation for most reasonable scheduling minimization problems.

An open question is if there are algorithms with an $O(1)$ approximation ratio for GSP on identical machines. It is also of interest to determine if the inequalities introduced can be extended to other bipartite assignment problems. Can these inequalities be extended to more general environments, such as the related machines or restricted assignment settings? Could a similar analysis be used when jobs arrive over time? These cases introduce new technical hurdles, but $O(1)$ approximation algorithms could be possible by leveraging the given constraints and assuming preemption and migration are allowed.[4] For example, if jobs arrive over time then a generalization of the LP with the strengthened constraints can be derived. However, the rounding becomes challenging because there appears to be no reduction showing only a polynomial number of time sets T' need to be considered in the set of constraints, like was established in this paper. Due to this, it is challenging to show all exponential number of constraints based on sets of times are satisfied by the rounding algorithm.

References

1. Antonios Antoniadis, Ruben Hoeksma, Julie Meißner, José Verschae, and Andreas Wiese. A QPTAS for the General Scheduling Problem with Identical Release Dates. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 31:1–31:14, 2017.
2. Nikhil Bansal and Kirk Pruhs. The Geometry of Scheduling. *SIAM J. Comput.*, 43(5):1684–1698, 2014.
3. Robert D. Carr, Lisa Fleischer, Vitus J. Leung, and Cynthia A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA.*, pages 106–115, 2000.
4. Timothy M. Chan, Elyot Grant, Jochen Könemann, and Malcolm Sharpe. Weighted capacitated, priority, and geometric set cover via improved quasi-uniform sampling. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1576–1585, 2012.
5. T. C. Edwin Cheng, C. T. Ng, J. J. Yuan, and Zhaohui Liu. Single machine scheduling to minimize total weighted tardiness. *European Journal of Operational Research*, 165(2):423–443, 2005.
6. Maurice Cheung, Julián Mestre, David B. Shmoys, and José Verschae. A Primal-Dual Approximation Algorithm for Min-Sum Single-Machine Scheduling Problems. *SIAM J. Discrete Math.*, 31(2):825–838, 2017.

[4] We remind the reader that any algorithm for GSP on identical machines has an unbounded approximation ratio if either preemption or migration are not allowed

7 Maurice Cheung and David B. Shmoys. A Primal-Dual Approximation Algorithm for Min-Sum Single-Machine Scheduling Problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 14th International Workshop, APPROX 2011, and 15th International Workshop, RANDOM 2011, Princeton, NJ, USA, August 17-19, 2011. Proceedings*, pages 135–146, 2011.

8 Julia Chuzhoy, Sudipto Guha, Sanjeev Khanna, and Joseph Naor. Machine Minimization for Scheduling Jobs with Interval Constraints. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 81–90, 2004.

9 Dorit S. Hochbaum and David B. Shmoys. Using Dual Approximation Algorithms for Scheduling Problems: Theoretical and Practical Results. In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 79–89, 1985.

10 Wiebke Höhn and Tobias Jacobs. On the Performance of Smith's Rule in Single-Machine Scheduling with Nonlinear Cost. *ACM Trans. Algorithms*, 11(4):25, 2015.

11 Wiebke Höhn, Julián Mestre, and Andreas Wiese. How Unsplittable-Flow-Covering Helps Scheduling with Job-Dependent Cost Functions. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 625–636, 2014.

12 Sungjin Im and Benjamin Moseley. Fair Scheduling via Iterative Quasi-Uniform Sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2601–2615, 2017.

13 E.L. Lawler. A fully polynomial approximation scheme for the total tardiness problem. *Operations Research Letters*, 1(6):207–208, 1982. doi:10.1016/0167-6377(82)90022-0.

14 Retsef Levi, Andrea Lodi, and Maxim Sviridenko. Approximation Algorithms for the Capacitated Multi-Item Lot-Sizing Problem via Flow-Cover Inequalities. *Math. Oper. Res.*, 33(2):461–474, 2008.

15 David Lo, Liqun Cheng, Rama Govindaraju, Parthasarathy Ranganathan, and Christos Kozyrakis. Heracles: Improving Resource Efficiency at Scale. In *Proceedings of the 42Nd Annual International Symposium on Computer Architecture*, ISCA '15, pages 450–462, Portland, Oregon, 2015. ACM. doi:10.1145/2749469.2749475.

16 Jason Mars, Lingjia Tang, Robert Hundt, Kevin Skadron, and Mary Lou Soffa. Bubble-Up: Increasing Utilization in Modern Warehouse Scale Computers via Sensible Co-locations. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44, pages 248–259, Porto Alegre, Brazil, 2011. ACM. doi:10.1145/2155620.2155650.

17 Paul Marshall, Kate Keahey, and Tim Freeman. Improving Utilization of Infrastructure Clouds. In *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, CCGRID '11, pages 205–214, Washington, DC, USA, 2011. IEEE Computer Society. doi:10.1109/CCGrid.2011.56.

18 Julián Mestre and José Verschae. A 4-approximation for scheduling on a single machine with general cost function. *CoRR*, abs/1403.0298, 2014. arXiv:1403.0298.

19 George L. Nemhauser and Laurence A. Wolsey. *Integer and combinatorial optimization*. Wiley interscience series in discrete mathematics and optimization. Wiley, 1988.

20 Tony J. Van Roy and Laurence A. Wolsey. Valid inequalities for mixed 0-1 programs. *Discrete Applied Mathematics*, 14(2):199–213, 1986.

21 Wayne E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3:59–66, 1956.

22 Kasturi R. Varadarajan. Weighted geometric set cover via quasi-uniform sampling. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 641–648, 2010.

Computing Optimal Epsilon-Nets Is as Easy as Finding an Unhit Set

Nabil H. Mustafa
Université Paris-Est, Laboratoire d'Informatique Gaspard-Monge, ESIEE Paris, France
mustafan@esiee.fr

Abstract

Given a set system $(X, \mathcal{R})$ with VC-dimension d, the celebrated result of Haussler and Welzl (1987) showed that there exists an ϵ-net for $(X, \mathcal{R})$ of size $O\left(\frac{d}{\epsilon} \log \frac{1}{\epsilon}\right)$. Furthermore, the algorithm is simple: just take a uniform random sample from X! However, for many geometric set systems this bound is sub-optimal and since then, there has been much work presenting improved bounds and algorithms tailored to specific geometric set systems.

In this paper, we consider the following natural algorithm to compute an ϵ-net: start with an initial random sample N. Iteratively, as long as N is not an ϵ-net for $\mathcal{R}$, pick *any* unhit set $S \in \mathcal{R}$ (say, given by an Oracle), and add $O(1)$ randomly chosen points from S to N.

We prove that the above algorithm computes, in expectation, ϵ-nets of asymptotically optimal size for all known cases of geometric set systems. Furthermore, it makes $O\left(\frac{1}{\epsilon}\right)$ calls to the Oracle. In particular, this implies that computing optimal-sized ϵ-nets are as easy as computing an unhit set in the given set system.

2012 ACM Subject Classification Theory of computation → Sketching and sampling

Keywords and phrases ϵ-nets, Geometric Set Systems

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.87

Category Track A: Algorithms, Complexity and Games

Funding This research was supported by the grant ANR SAGA (JCJC-14-CE25-0016-01).

Acknowledgements We thank the reviewers for insightful feedback that helped the content and presentation of this work.

1 Introduction

Let X be a set of n base elements, and $\mathcal{R}$ a collection of m sets over X. Given a parameter $\epsilon > 0$, an ϵ-net for $(X, \mathcal{R})$ is a set $N \subseteq X$ such that for all $S \in \mathcal{R}$ with $|S| \geq \epsilon n$, we have $N \cap S \neq \emptyset$.

The notion of ϵ-nets has been heavily studied across several disciplines, including discrete and computational geometry, machine learning, statistics, convex geometry and randomized algorithms. We refer the reader to the books [3, 7, 14, 16, 20] as well as recent surveys [19, 21].

For general set systems $(X, \mathcal{R})$, it is easy to see that there exists an ϵ-net of size $O\left(\frac{1}{\epsilon} \log |\mathcal{R}|\right)$. For more constrained set systems – for example, those arising in geometric configurations – one can show the existence of ϵ-nets of considerably smaller size. This was realized in the 1980s with the seminal work of Clarkson [8] and Haussler-Welzl [11]. In particular, when the VC-dimension of $(X, \mathcal{R})$, denoted by VC-dim$(\mathcal{R})$, is at most d, then there exist ϵ-nets of size $O\left(\frac{d}{\epsilon} \log \frac{1}{\epsilon}\right)$ – of size *independent* of $|X|$ or $|\mathcal{R}|$. Furthermore, to construct a net of this expected size, the algorithm is simple: just take a uniform random sample.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 87; pp. 87:1–87:12

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

Haussler-Welzl Net-Finder Algorithm.

Input: base elements X, a set system $\mathcal{R}$ on X, a parameter $\epsilon > 0$.

N: pick each $x \in X$ independently with probability $\Theta\left(\frac{d}{\epsilon|X|}\log\frac{1}{\epsilon}\right)$.

return N.

It has been observed over the past 30 years that improvements to the Clarkson and Haussler-Welzl bounds are possible for a variety of geometric set systems – e.g., $O(\frac{1}{\epsilon})$-sized nets exist for subsets induced by disks in the plane, half-spaces in $\mathbb{R}^2$ and $\mathbb{R}^3$, subsets induced by pseudo-disks, dual set systems of linear union complexity and so on. More recently, the work of Varadarajan [23], Aronov et al. [4] and Chan et al. [6] has settled the question on sizes of ϵ-nets for many basic geometric set systems precisely in terms of their so-called shallow-cell complexity – in particular, there exist ϵ-nets of size $O\left(\frac{1}{\epsilon}\log\varphi_{\mathcal{R}}\left(O\left(\frac{1}{\epsilon}\right), O\left(1\right)\right)\right)$. A set system $(X, \mathcal{R})$ has shallow-cell complexity $\varphi_{\mathcal{R}}(\cdot,\cdot)$ if for any $Y \subseteq X$, the number of subsets in $\mathcal{R}|_Y$ of size at most l is $|Y| \cdot \varphi_{\mathcal{R}}\left(|Y|, l\right)$ (here $\mathcal{R}|_Y = \{S \cap Y : S \in \mathcal{R}\}$ is the projection of the set system $\mathcal{R}$ on Y). Bounds for $\varphi_{\mathcal{R}}\left(\cdot,\cdot\right)$ has been well-studied and by now we know asymptotically optimal bounds for the basic geometric set systems (see [19]).

However, all the above algorithms for constructing ϵ-nets either have efficient implementations but then only work for very specific set systems (e.g., near-linear time algorithm for disks in $\mathbb{R}^2$ [5], half-spaces in $\mathbb{R}^3$ [15]), or are inefficient if they work for general set systems. Consider these algorithms from recent work:

1. Chan et al. [6] construction gives optimal-sized nets as a function of the shallow-cell complexity of the set system; however the algorithm is inefficient. It has to enumerate over each set of $\mathcal{R}$ – there can be $\Omega(n^d)$ such sets for some constant d – to compute a representative of each set of $\mathcal{R}$. Furthermore, it needs to have access to all the sets of $\mathcal{R}$ at the beginning to be able to compute these representatives.
2. Aronov et al. [4] and Varadarajan [23] algorithms can be implemented to work efficiently, but they work for special cases of set systems (so-called dual set systems induced by geometric objects in the plane) and further also need certain spatial decompositions (for the complement of the union) which are specific to the types of geometric objects.
3. Mustafa et al. [18] also give general bounds in terms of the shallow-cell complexity of a set system. However, the algorithm needs to first select a special maximal subset of $\mathcal{R}$ called a packing. This packing can have large size, and furthermore, computing this packing is inefficient, taking $\Omega(n^2)$ time.

2 Our Result

Our main insight is that the cause of inefficiency – the careful construction of sets needed for the "alterations" – can be avoided altogether. By extending the ideas present in the work in 1. and 3. above, we present a simple algorithm that

a) computes an ϵ-net of expected size matching the current-best bounds for known geometric set systems, and

b) avoids any pre-computation, hierarchical subdivisions, representation-computation, or partitioning.

Here is our algorithm – as it turns out, a slight addition of the **Haussler-Welzl Net-Finder Algorithm**.

General Net-Finder Algorithm.

Input: base elements X, a set system $\mathcal{R}$ on X, a parameter $\epsilon > 0$.

N: pick each $x \in X$ independently with probability p to be fixed later (Section 3).

while *there exists a set* $S \in \mathcal{R}$, $|S| \geq \epsilon|X|$, *not hit by* N **do**
    update N by adding $O(1)$ uniformly chosen elements of S to N.

return N

Oracle. In order to separate the set system-specific implementation details from the net algorithm, we will assume the existence of an Oracle that can return an unhit set in our set system with respect to the current candidate net N. We refer the reader to Chazelle [7] for details on oracle-based bounds for sampling in geometric set systems. For geometric set systems, the existence of efficient implementation of such oracles follow from the extensive work on range searching, reporting and emptiness data-structures (see [1]). For example, for the case where X is a set of points in $\mathbb{R}^2$ and the sets are induced by disks, the oracle can be implemented to run in overall time $O(n \cdot \text{polylog}(n))$ using standard techniques (e.g., see [5]).

We refer the reader to Agarwal-Pan [2] for details on data-structures for several other geometric set systems.

Our main theorem:

▶ **Theorem 1.** ***General Net-Finder Algorithm*** *computes an ϵ-net of expected optimal size for ϵ-nets for the following set systems:*

1. $O\left(\frac{1}{\epsilon}\log\varphi_{\mathcal{R}}\left(O\left(\frac{d}{\epsilon}\right), O(d)\right) + \frac{d}{\epsilon}\right)$*: abstract set systems as a function of their shallow-cell complexity $\varphi_{\mathcal{R}}(\cdot,\cdot)$ (we will assume that $\varphi_{\mathcal{R}}(\cdot,\cdot)$ is non-decreasing in both arguments) and VC-dimension d,*
2. $O\left(\frac{1}{\epsilon}\right)$*: half-spaces in $\mathbb{R}^2$, half-spaces in $\mathbb{R}^3$, pseudo-disks and disks in $\mathbb{R}^2$, dual systems of linear union complexity,*
3. $O\left(\frac{1}{\epsilon}\log\log\frac{1}{\epsilon}\right)$*: axis-parallel rectangles in $\mathbb{R}^2$,*
4. $O\left(\frac{\log(\epsilon\cdot\kappa_{\mathcal{R}}(1/\epsilon))}{\epsilon}\right)$*: dual set systems as a function of their union complexity $\kappa_{\mathcal{R}}(\cdot)$,*
5. $O\left(\frac{d}{\epsilon}\log\frac{1}{\epsilon}\right)$*: set systems of VC-dimension at most d, half-spaces in $\mathbb{R}^d$.*

See [19, Table 47.4.1] for the complete list of known bounds, all of which are produced by our algorithm.

Furthermore, it makes expected $O(\frac{1}{\epsilon})$ calls to the Oracle.

▶ Remark 1. For example, all the bounds presented in the work of Varadarajan [23], Clarkson and Varadarajan [10], Pyrga-Ray [22], Chan et al. [6], Aronov et al. [4], Mustafa et al. [18] are achieved by our **General Net-Finder Algorithm**.

▶ Remark 2. We note here that the unhit set S returned by the Oracle at each step need not be random – it can be any unhit set. The expectation is over the choice of the initial random sample as well as the $O(1)$ random points picked from S. The specific choice of S is irrelevant.

▶ Remark 3. In particular, Theorem 1 shows that computing an ϵ-net of optimal size is as easy/hard – within a multiplicative factor of $\frac{1}{\epsilon}$ – as computing one set unhit by $N \subseteq X$ in a set system $(X, \mathcal{R})$.

▶ Remark 4. This work is an example of the phenomenon where the "complexity" is moved from the algorithm to its analysis. Thus while our analysis uses specific combinatorial and geometric structures, the algorithm itself becomes very simple and oblivious to these structures (e.g., see [24]).

Lastly, we observe that for the case of set systems with linear-sized ϵ-nets, it is not even necessary to take an initial random sample. The algorithm simplifies even further to:

Special Net-Finder Algorithm.

Input: base elements X, a set system $\mathcal{R}$ on X, a parameter $\epsilon > 0$.

$N = \emptyset$.
while *there exists a set* $S \in \mathcal{R}$, $|S| \geq \epsilon|X|$, *not hit by* N **do**
    update N by adding $O(1)$ uniformly chosen elements of S to N.

return N.

Our second theorem is the following.

▶ **Theorem 2.** *If the shallow-cell complexity of* $\mathcal{R}$ *satisfies* $\varphi_{\mathcal{R}}(n,k) = O\left(k^c\right)$, *then* ***Special Net-Finder Algorithm*** *computes an* ϵ*-net of expected size* $O\left(\frac{1}{\epsilon}\right)$. *In particular, it computes a* $O\left(\frac{1}{\epsilon}\right)$*-sized net for the set systems induced by half-spaces in* $\mathbb{R}^2$, *half-spaces in* $\mathbb{R}^3$, *pseudo-disks and disks in* $\mathbb{R}^2$, *and dual systems of linear union complexity.*

Furthermore it makes expected $O(\frac{1}{\epsilon})$ *calls to the Oracle.*

Organization. In Section 3 we prove some key structural lemmas about the random process common to both the above algorithms. Then in Section 4 we give the proof of Theorem 1, and in Section 5 the proof of Theorem 2.

3 Key Lemmas

We first re-state our main method, **General Net-Finder Algorithm**, more precisely by filling in the exact constant values and probabilities that will be then used in the proofs.

Let $(X, \mathcal{R})$ be the given set system with VC-dim$(\mathcal{R}) \leq d$, and shallow-cell complexity $\varphi_{\mathcal{R}}(\cdot,\cdot)$.

General Net-Finder Algorithm.

Input: $(X, \mathcal{R})$ with VC-dim$(\mathcal{R}) \leq d$ and shallow-cell complexity $\varphi_{\mathcal{R}}(\cdot,\cdot)$, parameter $\epsilon > 0$.

β, γ, c_a are absolute constants (explicitly fixed later).

N_0: pick $x \in X$ i.i.d. with prob. $c_a \cdot \left(\frac{1}{\left(\frac{3}{4} - \frac{\beta}{2}\right)\epsilon|X|} \log\left(d^3 \varphi_{\mathcal{R}}\left(\frac{4d}{\beta\epsilon}, \frac{24d}{\beta}\right)^2 \right) + \frac{d}{\left(\frac{3}{4} - \frac{\beta}{2}\right)\epsilon|X|} \log \frac{1}{\left(\frac{3}{4} - \frac{\beta}{2}\right)} \right)$

$N = N_0$.
while *there exists a set* $S \in \mathcal{R}$, $|S| \geq \epsilon|X|$, *not hit by* N **do**
    N_S: pick each $x \in S$ independently with probability $c_a \cdot \left(\frac{1}{\gamma|S|} \log 2 \; + \; \frac{d}{\gamma|S|} \log \frac{1}{\gamma} \right)$.
    $N = N \cup N_S$.

return N.

For the **Special Net-Finder Algorithm**, simply omit the initial random sample, and start with $N = N_0 = \emptyset$.

The proof of our main result builds on the technique in [18]. There the packing lemma was used to construct a maximal packing. The new insight in this paper is that through the use of *two-level* packings, it is not necessary to even know or construct maximal packings (the computational bottleneck earlier). The algorithm is blind to the specific packing; however the analysis amortizes the cost of adding new points to a second-level packing constructed from the sets of the first packing. This is the key new idea, enabling us to bound the total number of points added after the initial random sample.

For the proof of our two main theorems, we will need the following results.

▶ **Theorem A** (Epsilon-net Theorem [11, 12]). *Let $(X, \mathcal{R})$ be a set system, $d \in \mathbb{N}^+$ a positive integer such that* VC-dim$(\mathcal{R}) \leq d$*, and $\epsilon \in [0, 1]$ a given real parameter. Then there exists an absolute constant $c_a > 0$ such that a random sample N constructed by picking each point of X independently with probability*

$$c_a \cdot \left(\frac{1}{\epsilon |X|} \log \frac{1}{\gamma} \; + \; \frac{d}{\epsilon |X|} \log \frac{1}{\epsilon} \right).$$

is an ϵ-net for $\mathcal{R}$ with probability at least $1 - \gamma$.

Given $(X, \mathcal{R})$, a (k, δ)-packing of $\mathcal{R}$ is a subset $\mathcal{P} \subseteq \mathcal{R}$ such that $i)$ for all $S \in \mathcal{P}$ we have $|S| \leq k$, and $ii)$ for all $S, S' \in \mathcal{P}$ we have $|\Delta(S, S')| \geq \delta$. Here $\Delta(A, B) = (A \setminus B) \cup (B \setminus A)$ is the symmetric difference of A and B.

▶ **Theorem B** (Shallow Packing Lemma [17]). *Let $(X, \mathcal{P})$ be a (k, δ)-packing on n elements, for integers $k, \delta > 0$. If* VC-dim$(\mathcal{P}) \leq d$ *and $\mathcal{P}$ has shallow-cell complexity $\varphi(\cdot, \cdot)$, then $|\mathcal{P}| \leq \frac{24dn}{\delta} \cdot \varphi\big(\frac{4dn}{\delta}, \frac{12dk}{\delta}\big)$.*

In the proof below, we assume that each set S considered by the algorithm has size $[\epsilon n, 2\epsilon n]$. Then we will show that, in expectation, $O(\frac{1}{\epsilon})$ additional points are added after the initial random sample N_0. The general case – where the sets S considered by the algorithm can have any size greater than ϵn – follows directly: we group the sets considered by the algorithm by their sizes – all sets of size $[2^i \epsilon n, 2^{i+1} \epsilon n]$ go into the same group i. So the algorithm can be seen as constructing different nets, a ϵ'-net where $\epsilon' = 2^i \epsilon$, for group i, simultaneously. The proof below shows that for each group i, the expected number of elements added is $O\left(\frac{1}{\epsilon'}\right) = O\left(\frac{1}{2^i \epsilon}\right)$. Then summing up gives a geometric series, with the overall bound of $O\left(\frac{1}{\epsilon}\right)$ points added over all groups. The initial random sample N_0 can be thought of as a different sample for each group, with the total size over all groups again forming a geometric series which sums up to the stated bound.

Let β, γ be two positive reals whose value will be fixed later, with the property that $\gamma \leq \frac{1}{4}$ and $0 \leq \beta + \gamma \leq 1$.

Fix any *maximal* $(2\epsilon n, \beta \epsilon n)$-packing $\mathcal{P}$ of $\mathcal{R}$ consisting of sets of size at least ϵn; say the packing consists of the sets

$$\mathcal{P} = \left\{P^1, \ldots, P^m\right\}, \text{ where } m \leq \frac{24dn}{\beta \epsilon n} \varphi_{\mathcal{R}} \left(\frac{4dn}{\beta \epsilon n}, \frac{24d\epsilon n}{\beta \epsilon n} \right)$$
$$= O\left(\frac{d}{\beta \epsilon} \cdot \varphi_{\mathcal{R}} \left(\frac{4d}{\beta \epsilon}, \frac{24d}{\beta} \right) \right) \text{ (by Theorem B).}$$

Say the **Net-Finder Algorithm** (both general and special) continues for t steps, and adds additional points of X to N for each of the sets $S_1, \ldots, S_t$, namely the points $N_{S_1}, \ldots, N_{S_t}$.

As $\mathcal{P}$ is a maximal $(2\epsilon n, \beta\epsilon n)$-packing of $\mathcal{R}$ and $\epsilon n \leq |S_i| \leq 2\epsilon n$ for each $i = 1, \ldots, t$, it must be that for each S_i, there exists an index $j \in [m]$ with $|\Delta(S_i, P^j)| < \beta\epsilon n$ (note that it is possible that $P^j = S_i$). Assign S_i to the set P^j (pick an arbitrary one if there are several such possibilities).

For each $j \in [m]$, let n_j be the number of sets of $\{S_1, \ldots, S_t\}$ assigned to $P^j \in \mathcal{P}$, and denote them by

$$\mathcal{S}^j = \langle S^j_1, \ldots, S^j_{n_j} \rangle, \quad \text{listed } \textit{in the order} \text{ considered by the } \textbf{Net-Finder Algorithm}.$$

Note that $\sum_{j=1}^{m} n_j = t$, and furthermore,

$$\text{for every } j \in [m], \;\; i \in [n_j], \quad \text{we have} \quad |\Delta(S^j_i, P^j)| < \beta\epsilon n. \tag{1}$$

▷ Claim 3. For each $j \in [m]$ and $i \in [n_j]$, we have

$$|P^j \cap S^j_i| > \frac{|P^j| + |S^j_i| - \beta\epsilon n}{2}.$$

Proof. For each $i \in [n_j]$, we have

$$|P^j| + |S^j_i| = |P^j \setminus S^j_i| + |S^j_i \setminus P^j| + 2|P^j \cap S^j_i| < \beta\epsilon n + 2|P^j \cap S^j_i|, \tag{2}$$

where the last inequality follows from (1). Re-arranging the terms above gives the required statement. ◁

For each $j \in [m]$, define

$$\mathcal{S}'^j = \left\{ S \in \mathcal{S}^j : N_S \text{ turns out to be a } \gamma\text{-net for the set system } (S, \mathcal{R}|_S) \right\}.$$

▷ Claim 4. For any $j \in [m]$,

$$|\mathcal{S}'^j| = \begin{cases} O\left(\frac{d}{\frac{3}{2}-\beta-\gamma} \cdot \varphi_{\mathcal{R}}\left(\frac{4d}{\frac{3}{2}-\beta-\gamma}, \frac{12d}{\frac{3}{2}-\beta-\gamma}\right)\right) & \text{if } \beta + \gamma \geq \frac{1}{2}, \\ O(1) & \text{otherwise.} \end{cases}$$

Proof. Let $n'_j = |\mathcal{S}'^j|$. By re-labeling the sets of $\mathcal{S}^j$, we can assume that $\mathcal{S}'^j = \langle S^j_1, \ldots, S^j_{n'_j} \rangle$, again listed here in the order that they were considered by the **Net-Finder Algorithm**.

Consider the set system $\mathcal{T}'^j = \langle T^j_1, \ldots, T^j_{n'_j} \rangle$ over the base set P^j, where $T^j_i = S^j_i \cap P^j$. Consider two distinct indices $k, l \in [n'_j]$ with $k < l$. The set $N_{S^j_k}$ was added to N in the algorithm *before* the set S^j_l was considered. In particular, as $N_{S^j_k}$ is a γ-net for $\left(S^j_k, \mathcal{R}|_{S^j_k}\right)$ (by the definition of $\mathcal{S}'^j$), it must be that the set S^j_l was not hit by the γ-net for $\left(S^j_k, \mathcal{R}|_{S^j_k}\right)$ and so $|S^j_k \cap S^j_l| < \gamma \cdot |S^j_k|$. In particular, this implies that

$$|T^j_k \cap T^j_l| = |S^j_k \cap S^j_l \cap P^j| \leq |S^j_k \cap S^j_l| < \gamma \cdot |S^j_k|. \tag{3}$$

Thus we have

$$\begin{aligned}
|\Delta(T_k^j, T_l^j)| &= |T_k^j| + |T_l^j| - 2|T_k^j \cap T_l^j| \qquad (4)\\
&= |S_k^j \cap P^j| + |S_l^j \cap P^j| - 2|T_k^j \cap T_l^j|\\
&> \frac{|P^j| + |S_k^j| - \beta\epsilon n}{2} + \frac{|P^j| + |S_l^j| - \beta\epsilon n}{2} - 2|T_k^j \cap T_l^j| \quad \text{(by Claim 3)}\\
&> \frac{|P^j| + |S_k^j| - \beta\epsilon n}{2} + \frac{|P^j| + |S_l^j| - \beta\epsilon n}{2} - 2\cdot\gamma\cdot|S_k^j| \qquad \text{(by Inequality (3))}\\
&= |P^j| - \beta\epsilon n + \frac{|S_l^j|}{2} + \left(\frac{1}{2} - 2\gamma\right)|S_k^j|\\
&\geq |P^j| - \beta|P^j| + \frac{|P^j|/2}{2} + \left(\frac{1}{2} - 2\gamma\right)\frac{|P^j|}{2} \quad (\text{as } \epsilon n \leq |P^j|, |S_k^j|, |S_l^j| \leq 2\epsilon n,\ \gamma \leq \tfrac{1}{4})\\
&= \left(\frac{3}{2} - \beta - \gamma\right)\cdot|P^j|. \qquad (5)
\end{aligned}$$

There are two cases to consider here:

- $\beta + \gamma < \frac{1}{2}$. In this case, Inequality (5) implies that $|\Delta(T_k^j, T_l^j)| > |P^j|$, which cannot happen as both T_k^j and T_l^j are subsets of P^j. Thus $\mathcal{S}'^j$ must consist of at most one set, and we're done.
- $\beta + \gamma \geq \frac{1}{2}$. In this case, the sets of $\mathcal{T}'^j$ form a $\left(|P^j|, \left(\frac{3}{2} - \beta - \gamma\right)\cdot|P^j|\right)$-packing over the elements of P^j. Thus by Theorem B, we have

$$|\mathcal{S}'^j| = |\mathcal{T}'^j| = O\left(\frac{d}{\frac{3}{2} - \beta - \gamma}\cdot\varphi_{\mathcal{R}}\left(\frac{4d}{\frac{3}{2} - \beta - \gamma}, \frac{12d}{\frac{3}{2} - \beta - \gamma}\right)\right).$$ ◁

▶ **Lemma 5.**

$$\mathbf{E}\left[|\mathcal{S}^j|\right] = \begin{cases} O\left(\frac{d}{\frac{3}{2}-\beta-\gamma}\cdot\varphi_{\mathcal{R}}\left(\frac{4d}{\frac{3}{2}-\beta-\gamma}, \frac{12d}{\frac{3}{2}-\beta-\gamma}\right)\right) & \text{if } \beta+\gamma \geq \frac{1}{2},\\ O(1) & \text{otherwise.}\end{cases}$$

Further, the above bound holds for any choice of N_0.

Proof. We prove this bound for any choice of N_0, relying, in the following analysis, only on the sets N_S that were added iteratively. Note that there may be complicated dependencies among the sets of $\mathcal{S}^j$. For example, a set S_l^j may only exist in $\mathcal{S}^j$ because of the choice of the random sample for some earlier set S_k^j, $k < l$. However, for a fixed $S \in \mathcal{R}$, the probability of the random sample N_S being a γ-net for the set system $(S, \mathcal{R}|_S)$ is independent of earlier iterations, and occurs with probability at least $\frac{1}{2}$ by Theorem A. Recalling that $|\mathcal{S}'^j|$ is the number of sets of $\mathcal{S}^j$ for which the random sample succeeds to be a γ-net, we have

$$\mathbf{E}\left[|\mathcal{S}'^j|\right] = \sum_{S\in\mathcal{S}^j}\Pr\left[N_S \text{ is a } \gamma\text{-net for } (S, \mathcal{R}|_S)\right] \geq \frac{|\mathcal{S}^j|}{2}.$$

On the other hand, Claim 4 upper-bounds $|\mathcal{S}'^j|$ and thus $\mathbf{E}[|\mathcal{S}'^j|]$. Putting them together implies the lemma. ◀

4 Proof of Theorem 1

We first give the key theorem from which we will then derive all the bounds promised in Theorem 1. We continue to use the notations and definitions defined earlier.

▶ **Theorem 6.** *Let $(X, \mathcal{R})$ be a set system with shallow-cell complexity $\varphi_{\mathcal{R}}(\cdot,\cdot)$ and VC-dimension at most d. Then the* ***General Net-Finder Algorithm*** *returns an ϵ-net of expected size $O\left(\frac{1}{\epsilon}\log\varphi_{\mathcal{R}}\left(O\left(\frac{d}{\epsilon}\right), O(d)\right) + \frac{d}{\epsilon}\right)$. Furthermore, it makes expected $O\left(\frac{1}{\epsilon}\right)$ calls to the Oracle.*

Proof. Clearly the algorithm only stops when N is an ϵ-net. Thus it remains to bound the expected size and the expected running time.

Consider an index $j \in [m]$. By Claim 3, we have that for any $i \in [n_j]$,

$$|P^j \cap S_i^j| > \frac{|P^j| + |S_i^j| - \beta\epsilon n}{2} \geq \frac{|P^j| + |P^j|/2 - \beta|P^j|}{2} = \left(\frac{3}{4} - \frac{\beta}{2}\right)\cdot|P^j|,$$

recalling that $\beta \leq 1$. Thus if N_0 is a $\left(\frac{3}{4}-\frac{\beta}{2}\right)$-net for $\left(P^j, \mathcal{R}|_{P^j}\right)$, then any $S \in \mathcal{S}^j$ would be hit by N_0 and so it must be that $\mathcal{S}^j = \emptyset$. By Theorem A, for a fixed index j, N_0 fails to be a $\left(\frac{3}{4}-\frac{\beta}{2}\right)$-net for $\left(P^j, \mathcal{R}|_{P^j}\right)$ with probability $O\left(\frac{1}{d^3\varphi_{\mathcal{R}}\left(\frac{4d}{\beta\epsilon}, \frac{24d}{\beta}\right)^2}\right)$.

At each iteration, for a set $S \in \mathcal{R}$ not hit by N, we add

$$\mathbf{E}\left[|N_S|\right] = |S|\cdot c_a \cdot \left(\frac{1}{\gamma|S|}\log 2 \;+\; \frac{d}{\gamma|S|}\log\frac{1}{\gamma}\right) = O\left(\frac{d}{\gamma}\log\frac{1}{\gamma}\right)$$

new points to N. Thus the points added to N over all iterations are

$$\begin{aligned}
&\mathbf{E}\left[\sum_{i=1}^{t}|N_{S_i}|\right] = O\left(\frac{d}{\gamma}\log\frac{1}{\gamma}\right)\cdot\mathbf{E}[t] = O\left(\frac{d}{\gamma}\log\frac{1}{\gamma}\right)\cdot\mathbf{E}\left[\sum_{j=1}^{m}|\mathcal{S}^j|\right]\\
&= O\left(\frac{d}{\gamma}\log\frac{1}{\gamma}\right)\cdot\sum_{j=1}^{m}\Pr\left[N_0 \text{ is not a } \left(\frac{3}{4}-\frac{\beta}{2}\right)\text{-net for } (P^j,\mathcal{R}|_{P^j})\right]\cdot\\
&\qquad\qquad\mathbf{E}\left[|\mathcal{S}^j|\,\middle|\,N_0 \text{ is not a } \left(\frac{3}{4}-\frac{\beta}{2}\right)\text{-net for } (P^j,\mathcal{R}|_{P^j})\right]\\
&= O\left(\frac{d}{\gamma}\log\frac{1}{\gamma}\right)\cdot m\cdot O\left(\frac{1}{d^3\varphi_{\mathcal{R}}\left(\frac{4d}{\beta\epsilon},\frac{12d}{\beta/2}\right)^2}\right)\cdot O\left(\frac{d}{\frac{3}{2}-\beta-\gamma}\cdot\varphi_{\mathcal{R}}\left(\frac{4d}{\frac{3}{2}-\beta-\gamma},\frac{12d}{\frac{3}{2}-\beta-\gamma}\right)\right).
\end{aligned}$$

As $\frac{3}{2}-\beta-\gamma \geq \frac{1}{2} \geq \max\left\{\beta\epsilon, \frac{\beta}{2}\right\}$ for $\epsilon \leq 0.5$ and by the assumption that $\varphi_{\mathcal{R}}(\cdot,\cdot)$ is non-decreasing in the first and second arguments,

$$\begin{aligned}
&\leq O\left(\frac{1}{\gamma\left(\frac{3}{2}-\beta-\gamma\right)}\log\frac{1}{\gamma}\right)\cdot m\cdot O\left(\frac{1}{d\varphi_{\mathcal{R}}\left(\frac{4d}{\beta\epsilon},\frac{24d}{\beta}\right)}\right)\\
&= O\left(\frac{1}{\gamma\left(\frac{3}{2}-\beta-\gamma\right)}\log\frac{1}{\gamma}\right)\cdot O\left(\frac{d}{\beta\epsilon}\cdot\varphi_{\mathcal{R}}\left(\frac{4d}{\beta\epsilon},\frac{24d}{\beta}\right)\right)\cdot O\left(\frac{1}{d\varphi_{\mathcal{R}}\left(\frac{4d}{\beta\epsilon},\frac{24d}{\beta}\right)}\right)\\
&= O\left(\frac{1}{\gamma\left(\frac{3}{2}-\beta-\gamma\right)}\log\frac{1}{\gamma}\right)\cdot O\left(\frac{1}{\beta\epsilon}\right).
\end{aligned}$$

For the initial set N_0, we have

$$\mathbf{E}\left[|N_0|\right] = O\left(\frac{1}{\left(\frac{3}{4}-\frac{\beta}{2}\right)\epsilon}\log\left(d\varphi_{\mathcal{R}}\left(\frac{4d}{\beta\epsilon},\frac{24d}{\beta}\right)\right) + \frac{d}{\left(\frac{3}{4}-\frac{\beta}{2}\right)\epsilon}\log\frac{1}{\left(\frac{3}{4}-\frac{\beta}{2}\right)}\right)$$

Putting everything together, we get

$$\begin{aligned}\mathbf{E}\left[|N|\right] &= \mathbf{E}\left[|N_0|\right] + \mathbf{E}\left[\sum_{i=1}^{t} |N_{S_t}|\right] \\ &= O\left(\frac{1}{\left(\frac{3}{4}-\frac{\beta}{2}\right)\epsilon}\log\left(d\varphi_{\mathcal{R}}\left(\frac{4d}{\beta\epsilon},\frac{24d}{\beta}\right)\right) + \frac{d}{\left(\frac{3}{4}-\frac{\beta}{2}\right)\epsilon}\log\frac{1}{\left(\frac{3}{4}-\frac{\beta}{2}\right)}\right) \\ &\quad + O\left(\frac{1}{\gamma\left(\frac{3}{2}-\beta-\gamma\right)}\log\frac{1}{\gamma}\right)\cdot O\left(\frac{1}{\beta\epsilon}\right).\end{aligned}$$

We can set γ to be any small-enough constant, say $\gamma = \frac{1}{100}$, and set $\beta = \frac{3}{4}$. Thus we get

$$\mathbf{E}\left[|N|\right] = O\left(\frac{1}{\epsilon}\log\left(d\varphi_{\mathcal{R}}\left(\frac{4d}{\beta\epsilon},\frac{24d}{\beta}\right)\right) + \frac{d}{\epsilon}\right) = O\left(\frac{1}{\epsilon}\log\varphi_{\mathcal{R}}\left(O\left(\frac{d}{\epsilon}\right), O\left(d\right)\right) + \frac{d}{\epsilon}\right). \blacktriangleleft$$

▶ **Theorem 1.** ***General Net-Finder Algorithm*** *computes an ϵ-net of expected optimal size for ϵ-nets for the following set systems:*

1. $O\left(\frac{1}{\epsilon}\log\varphi_{\mathcal{R}}\left(O\left(\frac{d}{\epsilon}\right), O(d)\right) + \frac{d}{\epsilon}\right)$*: abstract set systems as a function of their shallow-cell complexity $\varphi_{\mathcal{R}}(\cdot,\cdot)$ (we will assume that $\varphi_{\mathcal{R}}(\cdot,\cdot)$ is non-decreasing in both arguments) and VC-dimension d,*
2. $O\left(\frac{1}{\epsilon}\right)$*: half-spaces in $\mathbb{R}^2$, half-spaces in $\mathbb{R}^3$, pseudo-disks and disks in $\mathbb{R}^2$, dual systems of linear union complexity,*
3. $O\left(\frac{1}{\epsilon}\log\log\frac{1}{\epsilon}\right)$*: axis-parallel rectangles in $\mathbb{R}^2$,*
4. $O\left(\frac{\log(\epsilon\cdot\kappa_{\mathcal{R}}(1/\epsilon))}{\epsilon}\right)$*: dual set systems as a function of their union complexity $\kappa_{\mathcal{R}}(\cdot)$,*
5. $O\left(\frac{d}{\epsilon}\log\frac{1}{\epsilon}\right)$*: set systems of VC-dimension at most d, half-spaces in $\mathbb{R}^d$.*

See [19, Table 47.4.1] for the complete list of known bounds, all of which are produced by our algorithm.

Furthermore, it makes expected $O(\frac{1}{\epsilon})$ calls to the Oracle.

Proof. All except one required bound follows directly from Theorem 6; we refer the reader to the survey [19]. Briefly, for the case of halfspaces in $\mathbb{R}^2$, $\mathbb{R}^3$, disks and pseudo-disks in $\mathbb{R}^2$, we have $\varphi(n,k) = O(k^c)$, for a constant c (by the bound on $(\leq k)$-sets [9]) and so the algorithm returns ϵ-nets of size $O(\frac{1}{\epsilon})$. For the case of half-spaces in $\mathbb{R}^d$ and balls in $\mathbb{R}^{d-1}$, $d \geq 4$, we have $\varphi(n,k) = n^{\lfloor d/2\rfloor - 1}k^{\lceil d/2\rceil}$ and so the algorithm returns ϵ-nets of size $O\left(\frac{d}{\epsilon}\log\frac{1}{\epsilon}\right)$. Similarly the bounds follow for the dual set systems as a function of their union complexity.

The exception is the bound of $O\left(\frac{1}{\epsilon}\log\log\frac{1}{\epsilon}\right)$ for the primal system induced by axis-parallel rectangles in the plane. Here we use a result of Aronov et al. [4] which shows that given a set X of n points in the plane, and $\mathcal{R}$ the set system induced on X by all axis-parallel rectangles, there exists another set system $\mathcal{R}'$ on X with the following property:

1) For each set $R \in \mathcal{R}$ of size at least ϵn induced by an axis-parallel rectangle in the plane, there exists a set $f(R) \in \mathcal{R}'$ also induced by an axis-parallel rectangle in the plane, with $f(R) \subseteq R$ and further $|f(R)| \geq \frac{|R|}{2}$.

2) The shallow-cell complexity of $\mathcal{R}'$ is small – $\varphi_{\mathcal{R}'}(n,k) = O\left(\log n \cdot k^3\right)$.

Now **General Net-Finder Algorithm** takes a $O(1)$-sized uniform random sample from a currently unhit set of $\mathcal{R}$, say R. But then by property 1) above, it takes a $O(1)/2$-sized uniform random sample from $f(R)$. From property 2) above, we have $\varphi_{\mathcal{R}'}(n,k) = O\left(\log n \cdot k^3\right)$ and so Theorem 6 implies a bound of $O\left(\frac{1}{\epsilon}\log\log\frac{1}{\epsilon}\right)$. ◀

5 Proof of Theorem 2

All the known bounds of $O\left(\frac{1}{\epsilon}\right)$ for set systems follow because for such a system $\mathcal{R}$, we have $\varphi_{\mathcal{R}}(n,k) = k^c$ for some constant c. Thus Theorem 2 can be deduced from the following theorem.

▶ **Theorem 7.** *Let $(X,\mathcal{R})$ be a set system with shallow-cell complexity $\varphi(n,k) = O(k^c)$ for some absolute constant c. Then the **Special Net-Finder Algorithm** returns an ϵ-net of expected size $O\left(\frac{1}{\epsilon}\right)$.*

Proof. Clearly the algorithm only stops when N is an ϵ-net. Thus it remains to bound the expected size and the expected running time.

At each iteration, for a set $S \in \mathcal{R}$ not hit by N, we add

$$\mathbf{E}[|N_S|] = |S| \cdot c_a \cdot \left(\frac{1}{\gamma|S|}\log 2 \;+\; \frac{d}{\gamma|S|}\log\frac{1}{\gamma}\right) = O\left(\frac{d}{\gamma}\log\frac{1}{\gamma}\right)$$

new points to N. Thus the points added to N over all iterations are

$$\begin{aligned}
&\mathbf{E}\left[\sum_{i=1}^{t}|N_{S_i}|\right] = O\left(\frac{d}{\gamma}\log\frac{1}{\gamma}\right)\cdot\mathbf{E}[t] = O\left(\frac{d}{\gamma}\log\frac{1}{\gamma}\right)\cdot\mathbf{E}\left[\sum_{j=1}^{m}|\mathcal{S}^j|\right]\\
&= O\left(\frac{d}{\gamma}\log\frac{1}{\gamma}\right)\cdot m\cdot O\left(\frac{d}{\frac{3}{2}-\beta-\gamma}\cdot\varphi_{\mathcal{R}}\left(\frac{4d}{\frac{3}{2}-\beta-\gamma},\frac{12d}{\frac{3}{2}-\beta-\gamma}\right)\right)\\
&= O\left(\frac{d}{\gamma}\log\frac{1}{\gamma}\right)\cdot O\left(\frac{d}{\beta\epsilon}\cdot\varphi_{\mathcal{R}}\left(\frac{4d}{\beta\epsilon},\frac{24d}{\beta}\right)\right)\cdot O\left(\frac{d}{\frac{3}{2}-\beta-\gamma}\cdot\varphi_{\mathcal{R}}\left(\frac{4d}{\frac{3}{2}-\beta-\gamma},\frac{12d}{\frac{3}{2}-\beta-\gamma}\right)\right)
\end{aligned}$$

Using $\gamma = \frac{1}{100}$, $\beta = \frac{3}{4}$, c is an absolute constant, and $d = O(c)$ since $\varphi_{\mathcal{R}}(n,k) = O(k^c)$,

$$= O(d)\cdot O\left(\frac{d}{\epsilon}\cdot d^c\right)\cdot O(d\cdot d^c) = O\left(\frac{1}{\epsilon}\right). \qquad \blacktriangleleft$$

▶ **Theorem 2.** *If the shallow-cell complexity of $\mathcal{R}$ satisfies $\varphi_{\mathcal{R}}(n,k) = O(k^c)$, then **Special Net-Finder Algorithm** computes an ϵ-net of expected size $O\left(\frac{1}{\epsilon}\right)$. In particular, it computes a $O\left(\frac{1}{\epsilon}\right)$-sized net for the set systems induced by half-spaces in $\mathbb{R}^2$, half-spaces in $\mathbb{R}^3$, pseudo-disks and disks in $\mathbb{R}^2$, and dual systems of linear union complexity.*

Furthermore it makes expected $O(\frac{1}{\epsilon})$ calls to the Oracle.

Proof. For each of these specific cases, we have $\varphi_{\mathcal{R}}(n,k) = O(k^c)$ for some constant c, and thus Theorem 7 implies the bounds. ◀

6 Conclusion

Some final remarks:

- The algorithms, as they are stated, need the bound on the shallow-cell complexity (in fact, in the case of rectangles, even a finer decomposition bound) to set the initial sample size. However, by a standard exponential search trick, one can start with an initial guess of $O(1)$ for N_0, run the algorithm for $O\left(\frac{1}{\epsilon}\right)$ iterations and if the resulting set is not an ϵ-net, re-run the algorithm with a doubled initial sample size. This incurs an additional $O\left(\log\frac{1}{\epsilon}\right)$ penalty in the running time.

- Our algorithm, unlike earlier work, is adaptive: after the initial random sample, further additional points are added incrementally, and each additional set of $O(1)$ points that are added takes into account the previously added points.
- There has been recent work towards new algorithmic approaches for sketches and samples that work well in practice [13]. We leave for future work the experimental evaluation of our algorithm and comparison with earlier approaches, both in efficiency and whether the adaptive nature of our algorithm leads to improved size bounds in practice.

References

1. P. K. Agarwal. Range Searching. In J. E. Goodman, J. O'Rourke, and C. D. Tóth, editors, *Handbook of Discrete and Computational Geometry*. CRC Press LLC, 2017.
2. P. K. Agarwal and J. Pan. Near-Linear Algorithms for Geometric Hitting Sets and Set Covers. In *Symposium on Computational Geometry*, page 271, 2014.
3. N. Alon and J. Spencer. *The Probabilistic Method*. Wiley-Interscience, third edition, 2008.
4. B. Aronov, E. Ezra, and M. Sharir. Small-size ϵ-nets for axis-parallel rectangles and boxes. *SIAM Journal on Computing*, 39(7):3248–3282, 2010.
5. N. Bus, S. Garg, N. H. Mustafa, and S. Ray. Tighter estimates for ϵ-nets for disks. *Computational Geometry: Theory and Applications*, 53:27–35, 2016.
6. T. M. Chan, E. Grant, J. Könemann, and M. Sharpe. Weighted Capacitated, Priority, and Geometric Set Cover via Improved Quasi-uniform Sampling. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1576–1585, 2012.
7. B. Chazelle. *The Discrepancy Method: Randomness and Complexity*. Cambridge University Press, 2000.
8. K. L. Clarkson. A Randomized Algorithm for Closest-Point Queries. *SIAM J. Comput.*, 17(4):830–847, 1988.
9. K. L. Clarkson and P. W. Shor. Application of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4:387–421, 1989.
10. K. L. Clarkson and K. R. Varadarajan. Improved approximation algorithms for geometric set cover. In *Proceedings of the 21st Annual ACM Symposium on Computational Geometry (SoCG)*, pages 135–141, 2005.
11. D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete & Computational Geometry*, 2:127–151, 1987.
12. J. Komlós, J. Pach, and G. Woeginger. Almost tight bounds for ε-nets. *Discrete & Computational Geometry*, 7:163–173, 1992.
13. M. Matheny and J. M. Phillips. Practical Low-Dimensional Halfspace Range Space Sampling. In *26th Annual European Symposium on Algorithms (ESA)*, pages 62:1–62:14, 2018.
14. J. Matoušek. *Lectures on Discrete Geometry*. Springer, 2002.
15. J. Matoušek. Reporting Points in Halfspaces. *Computational Geometry: Theory and Applications*, 2(3):169–186, 1992.
16. J. Matoušek. *Geometric Discrepancy: An Illustrated Guide*. Algorithms and Combinatorics. Springer, 1999.
17. N. H. Mustafa. A Simple Proof of the Shallow Packing Lemma. *Discrete & Computational Geometry*, 55(3):739–743, 2016.
18. N. H. Mustafa, K. Dutta, and A. Ghosh. A Simple Proof of Optimal Epsilon-nets. *Combinatorica*, 38(5):1269–1277, 2018.
19. N. H. Mustafa and K. Varadarajan. Epsilon-approximations and Epsilon-nets. In J. E. Goodman, J. O'Rourke, and C. D. Tóth, editors, *Handbook of Discrete and Computational Geometry*. CRC Press LLC, 2017.
20. J. Pach and P. K. Agarwal. *Combinatorial geometry*. John Wiley & Sons, 1995.
21. J. Phillips. Coresets and Sketches. In J. E. Goodman, J. O'Rourke, and C. D. Tóth, editors, *Handbook of Discrete and Computational Geometry*. CRC Press LLC, 2017.

22 E. Pyrga and S. Ray. New existence proofs for ε-nets. In *Proceedings of the 24th Annual ACM Symposium on Computational Geometry (SoCG)*, pages 199–207, 2008.

23 K. R. Varadarajan. Weighted Geometric Set Cover via Quasi-uniform Sampling. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 641–648, 2010.

24 H. Yu, P. K. Agarwal, R. Poreddy, and K. R. Varadarajan. Practical Methods for Shape Fitting and Kinetic Data Structures using Coresets. *Algorithmica*, 52(3):378–402, 2008.

Tight Bounds for Online Weighted Tree Augmentation

Joseph (Seffi) Naor
Technion, Haifa, Israel
naor@cs.technion.ac.il

Seeun William Umboh
The University of Sydney, Australia
william.umboh@sydney.edu.au

David P. Williamson
Cornell University, Ithaca, NY, USA
http://www.davidpwilliamson.net/work
davidpwilliamson@cornell.edu

Abstract

The Weighted Tree Augmentation problem (WTAP) is a fundamental problem in network design. In this paper, we consider this problem in the online setting. We are given an n-vertex spanning tree T and an additional set L of edges (called links) with costs. Then, terminal pairs arrive one-by-one and our task is to maintain a low-cost subset of links F such that every terminal pair that has arrived so far is 2-edge-connected in $T \cup F$. This online problem was first studied by Gupta, Krishnaswamy and Ravi (SICOMP 2012) who used it as a subroutine for the online survivable network design problem. They gave a deterministic $O(\log^2 n)$-competitive algorithm and showed an $\Omega(\log n)$ lower bound on the competitive ratio of randomized algorithms. The case when T is a path is also interesting: it is exactly the online interval set cover problem, which also captures as a special case the parking permit problem studied by Meyerson (FOCS 2005). The contribution of this paper is to give tight results for online weighted tree and path augmentation problems. The main result of this work is a deterministic $O(\log n)$-competitive algorithm for online WTAP, which is tight up to constant factors.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Online algorithms, competitive analysis, tree augmentation, network design

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.88

Category Track A: Algorithms, Complexity and Games

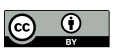
Related Version A full version of this paper is available at http://arxiv.org/abs/1904.11777.

Funding *Joseph (Seffi) Naor*: Supported in part by ISF grant 1585/15 and BSF grant 2014414. *Seeun William Umboh*: Supported in part by NWO grant 639.022.211 and ISF grant 1817/17. Part of this work was done while a postdoc at Eindhoven University of Technology, and while visiting the Hebrew University of Jerusalem and the Technion.

Acknowledgements This work was done in part while the authors were visiting the Simons Institute for the Theory of Computing.

1 Introduction

In the *weighted tree augmentation problem* (WTAP), we are given an n-vertex spanning tree $T = (V, E)$ together with an additional set of edges L called *links*, where $L \subseteq \binom{V}{2}$. Each link $\ell \in L$ has a cost $c(\ell) \geq 0$. Terminal pairs (s_i, t_i), $i = \{1, \ldots, k\}$, are given and the goal is to compute a minimum cost subset of links $F \subseteq L$ such that each terminal pair is (edge) 2-connected in $T \cup F$. In the unweighted version, the links have unit costs and the problem



46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 88; pp. 88:1–88:14


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

is known as the *tree augmentation problem* (TAP). If the spanning tree T is a path, then the unweighted problem is called the *path augmentation problem* (PAP), while the weighted version is called *weighted path augmentation* (WPAP).

TAP and WTAP are considered to be fundamental connectivity augmentation problems, and have been studied extensively. TAP is already known to be APX-hard and the best approximation algorithms for WTAP and TAP achieve approximation factors of 2 and 1.458 respectively [6, 7]. Improving these bounds is an important open problem.

We consider these problems in the online setting. In online WTAP, we are initially given a spanning tree $T = (V, E)$, and the set of links L together with their costs. The terminal pairs (s_i, t_i) arrive online one by one. Our goal is to maintain a low-cost subset of links $F \subseteq L$ such that each terminal pair seen so far is (edge) 2-connected in $T \cup F$.

Online WTAP occurs as a subproblem in the online survivable network design algorithm of Gupta, Krishnaswamy and Ravi [8]. They observed that the online tree augmentation problem can be cast as an instance of the online set cover problem[1] in which the elements are the fundamental cuts defined by the terminal pairs and the sets are the links. Since there are only n elements and $O(n^2)$ sets, applying the results of Alon et al. [1] yields a fractional $O(\log n)$-competitive algorithm. But, then, how does one round the fractional solution online? Randomized rounding seems to be the only rounding technique we have for this problem, and it yields a randomized $O(\log^2 n)$-competitive algorithm, as observed by [1]. This competitive factor can even be achieved deterministically at no further cost [1]. We note that the loss of a logarithmic factor in the rounding step seems inherent. Interestingly, Gupta, Krishnaswamy and Ravi [8] also showed for the rooted setting ($s_i = r$ for some root r) a lower bound of $\Omega(\log n)$ against randomized algorithms. It is easy to observe that this lower bound also holds against fractional online algorithms.

There has been a long line of work on maintaining connectivity online, starting in the seminal paper of Imase and Waxman [11]. A $\Theta(\log n)$-competitive algorithm is given there for the online Steiner problem in undirected graphs. In this problem the graph with a fixed root vertex is known in advance and the terminals are given one by one, and one must ensure that all terminals that have arrived so far are connected to the root. Other polylogarithmic (in n) competitive algorithms have been given for more complex models of connectivity, including those with node costs rather than edge costs and penalties for violating connectivity constraints; see [2, 3, 13, 9, 10, 16, 14]. Gupta, Krishnaswamy, and Ravi [8] consider the online survivable network design problem, which generalizes WTAP. In this problem, a graph is fixed in advance and terminal pairs (s_i, t_i) arrive with connectivity requirements r_i; one must ensure that there are at least r_i edge-disjoint paths between s_i and t_i for all pairs that have arrived thus far. They give a randomized $\tilde{O}(r_{\max} \log^3 n)$-competitive algorithm for the problem, where $r_{\max} = \max_i r_i$. Note that this problem with uniform requirements $r_i = 2$ already generalizes WTAP.

The online WPAP, when T is a path, is an interesting problem in its own right. This problem is equivalent to online interval set cover. It captures as a special case the parking permit problem introduced by Meyerson [12]. In this problem, there is a sequence of days; each day it is either sunny or it rains, and if it rains we must purchase a parking permit. Permits have various durations and costs. We can model the parking permit problem by online path augmentation by letting the edges of the path correspond to the sequence of days, the links to the permits, and the rainy days to a terminal pair request for the

[1] In the online set cover problem, elements arrive online and need to be covered upon arrival by sets from a set system known in advance. (Note that not necessarily all elements will appear.)

corresponding day. Meyerson [12] gives a deterministic $O(\log n)$-competitive algorithm for the problem and a randomized $O(\log \log n)$-competitive algorithm, and shows lower bounds on the competitive ratio of $\Omega(\log n / \log \log n)$ for deterministic algorithms and $\Omega(\log \log n)$ for randomized algorithms. Note that online WPAP is a strict generalization of the parking permit problem because the parking permit problem assumes that permits of the same duration have the same cost, whereas no such assumption is made of the links in WPAP.

1.1 Our Results

The contribution of this paper is to give tight results (within constant factors) for online tree and path augmentation problems. Our main result is that weighted online tree augmentation has a competitive ratio of $\Theta(\log n)$.

▶ **Theorem 1.** *There is a deterministic algorithm for online WTAP with competitive ratio* $O(\log n)$.

This result is tight up to constant factors because of the $\Omega(\log n)$ lower bound on randomized algorithms for WTAP given by [8]. As we mention above, [8] gives a randomized $\tilde{O}(r_{\max} \log^3 n)$-competitive algorithm for the online survivable network design problem. An intriguing open question is whether this competitive ratio can be improved, say to $O(r_{\max} \log n)$ or even $O(\log n)$. In fact, we are unaware of lower bounds that rule out the latter bound. We view our main result as a necessary stepping stone towards obtaining an $O(r_{\max} \log n)$ or $O(\log n)$ bound. Indeed, for $r_{\max} = 2$, plugging in our algorithm for online WTAP into the algorithm of [8] improves their competitive ratio from $\tilde{O}(\log^3 n)$ to $\tilde{O}(\log^2 n)$.

▶ **Corollary 2.** *For online survivable network design with* $r_{\max} = 2$*, there is a randomized algorithm with competitive ratio* $\tilde{O}(\log^2 n)$.

Our second result shows that the competitive ratio for deterministic algorithms for online path augmentation is also $\Theta(\log n)$. Meyerson [12] gives a lower bound of $\Omega(\log n / \log \log n)$ for deterministic algorithms for the parking permit problem, and hence for online path augmentation. We improve the analysis of his lower bound instance to show the following.

▶ **Theorem 3.** *Every deterministic algorithm for online WPAP has competitive ratio* $\Omega(\log n)$.

Since we use a parking permit instance to show the lower bound, we have the same lower bound for the parking permit problem.

Finally, we show that the fractional version of online path augmentation has competitive ratio $\Theta(\log \log n)$ for deterministic algorithms. Meyerson [12] gives a lower bound of $\Omega(\log \log n)$ for randomized algorithms for the parking permit problem, and hence for online fractional path augmentation. Our algorithm implies an exponential gap between the competitive ratios of fractional path augmentation and fractional tree augmentation. We show the following.

▶ **Theorem 4.** *There is a deterministic algorithm for online fractional WPAP with competitive ratio* $O(\log \log n)$.

Recall that online WPAP is equivalent to online interval set cover. Thus, Theorems 1 and 4 imply that restricting online set cover to interval sets allows for improved competitive ratios. Also, even though interval set cover and interval hitting set are equivalent in the offline case, the latter turns out to be exponentially more difficult than the former in the online case; in contrast to Theorem 4, Even and Smorodinsky [5] gave a lower bound of $\Omega(\log n)$ for online fractional hitting set.

1.2 Our Techniques

We now outline some of the ideas behind our algorithms.

Online WTAP

As mentioned before, there is an online *fractional* $O(\log n)$-competitive algorithm for WTAP that follows from the work of [1] on the online set cover problem. However, it is unclear how to exploit the special structure of the set system in hand in WTAP (as defined by the links) to avoid the loss of another factor of $O(\log n)$ when rounding the fractional solution into an integral one (either randomized or deterministic). Thus, our approach to proving Theorem 1 takes a completely different route. There are two key ingredients in our proof:

1. **Low-width path decomposition.** The first ingredient is a path decomposition of low "width": in particular, there is a decomposition of the tree into edge-disjoint paths such that any path in the tree intersects at most $O(\log n)$ paths of the decomposition. Such a decomposition can be obtained using the heavy-path decomposition of Sleator and Tarjan [15]. This immediately implies an $O(\log n)$-approximate black-box reduction from online tree augmentation to online path augmentation. Unfortunately, Theorem 3 gives a lower bound of $\Omega(\log n)$ for the latter problem. Since a tree may have width $\Omega(\log n)$ in the worst case (e.g., a binary tree), the best we can achieve for WTAP using a black-box reduction is a competitive ratio of $O(\log^2 n)$.
2. **Refined guarantee for path augmentation.** The second ingredient is our main technical contribution. We define a notion of *projection* for links onto paths in the path decomposition, and call the projected link *rooted* if it has, as its endpoint, the node of the path closest to the root of the tree. The key insight is that the path decomposition has a special structure: for each link, its projection is rooted for all but at most one of the paths in the decomposition. We then give a version of the path algorithm that treats rooted links differently from non-rooted links; in particular, an online path augmentation algorithm that finds a solution whose cost is within a constant factor of the rooted links of the optimal solution plus an $O(\log n)$ factor of the cost of the non-rooted links. Intuitively, then, summing the cost over all the paths in the decomposition, each link appears as a rooted link in at most $O(\log n)$ paths in the decomposition and as a non-rooted link in at most one path in the decomposition, yielding the $O(\log n)$ factor overall.

Online Fractional WPAP

Directly applying the online fractional set cover algorithm of [1] to online fractional WPAP only yields a competitive ratio of $O(\log n)$. However, for online set cover instances in which each element is covered by at most d sets, the algorithm of [1] is $O(\log d)$-competitive. Thus, to get a competitive ratio of $O(\log \log n)$, the basic idea is to reduce to a restricted instance in which each request can only be covered by $O(\log n)$ links. For such restricted instances, applying the algorithm of [1] gives a competitive ratio of $O(\log \log n)$.

1.3 Other Related Work

Recently, Dehghani et al. [4] studied online survivable network design, giving a bicriteria approximation algorithm, and considering several stochastic settings.

1.4 Organization of the Paper

We start with the preliminaries and describe the low-width path decomposition in Section 2. In Section 3, we present the refined guarantee needed for online path augmentation. Then, we show how to achieve the required refined guarantee in Section 4. Due to lack of space, we defer the proofs of Theorems 3 and 4 to the full version.

2 Preliminaries

We restate the formal definition of the problem. In the online weighted tree augmentation problem, we are initially given a spanning tree $T = (V, E)$, and an additional set of edges called *links* $L \subseteq \binom{V}{2}$ with costs $c(\ell) \geq 0$. Then, terminal pairs (s_i, t_i) arrive one by one. Our goal is to maintain a low-cost subset of links $F \subseteq L$ such that each terminal pair seen so far is 2-connected in $T \cup F$.

Notation

Denote by $P(u, v)$ the path between u and v in T. For a link $\ell = (u, v)$, we write $P(\ell) = P(u, v)$ and for a set S of links, we write $P(S) = \bigcup_{\ell \in S} P(\ell)$. We say that a link $\ell \in L$ *covers* an edge $e \in E$ if $e \in P(\ell)$. Define $\text{cov}(e) = \{\ell \in L : e \in P(\ell)\}$ to be the set of links covering e. Note that $\text{cov}(e)$ is exactly the set of links crossing the cut induced by the tree edge e. Let $R \subseteq E$ be a set of requests. Then, a solution F is feasible if and only if for every edge $e \in R$, we have $F \cap \text{cov}(e) \neq \emptyset$; or equivalently, if $P(F) \supseteq R$.

Simplifying assumptions

In the rest of this paper, we assume that link costs are powers of 2; this assumption is without loss of generality since we can round up all edge costs and lose only a factor of 2 in the competitive ratio. Given that link costs are powers of 2, we say that the *class* of a link ℓ is j if $c(\ell) = 2^j$ and we write $\text{class}(\ell) = j$.

Given an instance in which link costs are powers of 2, we also assume that requests are *elementary*: each request (s_i, t_i) is a tree edge $e \in E$. This is without loss of generality because an adversary can simulate a non-elementary request (s_i, t_i) by a sequence of requests, where each request is an edge along the path between s_i and t_i in T.

Path decomposition

We next define a rooted path decomposition, see Figure 1 for an example.

▶ **Definition 5** (Rooted Path Decomposition). *Let T be a tree. A* path decomposition *of T is a partition $\mathcal{P}$ of its edge set into edge-disjoint paths. We say $\mathcal{P}$ is* rooted *if there is a vertex $r \in T$ such that if we root T at r, then for each path $p \in \mathcal{P}$, the least common ancestor of the vertices on p is an endpoint of the path (we call this endpoint the* root *of p). The* width *of $\mathcal{P}$ is* $\text{width}(\mathcal{P}) = \max_{u,v \in V(T)} |\{p \in \mathcal{P} : P(u, v) \cap p \neq \emptyset\}|$*, the maximum number of paths $p \in \mathcal{P}$ that any path in T intersects.*

▶ **Lemma 6** (Existence of Low Width Rooted Path Decompositions). *Every tree on n vertices admits a rooted path decomposition of width $O(\log n)$.*

An $O(\log n)$-width rooted path decomposition can be obtained using the so-called *heavy path decomposition* of Sleator and Tarjan [15]. For the sake of completeness, we give a proof here. The following notion of a caterpillar decomposition will be convenient.

Figure 1 Example of a graph and its rooted path decomposition. The edge colors reflect the partition of the edges. The root of each path is the highest vertex of the path.

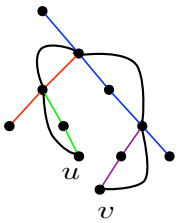


Figure 2 Illustration of the projections of the link (u, v) onto the paths of the decompositon. Only the projection onto the blue path is non-rooted.

▶ **Definition 7** (Caterpillar Decomposition)**.** *Let T be a rooted tree on n vertices. A* caterpillar decomposition *of T is a vertex-disjoint decomposition of T into a root-to-leaf path B (called the* backbone*) and subtrees T_i that are connected to B. The decomposition is* balanced *if for each subtree T_i, we have $|V(T_i)| \leq n/2$.*

▶ **Lemma 8.** *Every tree admits a balanced caterpillar decomposition.*

Proof. The existence of a balanced caterpillar decomposition is an easy consequence of the fact that every tree T has a balanced vertex separator v, i.e. after removing v from T, each of the remaining connected components has at most $n/2$ vertices. The following is a balanced caterpillar decomposition of T: pick an arbitrary root-to-leaf path containing v to be the backbone B, and the subtrees T_i to be the connected components of T after removing B. ◀

Proof of Lemma 6. The lemma easily follows by choosing an arbitrary root vertex of T and recursively applying Lemma 8. ◀

3 Refined Guarantee for Online Path Augmentation

As already mentioned, Lemma 6 implies an $O(\log n)$-approximate black-box reduction to online path augmentation: given an α-competitive algorithm for online path augmentation, we have an $O(\alpha \log n)$-competitive algorithm for online tree augmentation. However, Theorem 3 says that $\alpha = \Omega(\log n)$ for deterministic algorithms. To get around this lower bound, a more refined guarantee for online path augmentation is needed.

We need some notation to describe this refined guarantee. Suppose $\mathcal{P}$ is a rooted path decomposition of T and ℓ a link. For $Q \in \mathcal{P}$, let $\pi_Q(\ell)$ be the link whose endpoints are endpoints of the path $P(\ell) \cap Q$; we call $\pi_Q(\ell)$ the *projection* of ℓ onto Q. We say that ℓ is *Q-rooted* if one of the endpoints of $\pi_Q(\ell)$ is the root of Q, and *Q-non-rooted* otherwise. (See Fig. 2 for an illustration.) The main ingredient for the refined guarantee is the next lemma.

▶ **Lemma 9.** *Consider a tree T and link $\ell = (u, v)$. Suppose $\mathcal{P}$ is a rooted path decomposition of T. Then, there is at most one path $Q \in \mathcal{P}$ for which ℓ is Q-non-rooted.*

Proof. We claim that for any path $Q \in \mathcal{P}$ such that ℓ is a non-rooted link, the least common ancestor a of u and v must lie in Q but is not an endpoint of Q, i.e. it lies strictly in the middle of Q. Since $\mathcal{P}$ is an edge-disjoint decomposition of T, there can be at most one such path and thus the claim implies the lemma.

We proceed to prove the claim. Let u', v' be the endpoints of $\pi_Q(\ell)$. Since $\mathcal{P}$ is a rooted path decomposition, either u' is an ancestor of v' or vice versa; suppose the former. We now argue that u' is the least common ancestor of u and v. There are two cases: (1) either u' is an endpoint of ℓ; (2) or there is a vertex z of $P(u, v)$ adjacent to u' but is not on Q. In case (1), we are done. Consider case (2). Since u' is not an endpoint of Q, its parent must be on Q, and thus z is a child of u'. Therefore, u' is the least common ancestor of u and v. ◀

Motivated by Lemma 9, we define the online rooted path augmentation problem. An instance of online rooted path augmentation consists of a rooted path Q where the root r is an endpoint of Q. For such an instance, we say that a link is rooted if one of its endpoints is r. Lemma 9 suggests that we should devise an algorithm for online rooted path augmentation with the following refined guarantee.

▶ **Definition 10** (Nice Solution). *A solution F for an instance of online rooted path augmentation is* nice *if for any feasible solution F^*, we have $c(F) \leq O(1)c(R^*) + O(\log n)c(S^*)$ where R^* is the set of rooted links and S^* is the set of non-rooted links of F^*, respectively. An algorithm is nice if it always produces a nice solution.*

▶ **Lemma 11.** *Suppose that there exists a deterministic nice algorithm Path-ALG for online rooted path augmentation. Then, there exists an $O(\log n)$-competitive deterministic algorithm for online tree augmentation.*

Proof. Here is our algorithm for general instances. Consider a general instance of online weighted tree augmentation with tree $T = (V, E)$, requests $e_1, \ldots, e_k \subseteq E$ and links $L = \binom{V}{2}$ with costs $c(\ell)$. Our algorithm works as follows. By Lemma 6, there exists a rooted path decomposition $\mathcal{P}$ of T with width $w = O(\log n)$. Now, each rooted path $Q \in \mathcal{P}$ defines an instance of online rooted path augmentation: the links are $L_Q = \{\pi_Q(\ell) : \ell \in L\}$ where $\pi_Q(\ell)$ has cost $c(\ell)$, and the sequence of requests is exactly the subsequence of requests that lie on Q. So, our algorithm runs in parallel $|\mathcal{P}|$ instantiations of Path-ALG, one per rooted path $Q \in \mathcal{P}$. When request e_i arrives, if $e_i \in Q$ (since e_i is an elementary request, it must lie on some path of $\mathcal{P}$), then our algorithm uses the instantiation of Path-ALG on Q to handle that request; in particular, if Path-ALG buys the projected link $\pi_Q(\ell)$, then our algorithm buys the link ℓ.

Let us now analyze the competitive ratio of this algorithm. Let F^* be a feasible solution. For $Q \in \mathcal{P}$, we denote by R^*_Q, and S^*_Q the subset of F^* which is Q-rooted, and Q-non-rooted, respectively. Since Path-ALG is nice, we have that our algorithm's solution F has cost

$$c(F) \leq \sum_{Q \in \mathcal{P}} \left[O(1)c(R^*_Q) + O(\log n)c(S^*_Q) \right] \leq O(\log n)c(F^*),$$

where the last inequality is because Lemma 9 implies that each link of F^* is in S^*_Q for at most one $Q \in \mathcal{P}$ and is in R^*_Q for at most $w = O(\log n)$ paths $Q \in \mathcal{P}$. ◀

In the next section, we construct a nice deterministic algorithm. Together with Lemma 11 this gives a deterministic $O(\log n)$-competitive algorithm for online tree augmentation, thus proving Theorem 1.

4 A Nice Algorithm for Online Path Augmentation

In this section, we devise a nice algorithm for online rooted path augmentation. In the following, we use the convention that the root of the path is the left endpoint of the path.

We begin by showing in Section 4.1 that it suffices to consider simpler instances that we call *minimal instances*. Then, we describe in Section 4.2 how to prove niceness using an LP for the problem. Finally, we describe and analyze the algorithm in Sections 4.3 and 4.4.

4.1 Minimal Instances

The first step is a preprocessing step that simplifies the structure of the link set. In particular, we prune the instance so that it is of the following type.

▶ **Definition 12** (Minimal Instances). *A set of links L and its costs c are* minimal *if they satisfy the following properties:*

1. *for each class j, there is at most one rooted link and for every edge e, there are at most two links ℓ with $e \in P(\ell)$;*
2. *for any two rooted links ℓ and ℓ', if $\text{class}(\ell) > \text{class}(\ell')$, then $P(\ell) \supsetneq P(\ell')$.*

An instance is minimal if its links and costs are minimal.

Given a set of links L and its costs c, we prune L to get a minimal subset of links $L' \subseteq L$ as follows. We begin by pruning the rooted links: while there exists a rooted link ℓ and a rooted link ℓ' of the same or lower class such that $P(\ell') \supseteq P(\ell)$, remove ℓ. Then we prune the non-rooted links for each class j: let L_j be the set of class j links and L'_j be a minimum-size subset of L_j that covers L_j, i.e. $P(L'_j) \supseteq P(L_j)$; then, remove the links $L_j \setminus L'_j$. Such a minimum cover may be computed efficiently using an algorithm for minimum interval cover. By minimality, we have that for any edge e, there are at most two links $\ell, \ell' \in L'_j$ such that $e \in P(\ell) \cap P(\ell')$. The following claim shows that any link $\ell \in L_j$ that was pruned away can be replaced with at most three links of L'_j and so restricting to L' only causes the value of the optimal solution to increase by at most a factor of 3. We defer the proof to the full version.

▷ Claim 13. For every link $\ell \in L_j \setminus L'_j$, there exists (at most) three links $\ell_1, \ell_2, \ell_3 \in L'_j$ with $P(\ell) \subseteq P(\ell_1) \cup P(\ell_2) \cup P(\ell_3)$.

Given a subset of links $L' \subseteq L$, we say that a solution $F \subseteq L'$ is *nice for* L' if for any feasible solution $F' \subseteq L'$, we have $c(F) \leq O(1)c(R') + O(\log n)c(S')$ where R' is the set of rooted links and S' is the set of non-rooted links of F', respectively. The following lemma says that it suffices to have a solution that is nice for a pruning of L and thus it suffices to devise a nice algorithm for minimal instances. We defer the proof of the lemma to the full version.

▶ **Lemma 14.** *Let $L' \subseteq L$ be a pruning of L. Then, a solution that is nice for L' is also nice for L.*

Henceforth, we will focus on devising a nice algorithm for minimal instances.

4.2 Proving Niceness via the Dual LP

Our algorithm uses the standard LP formulation of the problem. Let $\mathcal{R}$ be the set of requests. The following are the primal and dual LPs, respectively.

$$\begin{aligned} &\text{minimize} && \sum_{\ell \in L} x(\ell) c(\ell) \\ &\text{subject to} && \sum_{\ell \in \text{cov}(e)} x(\ell) \geq 1 \qquad \forall e \in \mathcal{R} \end{aligned} \tag{1}$$

$$\begin{aligned} &\text{maximize} && \sum_{e \in \mathcal{R}} y(e) \\ &\text{subject to} && \sum_{e \in P(\ell)} y(e) \leq c(\ell) \qquad \forall \ell \in L \end{aligned} \tag{2}$$

We say that a link ℓ is *tight* with respect to a dual solution y if $\sum_{e \in P(\ell)} y(e) = c(\ell)$.

The following lemma tells us how to use the dual to prove niceness.

▶ **Lemma 15.** *Let F be a solution. Suppose y is a dual solution such that*

1. $c(F) \leq O(1) \sum_e y(e)$,
2. $\sum_{e \in P(\ell)} y(e) \leq O(\log n) c(\ell)$ *for every non-rooted link ℓ, and*
3. $\sum_{e \in P(\ell)} y(e) \leq O(1) c(\ell)$ *for every rooted link ℓ.*

Then, F is a nice solution.

Proof. Let F^* be a feasible solution, R^* be the subset of F^* that is rooted and S^* the subset that is non-rooted. We now show that $\sum_e y(e) \leq O(1) c(R^*) + O(\log n) c(S^*)$, which then implies that $c(F) \leq O(1) c(R^*) + O(\log n) c(S^*)$. Since we have a dual variable $y(e)$ for each request e and F^* is feasible, we have that

$$\sum_e y(e) \leq \sum_{e \in P(R^*)} y(e) + \sum_{e \in P(S^*)} y(e).$$

Using the fact that $\sum_{e \in P(\ell)} y(e) \leq O(1) c(\ell)$ for every rooted link ℓ, we also have

$$\sum_{e \in P(R^*)} y(e) \leq \sum_{\ell \in R^*} \sum_{e \in P(\ell)} y(e) \leq O(1) c(R^*).$$

Similarly, we get that $\sum_{e \in P(S^*)} y(e) \leq O(\log n) c(S^*)$. Putting all of these together, we conclude that $\sum_e y(e) \leq O(1) c(R^*) + O(\log n) c(S^*)$, as desired. ◀

4.3 Algorithm

We now give some of the ideas behind our algorithm.

An $O(\log n)$-competitive algorithm

First, we describe a simple algorithm that constructs a solution F and a dual solution y that satisfies $c(F) \leq O(1) \sum_e y(e)$ and $\sum_{e \in P(\ell)} y(e) \leq O(\log n) c(\ell)$ for *every* link ℓ. The algorithm maintains a maximal feasible dual solution y and is as follows: when a request e_i arrives, raise its dual variable $y(e_i)$ until some link ℓ with $e_i \in P(\ell)$ goes tight; add this link to F. There are two parts to the analysis. First, let $\widehat{F}$ be the set of links in F that

cost at least $\max_{\ell \in F} c(\ell)/n^2$. Since $|F| \leq n^2$, we get that $c(F) \leq 2c(\widehat{F})$ so it suffices to bound $c(\widehat{F})$. The second part of the analysis uses the following charging argument to bound $c(\widehat{F})$: whenever we add a tight link ℓ to $\widehat{F}$, we charge its cost to the dual variables $y(e)$ for $e \in P(\ell)$. Let $\lambda(e)$ be the total number of links charged to $y(e)$ and $\widehat{y}$ be the dual solution where $\widehat{y}(e) = \lambda(e)y(e)$. We have $c(\widehat{F}) \leq O(1)\sum_e \lambda(e)y(e)$. Now observe that $\lambda(e) \leq O(\log n)$ because Property 1 of minimal instance implies that there can be at most 2 links $\ell \in \widehat{F}$ with $e \in P(\ell)$ for a single cost class, and, by definition, $\widehat{F}$ can have at most $O(\log n)$ cost classes. So, for each link ℓ, we have

$$\sum_{e \in P(\ell)} \lambda(e)y(e) \leq O(\log n) \sum_{e \in P(\ell)} y(e) \leq O(\log n)c(\ell)$$

where the last inequality follows from the fact that y is feasible.

Saving the rooted links

A natural idea to ensure that $\sum_{e \in P(\ell)} \lambda(e)y(e) \leq O(1)c(\ell)$ for each rooted link ℓ is to modify the above algorithm to explicitly take into account the charging method as follows: after buying the tight link (we call this a type-1 link), if there is a rooted link ℓ' such that $\sum_{e \in P(\ell')} \lambda(e)y(e) > c(\ell')$, buy the one of highest class among such links (we call this a type-2 link). Moreover, we also modify the charging method to only charge each type-1 link ℓ to the dual variables $y(e)$ for $e \notin P(\ell')$ where ℓ' is the last type-2 link bought.

As we will see later, these modifications allow us to argue that $\sum_{e \in P(\ell)} \lambda(e)y(e) \leq O(1)c(\ell)$ for each rooted link ℓ. However, it also introduces a complication: it might be possible that for some type-1 link ℓ, most of the dual variables $y(e)$ paying towards its cost have $e \in P(\ell')$ where ℓ' is the last type-2 link bought. Since the charging method only charges to dual variables $y(e)$ for $e \notin P(\ell')$, this would mean that it might charge an amount that is much less than the cost of ℓ'.

Fixing the complication

To fix the above issue, whenever we buy a type-2 link ℓ', we also buy all links ℓ'' of class at most class(ℓ') that crosses ℓ', i.e. $\emptyset \subsetneq P(\ell'') \cap P(\ell') \subsetneq P(\ell')$. Property 1 implies that the total cost of these links is at most $O(1)c(\ell')$. We call these links type-3 links. This ensures that later on, when we buy a type-1 link ℓ, if $P(\ell) \cap P(\ell') \neq \emptyset$, then ℓ must be of higher class than ℓ' and thus most of its cost is paid for by dual variables $y(e)$ for $e \notin P(\ell')$.

We describe the complete algorithm formally in Algorithm 1. In Algorithm 1, we use Z to keep track of $P(\ell)$ where ℓ is the last type-2 link bought so far ($Z = \emptyset$ if no type-2 link is bought yet). The links bought in Step 4, 9, 11, are type-1, type-2, and type-3 links, respectively.

4.4 Analysis of Algorithm

We now prove that Algorithm 1 is nice. Let $F_1, F_2, F_3 \subseteq F$ be the sets of type-1, type-2 and type-3 links, respectively. The proof consists of three steps. First, we show that $c(F) \leq O(1)c(F_1)$ (Lemma 17) and thus it suffices to bound the cost of type-1 links. Then, we construct a dual solution $\widehat{y}$ that accounts for the cost of type-1 links (Lemma 18). This shows that $\widehat{y}$ satisfies the first condition of Lemma 15. Finally, Lemmas 20 and 19 show that $\widehat{y}$ satisfies the remaining conditions of Lemma 15.

For each type-1 link $\ell \in F_1$, define $C(\ell)$ to be the set of edges e such that $\lambda(e)$ was incremented during the iteration that ℓ was assigned to F_1, i.e. each dual variable $y(e)$ for $e \in C(\ell)$ contributes towards paying $c(\ell)$. Observe that $\lambda(e) = |\{\ell : e \in C(\ell)\}|$ and $C(\ell) \subseteq P(\ell)$.

Algorithm 1 Nice algorithm for online rooted path augmentation.

$F \leftarrow \emptyset; y \leftarrow 0; \lambda \leftarrow 0; Z \leftarrow \emptyset$
for each unsatisfied request e_i **do**
  Increase $y(e_i)$ until some link ℓ with $e_i \in P(\ell)$ goes tight
  Add such a link ℓ to F
  for each $e \in P(\ell) \setminus Z$ such that $y(e) > 0$ **do**
    $\lambda(e) \leftarrow \lambda(e) + 1$
  end for
  if there exists a rooted link $\ell \notin F$ such that $\sum_{e \in P(\ell)} \lambda(e) y(e) \geq c(\ell)$ **then**
    Among such links, add to F the link ℓ of highest class
    for $j \leq \text{class}(\ell)$ **do**
      Add to F all class-j links ℓ' that cross ℓ, i.e. $\emptyset \subsetneq P(\ell') \cap P(\ell) \subsetneq P(\ell)$
    end for
    $Z \leftarrow P(\ell)$
  end if
end for

▶ **Proposition 16.** *Algorithm 1 satisfies the following properties. Let Z_i and λ_i denote Z and λ at the end of the i-th iteration. Then, for every iteration i, we have*

1. $Z_i \supseteq Z_{i-1}$;
2. *if* $y(e_i) > 0$, *then* $\lambda_i(e_i) > 0$.

Proof. The first follows from Property 2 of minimal instances. The second follows from the fact that in the iteration that e_i arrives, since it is unsatisfied, it must not be contained in Z. Let ℓ be the link added to F in that iteration. Since $e_i \in P(\ell) \setminus Z$ and $y(e_i) > 0$, we have that $\lambda(e_i)$ is increased by 1 during the iteration and thus $\lambda_i(e_i) > 0$. ◀

▶ **Lemma 17.** $c(F) \leq O(1)c(F_1)$.

Proof. We will show that $c(F_3) \leq O(1)c(F_2)$, that $c(F_2) \leq O(1)\sum_e \lambda(e)y(e)$ and that $\sum_e \lambda(e)y(e) \leq c(F_1)$. Let ℓ_r be the last type-2 link bought. We have that $c(\ell_r) \leq \sum_{e \in P(\ell_r)} \lambda(e)y(e)$ by construction. Moreover, since $c(\ell_r) \geq c(\ell)$ for every $\ell \in F_2$ and there is at most one rooted link of each class, we get that $c(F_2) \leq 2c(\ell_r)$. Thus, we get that $c(F_2) \leq 2\sum_{e \in P(\ell_r)} \lambda(e)y(e)$. For each type-2 link ℓ bought, we buy at most two type-3 links per class $j \leq \text{class}(\ell)$ because of Property 1 of minimal instances. Therefore, we have $c(F_3) \leq 2c(F_2) \leq 4\sum_{e \in P(\ell_r)} \lambda(e)y(e)$.

Finally, we show that $\sum_e \lambda(e)y(e) \leq c(F_1)$. Since $\lambda(e) = |\{\ell : e \in C(\ell)\}|$, we have $\sum_e \lambda(e)y(e) = \sum_{\ell \in F_1} \left(\sum_{e \in C(\ell)} y(e)\right)$. Now, since $C(\ell) \subseteq P(\ell)$ and y is feasible, we get $\sum_{e \in C(\ell)} y(e) \leq \sum_{e \in P(\ell)} y(e) \leq c(\ell)$. Combining the previous two inequalities gives us that $\sum_e \lambda(e)y(e) \leq c(F_1)$. ◀

Let $c_{\max} = \max_{\ell \in F_1} c(\ell)$. Define $\widehat{F}_1 = \{\ell \in F_1 : c(\ell) \geq c_{\max}/n^2\}$ and $\widehat{\lambda}(e) = |\{\ell \in \widehat{F}_1 : e \in C(\ell)\}|$. We now show that F and the dual solution $\widehat{y}$ where $\widehat{y}(e) = \widehat{\lambda}(e)y(e)$ satisfies the conditions of Lemma 15.

▶ **Lemma 18.** $c(F_1) \leq O(1)\sum_e \widehat{\lambda}(e)y(e)$.

Proof. Observe that $c(F_1) \leq 2c(\widehat{F}_1)$ so it suffices to prove that

$$c(\widehat{F}_1) \leq O(1)\sum_e \widehat{\lambda}(e)y(e). \tag{3}$$

We now show that this inequality holds at the end of each iteration of the algorithm. Consider an iteration in which the current request e_i is not already covered and suppose $\ell \in \widehat{F}_1$ is the type-1 link bought in this iteration. The LHS of Inequality (3) increases by $c(\ell)$ in this iteration. We now show that $\sum_e \widehat{\lambda}(e)y(e)$ increases by at least $c(\ell)/2$. In this iteration, $\widehat{\lambda}(e)$ increases by 1 for every $e \in P(\ell) \setminus Z$ and $y(e) > 0$, and so $\sum_e \widehat{\lambda}(e)y(e)$ increases by exactly $\sum_{e \in P(\ell)\setminus Z} y(e)$.

In the remainder of the proof, we show that $\sum_{e \in P(\ell)\setminus Z} y(e) \geq c(\ell)/2$. If $P(\ell) \cap Z = \emptyset$, then $\sum_{e \in P(\ell)\setminus Z} y(e) = \sum_{e \in P(\ell)} y(e) = c(\ell)$ since ℓ is tight. Now suppose $P(\ell) \cap Z \neq \emptyset$. Let ℓ' be the type-2 link such that $Z = P(\ell')$. Since $P(\ell) \cap P(\ell') \neq \emptyset$, it must be the case that ℓ is of type higher than class(ℓ'). This is because otherwise, ℓ would have been bought earlier as a type-3 link in the same iteration as ℓ'. But then since $e_i \in P(\ell)$, it would contradict the assumption that e_i is not already covered at the start of the current iteration. Thus, class$(\ell) >$ class(ℓ') and so $c(\ell) \geq 2c(\ell')$. So, we now have

$$\sum_{e \in P(\ell)\setminus Z} y(e) \geq \sum_{e \in P(\ell)} y(e) - \sum_{e \in P(\ell')} y(e) \geq c(\ell) - c(\ell') \geq c(\ell)/2,$$

where the second last inequality follows from the fact that y is a feasible dual and ℓ is tight. Therefore, Inequality (3) holds at the end of each iteration, as desired. ◀

Lemmas 17 and 18 imply that $c(F) \leq O(1) \sum_e \widehat{y}(e)$.

▶ **Lemma 19.** *For each non-rooted link ℓ, we have $\sum_{e \in P(\ell)} \widehat{\lambda}(e)y(e) \leq O(\log n)c(\ell)$.*

Proof. Property 1 of minimal instances implies that for each j, there are at most two links $\ell' \in \widehat{F}_1$ of class j with $e \in C(\ell')$. Since each link in $\widehat{F}_1$ has cost between $c_{\max}/n^2$ and $c_{\max}$ and link costs are powers of 2, we have that $\widehat{\lambda}(e) \leq O(\log n)$. Thus we get that $\sum_{e \in P(\ell)} \widehat{\lambda}(e)y(e) \leq O(\log n) \sum_{e \in P(\ell)} y(e) \leq O(\log n)c(\ell)$, where the last inequality follows from the fact that y is a feasible dual. ◀

▶ **Lemma 20.** *For each rooted link ℓ, we have $\sum_{e \in P(\ell)} \widehat{\lambda}(e)y(e) \leq O(1)c(\ell)$.*

Proof. We will in fact show that $\sum_{e \in P(\ell)} \lambda(e)y(e) \leq O(1)c(\ell)$. Suppose, at the end of some iteration, that we have $\sum_{e \in P(\ell)} \lambda(e)y(e) > c(\ell)$. Consider the earliest iteration that this happens. We now show that $\sum_{e \in P(\ell)} \lambda(e)y(e) \leq O(1)c(\ell)$ at the end of the iteration and later show that the LHS cannot increase in future iterations.

Let $\lambda^{\text{old}}(e)$ and $y^{\text{old}}(e)$ denote the values of $\lambda(e)$ and $y(e)$ at the start of the iteration and $\lambda^{\text{new}}(e)$ and $y^{\text{new}}(e)$ denote their values at the end. We have that $\sum_{e \in P(\ell)} \lambda^{\text{old}}(e)y^{\text{old}}(e) < c(\ell)$. We now show that $\sum_{e \in P(\ell)} \lambda^{\text{new}}(e)y^{\text{new}}(e) \leq 3c(\ell)$. Let e_i be the request of the current iteration. During this iteration, we only increase $y(e)$ for $e = e_i$ and we set $\lambda(e_i) = 1$ so $\lambda^{\text{new}}(e_i)y^{\text{new}}(e_i) = y(e_i)$. So, we have

$$\sum_{e \in P(\ell)} \lambda^{\text{new}}(e)y^{\text{new}}(e) = \sum_{e \in P(\ell)\setminus\{e_i\}} \lambda^{\text{new}}(e)y^{\text{old}}(e) + y(e_i).$$

Since y is a feasible dual, we have that $y(e_i) \leq c(\ell)$. Now, Proposition 16 implies that $\lambda^{\text{old}}(e) \geq 1$ if $y^{\text{old}}(e) > 0$. Together with the fact that $\lambda^{\text{new}}(e) \leq \lambda^{\text{old}}(e) + 1$, we get that $\lambda^{\text{new}}(e)y^{\text{old}}(e) \leq 2\lambda^{\text{old}}(e)y^{\text{old}}(e)$ and so

$$\sum_{e \in P(\ell)\setminus\{e_i\}} \lambda^{\text{new}}(e)y^{\text{new}}(e) \leq 2 \sum_{e \in P(\ell)\setminus\{e_i\}} \lambda^{\text{old}}(e)y^{\text{old}}(e) < 2c(\ell).$$

Thus, $\sum_{e \in P(\ell)} \lambda^{\text{new}}(e)y^{\text{new}}(e) \leq 3c(\ell)$ at the end of the current iteration.

Finally, we show that $\sum_{e \in P(\ell)} \lambda(e) y(e)$ does not increase in future iterations. At the end of the current iteration, ℓ is a candidate to be added to F. Among all candidates, the one with highest class is added, so either ℓ is added to F or a rooted link ℓ' of higher class is added to F. In the second case, by Proposition 16, we have $P(\ell') \supseteq P(\ell)$. Thus, in either case, we have that $Z \supseteq P(\ell)$ at the end of the current iteration. Moreover, in future iterations, we still have $Z \supseteq P(\ell)$ by Proposition 16. Therefore, $\sum_{e \in P(\ell)} \lambda(e) y(e)$ does not increase in future iterations. Thus, we conclude that $\sum_{e \in P(\ell)} \lambda(e) y(e) \leq 3c(\ell)$ at the end of the algorithm. ◀

Therefore, we conclude that Algorithm 1 is nice. Together with Lemma 11, we get Theorem 1.

References

1. Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. The Online Set Cover Problem. *SIAM J. Comput.*, 39(2):361–370, 2009. doi:10.1137/060661946.
2. Baruch Awerbuch, Yossi Azar, and Yair Bartal. On-line generalized Steiner problem. *Theoretical Computer Science*, 324:313–324, 2004.
3. Piotr Berman and Chris Coulston. On-Line Algorithms for Steiner Tree Problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 344–353, 1997.
4. Sina Dehghani, Soheil Ehsani, MohammadTaghi Hajiaghayi, Vahid Liaghat, and Saeed Seddighin. Greedy Algorithms for Online Survivable Network Design. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 152:1–152:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.152.
5. Guy Even and Shakhar Smorodinsky. Hitting sets online and unique-max coloring. *Discrete Applied Mathematics*, 178:71–82, 2014. doi:10.1016/j.dam.2014.06.019.
6. Greg N. Frederickson and Joseph JáJá. Approximation Algorithms for Several Graph Augmentation Problems. *SIAM J. Comput.*, 10(2):270–283, 1981. doi:10.1137/0210019.
7. Fabrizio Grandoni, Christos Kalaitzis, and Rico Zenklusen. Improved approximation for tree augmentation: saving by rewiring. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 632–645, 2018. doi:10.1145/3188745.3188898.
8. Anupam Gupta, Ravishankar Krishnaswamy, and R. Ravi. Online and Stochastic Survivable Network Design. *SIAM J. Comput.*, 41(6):1649–1672, 2012. doi:10.1137/09076725X.
9. MohammadTaghi Hajiaghayi, Vahid Liaghat, and Debmalya Panigrahi. Online Node-weighted Steiner Forest and Extensions via Disk Paintings. In *Proceedings of the 54th Annual Symposium on Foundations of Computer Science*, pages 558–567, 2013.
10. MohammadTaghi Hajiaghayi, Vahid Liaghat, and Debmalya Panigrahi. Near-Optimal Online Algorithms for Prize-Collecting Steiner Problems. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming, 41st International Colloquium, ICALP 2014*, volume 8572 of *Lecture Notes in Computer Science*, pages 576–587. Springer, 2014.
11. Makoto Imase and Bernard M. Waxman. Dynamic Steiner Tree Problem. *SIAM Journal on Discrete Mathematics*, 4:369–384, 1991.
12. Adam Meyerson. The Parking Permit Problem. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 274–282, 2005.
13. Joseph Naor, Debmalya Panigrahi, and Mohit Singh. Online Node-Weighted Steiner Tree and Related Problems. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 210–219, 2011. doi:10.1109/FOCS.2011.65.

14 Jiawei Qian, Seeun William Umboh, and David P. Williamson. Online Constrained Forest and Prize-Collecting Network Design. *Algorithmica*, 80(11):3335–3364, 2018.

15 Daniel Dominic Sleator and Robert Endre Tarjan. A Data Structure for Dynamic Trees. *J. Comput. Syst. Sci.*, 26(3):362–391, 1983. doi:10.1016/0022-0000(83)90006-5.

16 Seeun Umboh. Online Network Design Algorithms via Hierarchical Decompositions. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1373–1387, 2015.

Optimal Short Cycle Decomposition in Almost Linear Time

Merav Parter
Weizmann IS, Rehovot, Israel
http://www.weizmann.ac.il/math/parter/home

Eylon Yogev
Technion, Haifa, Israel
https://www.eylonyogev.com/about

Abstract

Short cycle decomposition is an edge partitioning of an unweighted graph into edge-disjoint short cycles, plus a small number of extra edges not in any cycle. This notion was introduced by Chu et al. [FOCS'18] as a fundamental tool for graph sparsification and sketching. Clearly, it is most desirable to have a *fast* algorithm for partitioning the edges into as *short* as possible cycles, while omitting *few* edges.

The most naïve procedure for such decomposition runs in time $O(m \cdot n)$ and partitions the edges into $O(\log n)$-length edge-disjoint cycles plus at most $2n$ edges. Chu et al. improved the running time considerably to $m^{1+o(1)}$, while increasing both the length of the cycles and the number of omitted edges by a factor of $n^{o(1)}$. Even more recently, Liu-Sachdeva-Yu [SODA'19] showed that for every constant $\delta \in (0, 1]$ there is an $O(m \cdot n^{\delta})$-time algorithm that provides, w.h.p., cycles of length $O(\log n)^{1/\delta}$ and $O(n)$ extra edges.

In this paper, we significantly improve upon these bounds. We first show an $m^{1+o(1)}$-time *deterministic* algorithm for computing nearly optimal cycle decomposition, i.e., with cycle length $O(\log^2 n)$ and an extra subset of $O(n \log n)$ edges not in any cycle. This algorithm is based on a reduction to *low-congestion cycle covers*, introduced by the authors in [SODA'19].

We also provide a simple deterministic algorithm that computes edge-disjoint cycles of length $2^{1/\epsilon}$ with $n^{1+\epsilon} \cdot 2^{1/\epsilon}$ extra edges, for every $\epsilon \in (0, 1]$. Combining this with Liu-Sachdeva-Yu [SODA'19] gives a *linear* time randomized algorithm for computing cycles of length $\mathsf{poly}(\log n)$ and $O(n)$ extra edges, for every n-vertex graphs with $n^{1+1/\delta}$ edges for some constant δ.

These decomposition algorithms lead to improvements in all the algorithmic applications of Chu et al. as well as to new distributed constructions.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Cycle decomposition, low-congestion cycle cover, graph sparsification

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.89

Category Track A: Algorithms, Complexity and Games

Funding *Merav Parter*: The Israel Science Foundation grant no. 2084/18
Eylon Yogev: The European Union's Horizon 2020 research and innovation program under grant agreement No. 742754.

1 Introduction

Short cycle decomposition introduced by Chu et al. [3] is a partitioning of the graph edges into edge-disjoint short cycles and a small subset of extra edges that are not in any cycle.

▶ **Definition 1** (Short Cycle Decomposition). *An $(\widehat{m}, L)$-short cycle decomposition of an unweighted undirected graph G is a collection of edge-disjoint cycles in G, each of length at most L, such that at most $\widehat{m}$ edges of G are not covered by these cycles.*

In their recent paper, Chu et al. [3] demonstrated the power of short cycle decomposition as a fundamental tool for a number of problems in graph sparsification. This includes the construction of degree preserving sparsifiers, resistance sparsifiers, graphical spectral sketches, approximation of the Laplacian's determinant and more. Chu et al. [3] showed that the efficiency of this long list of problems is determined by the time complexity, the cycle length, and the number of uncovered edges of the short cycle decomposition routine in hand. Clearly, it is most desirable to compute fast a decomposition into the shortest possible edge-disjoint cycles, while omitting as few as possible edges.

As they have observed, there is a naïve short cycle decomposition which runs in time $O(mn)$ (where n is the number of vertices in the graph, and m is the number of edges), and outputs an optimal[1] decomposition: cycle length of $O(\log n)$ and $O(n)$ extra edges. In their key algorithmic result, [3] significantly improved this time complexity by presenting an almost-linear time algorithm[2] which decomposes G into edge-disjoint cycles of length $n^{o(1)}$, and an extra number of $n^{1+o(1)}$ edges. Thus, the improvement in the time complexity came with the cost of increasing both the cycle length as well as the number of left-over edges by a multiplicative factor of $n^{o(1)}$. Improving the efficiency of the short cycle decomposition was stated in [3] as a highly motivated target, as it immediately leads to a large sequence of algorithmic consequences:

> *"Critically, any improvement to our short-cycle decomposition algorithm will achieve an improvement in all of our results."*

Very recently, Liu, Sachdeva and Yu [8] provided the first improvement for the problem, by presenting an $O(m \cdot n^{\delta})$-time algorithm that decomposes G into edge-disjoint cycles of length $O(\log n)^{1/\delta}$ and an extra subset of $O(n)$ edges, for any *constant* $\delta \in (0, 1]$. This simplifies and improves the decomposition algorithm of [3] in terms of all parameters, but still leaves the following fundamental question open:

Is there an ***optimal*** *cycle decompositios in almost* ***linear*** *time?*

In this paper, we answer this question in the affirmative and present an $m^{1+o(1)}$-time algorithm for decomposing the graph into edge-disjoint cycles of length $O(\log^2 n)$, and an extra number of $O(n \log n)$ edges.

▶ **Theorem 2** (Nearly Optimal Decomposition in Almost-Linear Time). *There is an almost-linear time algorithm for computing a cycle decomposition with cycle length of $O(\log^2 n)$, and $O(n \log n)$ extra edges.*

We also have a much simpler algorithm that achieves the same quality of cycle decomposition[3] as by [3], only in $\widetilde{O}(m)$ time. An additional benefit of this algorithm is that it is deterministic.

▶ **Theorem 3** (Longer Cycles in Linear Time). *For every n-vertex graph $G = (V, E)$ with m edges, one can compute in $\widetilde{O}(m)$ time, a decomposition with cycle length $n^{o(1)}$ and $n^{1+o(1)}$ extra edges.*

---

[1] Optimality in this context is up to poly-logarithmic terms.

[2] A graph algorithm is almost-linear if runs in time $m^{1+o(1)}$.

[3] In fact, the quality here is slightly better, as the $n^{o(1)}$ factor (in the cycle length and number of uncovered edges) is $2^{\sqrt{\log n}}$ and in [3] et al. it is $2^{(\log n)^{3/4}}$.

The latter provides an improvement for approximating the determinant of the Laplacian. We can also combine with the algorithm of [8] to obtain a randomized decomposition that is optimal (up to log-n factors) in all three complexity measures: time, length and number of leftover edges provided that the graph is sufficiently dense.

▶ **Theorem 4** (Optimal Decomposition for Dense Graphs). *For every constant $\delta \in (0,1]$, there exists a randomized algorithm that computes in time $\widetilde{O}(m) + n^{1+1.1\delta}$ a collection of edges disjoint cycles of length $O(\log n)^{1/\delta}$ and at most $O(n)$ leftover edges.*

Table 1 provides a summary of our results in comparison to [3, 8]. Our algorithms are based on independent ideas compared to [3, 8], which are related to the concept of *low-congestion cycle covers*.

Table 1 Summary of our results.

	Cycle Length	#Uncovered Edges	Time	Type
Chu et al. [3]	$n^{o(1)}$	$n^{1+o(1)}$	$m^{1+o(1)}$	Randomized
Liu-Sachdeva-Yu [8]	$O(\log n)^{1/\delta-1}$ for constant $\delta \leq 1$	$O(n)$	$O(mn^{\delta})$	Randomized
This Work	$O(\log^2 n)$	$O(n \log n)$	$m^{1+o(1)}$	Deterministic
This Work	$O(\log n)^{1/\delta}$ for constant $\delta \leq 1$	$O(n)$	$m + n^{1+1.1\delta}$	Randomized
This Work	$n^{o(1)}$	$n^{1+o(1)}$	$m + n^{1+o(1)}$	Deterministic

1.1 Low-Congestion Cycle Covers

A cycle cover of a graph G is a collection of cycles such that each edge of G appears in at least one of the cycles. Cycle covers were introduced by Itai and Rodeh [6] in 1978 with the objective to cover all edges of a bridgeless[4] graph with cycles of minimum *total* length. Motivated by applications to distributed computation, [11] recently introduced[5] the notion of *low-congestion* cycle covers: a collection of cycles that are both *short*, nearly *edge-disjoint* and covering all edges.

▶ **Definition 5** (Low-Congestion Cycle Cover). *A $(\mathtt{d}, \mathtt{c})$-cycle cover of a graph G is a cycle collection that covers all edges in G. Each cycle has length at most $\mathtt{d}$, and each edge participates in at least one cycle and at most $\mathtt{c}$ cycles.*

Low-congestion cycle covers provide the basic communication backbone in different settings of resilient distributed computation such as Byzantine fault model and secure computation [11, 10]. Whereas a-priori it is not clear that cycles of short length and small overlap exist, our main result in [11] shows that one can enjoy a dilation of $O(D \log n)$ while incurring only a poly-logarithmic congestion, where D is the diameter of the graph.

Comparison to Short Cycle Decomposition. Low-congestion covers bare some similarity to short cycle decomposition but differs from it in two main aspects. The first aspect follows from the definition: Low-congestion covers allow a small *overlap* between the cycles,

[4] A graph G is bridgeless, if any single edge removal keeps the graph connected.
[5] In an independent manner to the notion of short cycle decomposition.

but require covering *all* edges. On the other hand, short cycle decomposition insists on edge-disjoint cycles, i.e., with no-overlap, but allows omitting a subset of leftover edges that are not in any cycle. The second difference concerns the algorithmic focus. In low-congestion covers, the main goal was in showing that optimal covers which are both short and with small overlap *exist*. Computation time was not the primary concern, and in fact, the first step in the construction in [11] used the naïve decomposition algorithm (that runs in time $O(m \cdot n)$) to reduce the number of uncovered edges into $2n$. In contrast, for short cycle decomposition, it is easy to obtain the optimal decomposition in $O(m \cdot n)$ time, and hence the primary algorithmic focus is computation time.

1.2 Improved Graph Sparsification Algorithms via Short Cycles

Spectral sparsifiers introduced by Spielman and Teng [13] are sparse (weighted) subgraphs that approximately preserve the Laplacian quadratic form of the graph. Recall that the *Laplacian*, $\mathbf{L}_G$, of an undirected weighted graph $G = (V, E_G, w_G)$ is the unique symmetric $V \times V$ matrix such that for all $x \in \mathbb{R}^{|V|}$, it holds that

$$x^T \mathbf{L}_G x = \sum_{(u,v)\in E_G} w_G(u,v) \cdot (x_u - x_v)^2 \ .$$

For $\epsilon < 1$, a graph $H = (V, E_H, w_H)$ is an *ϵ-sparsifier* for G if

$$\forall x \in \mathbb{R}^n,\ x^T \mathbf{L}_G x \in (1 \pm \epsilon) x^T \mathbf{L}_H x \ .$$

Batson, Spielman and Srivastava [2] presented a construction of spectral sparsifiers with $O(n/\epsilon^2)$ edges, which is tight. In the last years, related graph structures have been defined, which are weaker than spectral sparsifiers, and thus potentially sparser. The recent work of Chu et al. [3] used short cycle decomposition to derive new existential results on the sparsity of sparsifiers and spectral sketches. As the time complexity and the quality of their algorithms depend on the efficiency of the decomposition, our decomposition algorithm leads to immediate improvements for all the algorithmic results from [3].

Graphic ϵ-Spectral Sketch and Resistance Sparsifiers. A spectral sketch [1] is a data structure for a graph G that given a query vector $x \in \mathbb{R}^n$ returns w.h.p. a $(1+\epsilon)$ approximation for the quadratic form $x^T \mathbf{L}_G x$. A data structure that works w.h.p. for *all* $x \in \{\pm 1\}^n$ requires n/ϵ^2 space. However, Jambulapati and Sidford [7] showed that when requiring the high probability guarantee for a *fixed* unknown vector, the size of the data structure can be made $\widetilde{O}(n/\epsilon)$. In the same manner, one can define the *graphic spectral sketch* of G to be a sparse graph H satisfying $x^T \mathbf{L}_G x \in (1 \pm \epsilon) x^T \mathbf{L}_H x$ for a fixed unknown vector x with high probability. Chu et al. showed that graphical spectral sketches with $\widetilde{O}(n/\epsilon)$ edges exist.

Resistance sparsifiers are sparse subgraphs that preserve the effective resistance[6] of all vertex pairs up to a multiplicative factor of $(1 + \epsilon)$. This notion was introduced by Dinitz-Krauthgamer-Wagner [4] who conjectured that resistance sparsifiers with $\widetilde{O}(n/\epsilon)$ edges always exist. The conjecture was indeed resolved by [3] using the tool of short cycle decomposition. By combining Theorem 2 with [3, Theorem 6.1] we get:

[6] The effective resistance between a pair u, v is the difference in the voltage between u, v when the graph is an electrical network, with every edge e of weight w_e has a resistor of resistance $1/w_e$ and 1 unit of current is sent from u to v.

▶ **Theorem 6** (Graphic Spectral Sketches and Resistance Sparsifiers). *One can compute an ϵ-resistance sparsifier H and a graphical spectral sketch H' with $\widetilde{O}(n/\epsilon)$ edges in time $m^{1+o(1)}$. Alternatively, these algorithms can be tuned to run in $\widetilde{O}(m)$ time while producing such graphs H, H' with $n^{1+o(1)}/\epsilon$ edges.*

These two results should be compared with (i) the $O(m \cdot n^{\Theta(1)})$-time algorithm of [8] that gives sparsifiers with $\widetilde{O}(n/\epsilon)$ edges; and (ii) the $m^{1+o(1)}$-time algorithms of [3, 8] that give sparsifiers with $n^{1+o(1)}/\epsilon$ edges. Hence, we provide the first almost-linear time algorithm for optimal size sparsifiers with $\widetilde{O}(n/\epsilon)$ edges, and the first near linear time algorithm[7] for almost-linear size sparsifiers with $\widetilde{O}(n^{1+o(1)}/\epsilon)$ edges.

Degree Preserving Sparsifiers and Sparsifiers for Eulerian Directed Graphs. Short cycle decompositions are useful for providing spectral sparsifiers that preserve additional key properties in the original graphs. *Degree preserving sparsifier* is a spectral sparsifier H for a graph G which preserves (exactly) the *weighted degree* of each vertex $v \in V$. To get an intuition for the usefulness of edge-disjoint cycles in this context, imagine G to be an unweighted union of edge-disjoint cycles of even length. A degree preserving sparsifier H that contains half of the edges in G, can be obtained by the following correlated sampling: For each cycle, with probability $1/2$ add to H the odd edges with weight 2, and with probability $1/2$ add to H the even edges with weight 2. It is easy to see that every vertex v has exactly the same weighted degree in H as in G, and the number of edges in G was cut by half. By combining Theorem 3.3 of Chu et al. [3] with Theorem 2, we get:

▶ **Theorem 7** (Optimal Degree Preserving Sparsifiers in Almost Linear Time). *There exists an algorithm that runs in time $m^{1+o(1)}$ and constructs a degree-preserving ϵ-sparsifier of G with $\widetilde{O}(n/\epsilon^2)$ edges, with high probability. Alternatively, an $n^{1+o(1)}/\epsilon^2$-size degree-preserving sparsifier can be computed in $\widetilde{O}(m)$ time.*

A similar approach has been taken in Chu et al. [3] to construct a sparsification of Eulerian directed graphs. By plugging Theorem 2 in Theorem 5.1 of [3], we get:

▶ **Theorem 8** (Sparsification of Eulerian Directed Graphs). *There exists an algorithm $\mathcal{A}$ that given an Eulerian directed graph $\overrightarrow{G}$ with polynomial bounded edge weights returns an Eulerian directed graph $\overrightarrow{H}$ such that either: (i) $\overrightarrow{H}$ has $\widetilde{O}(n/\epsilon^2)$ edges and $\mathcal{A}$ has time complexity $m^{1+o(1)}$, or (ii) $\overrightarrow{H}$ has $n^{1+o(1)}/\epsilon^2$ edges and $\mathcal{A}$ has time complexity $\widetilde{O}(m)$.*

Estimation of the Effective Resistance and the Determinant of the Laplacian. Finally, we show how incorporating our improved construction of resistance sparsifiers can yield a faster approximation of the determinant of the graph Laplacian with the last row and column removed. In particular, this yields the first linear time algorithm for sufficiently dense graphs. Recall that by Theorem 6, given a graph G with m edges, we can compute in time $\widetilde{O}(m)$, a resistance sparsifier H with $n^{1+o(1)}/\epsilon$ edges, with high probability. By applying [3, Thm. 3.8] on H, we get:

▶ **Theorem 9** (Faster Approximation of Effective Resistance). *Given an undirected graph G with m edges, one can compute with high probability an ϵ-approximations to the effective resistances between a given set of t vertex pairs in time $\widetilde{O}(m) + (n+t)n^{o(1)}\epsilon^{-1.5}$.*

[7] A graph algorithm is *near linear* if it runs in $O(m \cdot \mathsf{poly}(\log n))$ time.

Hence, for $t = o(m^{1-o(1)} \cdot \epsilon^{1.5})$, we obtain a linear time algorithm for approximating the effective resistance. As observed in [3], the running time bottleneck of the determinant estimation algorithm for Laplacians by Durfee et al. [5] is the estimation of the effective resistance of $O(n^{1.5})$ pairs with an error of $\epsilon = n^{-0.25}$. Plugging our improved Theorem 9 in Lemma B.1 of [3] yields:

▶ **Corollary 10** (Faster Approx. of Laplacian's Determinant). *Given a graph Laplacian* $\mathbf{L}$ *and any error* $0 < \epsilon < 1/2$*, one can compute an* $1 \pm \epsilon$ *estimate to* $\det(\mathbf{L}_{-n})$[8] *in time* $\widetilde{O}(m) + n^{15/8+o(1)} \cdot \epsilon^{-7/4}$*, thus in linear time for sufficiently dense graphs.*

1.3 Distributed Implementation

Our centralized construction has the benefit of naturally being implemented in the standard CONGEST model of distributed computing. As in the centralized setting, the decomposition is based on constructing low-congestion cycle covers. We show:

▶ **Lemma 11** (Distributed Low-Congestion Covers). *There exists a distributed algorithm that given n-vertex graph* $G = (V, E)$ *constructs a* $(\mathtt{d}, \mathtt{c})$ *cycle cover within* $O(\mathtt{d} \cdot \mathtt{c})$ *rounds for* $\mathtt{d}, \mathtt{c} = 2^{O(\sqrt{\log n})}$.

The proof appears in the full version of the paper. This improves upon the linear time algorithm of [11, 10]. By combining this algorithm with Luby-MIS algorithm [9], we get:

▶ **Theorem 12** (Distributed Short Cycle Decomposition). *There exists an* $2^{O(\sqrt{\log n})}$*-round distributed algorithm that given n-vertex graph* $G = (V, E)$ *decomposes* G *into edge-disjoint cycles of length* $2^{O(\sqrt{\log n})}$ *plus* $O(n \log n)$ *extra edges.*

One might hope that using this distributed construction of cycle decomposition we would get a distributed algorithm for all the above application. Unfortunately, in the distributed setting, to this point, we do not have an efficient algorithm that approximates (even up to a constant factor) the effective resistance of all edges[9] in G. This is the only missing piece for obtaining the above mentioned algorithmic applications of the short cycle decomposition in a distributed setting.

Comparison to the work of Chu et al. [3] and Liu-Sachdeva-Yu [8]. We first observe that in [8], the number of leftover edges is $O(n)$, whereas in our case it is $O(n \log n)$. By applying the algorithm of [8] on these last $O(n \log n)$ edges, for any constant $\delta \in (0, 1)$, we can compute an $O(n, (\log n)^{1/\delta-1})$-decomposition in time $O(n^{1+\delta} \log n + m^{1+1/\log\log n})$, which considerably improves upon the time complexity of $O(m \cdot n^{\delta})$ of [8]. Since we consider log-n factors to be negligible in this work (i.e., the size of the sparsifiers is $\Omega(n \log n)$ in any case), we omit this step.

Fixing the number of leftover edges to $\widetilde{O}(n)$, then [8] computes cycles of length $O(\log n)^{1/\delta-1}$ in time $2^{O(1/\delta)} \cdot n^{\delta} \cdot m$. In comparison, our algorithm computes cycles of length $2^{1/\delta} \cdot O(\log n)$ in roughly the same time $2^{O(1/\delta)} \cdot n^{\delta} \cdot m$. For example, when taking $\delta = 1/\log\log n$, both algorithms have roughly the same time complexity, but our algorithm produces cycles of length $O(\mathsf{poly}\log n)$ and their algorithm has cycle length $O(\log n)^{\log\log n}$.

From an algorithmic point of view, both our algorithm and the algorithm of [8] use low-diameter decomposition[10]. This allows one to restrict attention to $O(\log n)$-diameter graphs.

---

[8] the determinant of $\mathbf{L}$ with the last row and column removed

[9] In the output of such an algorithm we want each edge (u, v) to obtain a constant approximation for the effective resistance between u and v in G.

[10] We use a neighborhood covers which are close variant of low-diameter decomposition.

The approach of [8] contracts each of the components of the low-diameter decomposition and recursively computes vertex-disjoint short cycles on the contracted graph. The diameter of each super-node is $O(\log n)$, when lifting the contracted cycles back to edges in G the length of the cycles increases exponentially with the number of recursive layers. In particular, halving within $1/\delta$ recursive calls the length of the cycles becomes $O(\log n)^{1/\delta-1}$. Our approach is quite different. We also decompose trees into smaller components, but instead of contracting these components we use their internal edges carefully in our cycles. By enjoying the internal edges inside each cycle, the length of the cycles increases by a factor of at most 2 in each level of the recursion, thus after $1/\delta$ recursion levels, the length of the cycle is $2^{1/\delta}$.

2 Longer Edge-Disjoint Cycles in Nearly Linear Time

We begin by describing a deterministic algorithm for computing cycle decomposition of the same quality as that of Chu et al. [3], but in nearly linear time $\widetilde{O}(m)$. In particular, the cycle length will be bounded by $2^{\sqrt{\log n}}$ and at most $2^{\sqrt{\log n}} \cdot n$ edges will be left uncovered. Note that the recent randomized algorithm of [8] achieves such cycles in almost linear time $m \cdot n^{O(\log\log n/\sqrt{\log n})} \cdot 2^{\sqrt{\log n}/\log\log n}$. We can also reduce the number of leftover edges to $O(n)$, by running the algorithm of [8] on the remaining subset of $n^{1+o(1)}$ leftover edges. However, note that in any case, the efficiency of the algorithmic applications for these cycles depends on $\widehat{m} + nL$ where $\widehat{m}$ is the number of leftover edges and L is the largest cycle length.

▶ **Theorem 13.** *For every $\epsilon \in (0,1]$, there exists an $\widetilde{O}(m)$-time algorithm that computes an $(\widehat{m}, L)$ short cycle decomposition with $\widehat{m} = 1/\epsilon \cdot 2^{1/\epsilon} \cdot n^{1+O(\epsilon)}$ and $L = 2^{O(1/\epsilon)}$. Setting $\epsilon = 1/\sqrt{\log n}$, gives $\widehat{m} = 2^{O(\sqrt{\log n})} \cdot n$ and $L = 2^{O(\sqrt{\log n})}$.*

Thm. 4 follows immediately: Set $\epsilon = 1/\log\log n$ in Theorem 13, yeilding cycles of length $O(\log n)$ and $n^{1+o(1)}$ leftover edges. Then, using the randomized algorithm of [8] on this remaining subgraph with some constant $\delta \in (0,1]$, covers the remaining edges with cycles of length $O(\log n)^{1/\delta}$ with time complexity of $n^{1+1.1\delta}$. In addition, Thm. 3 follows by plugging $\epsilon = 1/\sqrt{\log n}$ in Theorem 13.

Throughout, a *block* is a subset of vertices with a bounded size. The algorithm is recursive and has $\ell = \lceil 1/\epsilon \rceil$ levels of recursion. During each recursion level, some virtual edges $\widetilde{E}$ will be added to the set of edges E' that we wish to cover by cycles (initially $E' = E$). Informally speaking, whenever the algorithm adds a virtual edge between two nodes u and v, it implies that the algorithm has already computed a u-v walk denoted by $W((u,v))$, and the virtual edge (u,v) indicates the need for computing another u-v walk so that we will end up with a cycle. In other words, adding a virtual edge means that we defer the closure of the cycle to future iterations. We also maintain a leftover subgraph H, and in certain cases, we give up on completing the cycles of the virtual edges, and add their walk edges to this subgraph.

We now describe Alg. FasterLongerCycles. The algorithm is recursive and has $O(1/\epsilon)$ levels of recursion. Initially, let $E' = E(G)$. The preliminary walk collection is $\mathcal{W} = \{e \mid e \in G\}$, the cycle collection $\mathcal{C}'$ is empty. In addition, we have a subgraph $H \leftarrow \emptyset$ that will contain the edges that are not covered by cycles.

In each independent level $i \geq 1$ of the recursion, we are given a block B, and a subset of edges E' with both endpoints in B. In addition, we are given a set of current walks $\mathcal{W}$ for the edges in E', a current cycle collection $\mathcal{C}'$, and a leftover subgraph H.

Block Partitioning. First the algorithm partitions B into $k = n^{\epsilon}$ balanced blocks $B_1, \ldots, B_k$ each with $\Theta(|B|/n^{\epsilon})$ vertices.

Taking care of edges between blocks. Our goal is to replace edges between blocks, to edges inside blocks. For block B_a, we do as follows for every $v \in B_a$ and every $b \in \{a+1, \ldots, k\}$. Define by $N_{a,b}(v) = \{u \in B_b \mid (u,v) \in E'\}$ to be the E'-neighbors of v in B_b. If $N_{a,b}(v)$ is odd, we will omit from it at most one vertex u, in order to make it even. The edge (u,v) of the omitted vertex u is omitted from E', and its walk $W((u,v))$ is added to the leftover subgraph H.

From now on, we can assume that the set $N_{a,b}(v)$ is even. We then (arbitrarily) match the vertices in $N_{a,b}(v)$ into pairs $\langle x,y \rangle$. Each matched pair $\langle x,y \rangle$ is handled as follows:

Case (1): The set E' already contains an (x,y) edge. If E' already contains an edge (x,y) (this edge might be virtual), we define a cycle $C = W((v,x)) \circ W((x,y)) \circ W((y,v))$ and add it to the cycle collection $\mathcal{C}'$. In addition, we omit the edges $(v,x),(x,y)$ and (y,v) from E', and omit their walks from $\mathcal{W}$.

Case (2): The set E' does not contain an (x,y) edge. In this case, we add a virtual edge (x,y) to E', as well as a walk $W((x,y)) = W((x,v)) \circ W((v,y))$ to $\mathcal{W}$. This completes the description of the i^{th} recursion level. The algorithm then recurses on each of the blocks $B_1, \ldots, B_k$. See Fig. 1 for pseudocode. Finally, as in Alg. PartialCycleCover, the cycles of $\mathcal{C}'$ might re-visit the same vertex, and hence in the final cleanup phase, the algorithm traverses each of the cycles in $\mathcal{C}'$ and simplify them.

Algorithm FasterLongerCycles$(B, E', \mathcal{W}, \mathcal{C}', H)$**.**
Level i of the Recursion (for non-singleton block B):

1. Decompose B into $k = n^{\epsilon}$ blocks $B_1, \ldots, B_k$ each with $|B|/k$ vertices.
2. For every block B_a and every vertex $v \in B_a$, do the following for every $b > a$:
   - **a.** Let $N_{a,b}(v) = \{u \in B_b \mid (u,v) \in E'\}$
   - **b.** If $|N_{a,b}(v)|$ is odd:
     - Omit an arbitrary u from $N_{a,b}(v)$.
     - Omit the walk $W((u,v))$ from $\mathcal{W}$ and the edge (u,v) from E'.
     - Add $W((u,v))$ to H.
   - **c.** Match the vertices in $N_{a,b}(v)$ into pairs $\langle x,y \rangle$ (in an arbitrary manner).
   - **d.** For each matched pair $\langle x,y \rangle$ do:
     - If $(x,y) \in E'$:
       - Add the cycle $W((v,x)) \circ W((x,y)) \circ W((y,v))$ to $\mathcal{C}'$.
       - Remove the edges $(v,x),(x,y),(y,v)$ from E', and their walks from $\mathcal{W}$.
     - Otherwise:
       - Add a virtual edge (x,y) to E', and add to $\mathcal{W}$ the x-y walk:

$$W((x,y)) = W((x,v)) \circ W((v,y)).$$

3. For every $a \in \{1, \ldots, k\}$ do:
   - Let E'_a be the edges in E' with both endpoints in B_a.
   - Let $\mathcal{W}_a = \{W(e) \in \mathcal{W} \mid e \in E'_a\}$.
   - Apply FasterLongerCycles$(B_a, E'_a, \mathcal{W}_a, \mathcal{C}', H)$.

Figure 1 Description of $n^{o(1)}$-length cycle decomposition in $\widetilde{O}(m)$ time.

Analysis. Let $E_i, \mathcal{W}_i$ be the union of the $E', \mathcal{W}$ sets over all the recursion calls in level i.

▷ Claim 14 (Cycle Length). (a) All walks added in level i have length $\leq 2^i$; (b) All cycles added in level i have length $\leq 2^{i+1}$.

Proof. Consider the first level for the base of the induction. Let $B_1, \ldots, B_k$ be the first level blocks. Fix a pair of blocks B_a, B_b for $a < b$, and $v \in B_a$. Let $\langle x, y \rangle$ be a matched pair in $N_{a,b}(v)$. First, assume that when considering $\langle x, y \rangle$, the current edge set E' does not contain (x, y). In such a case, we add an x-y walk $W((x, y)) = (x, v) \circ (v, x)$ of length 2 as required. Otherwise, E' already contains the edge (x, y) and by the explanation above, $|W((x, y))| \leq 2$. In such a case, we add a cycle $C = (v, x) \circ W((x, y)) \circ (y, v)$ which has length at most 4 as required.

Assume that the claim holds up to level $i-1$, and consider level i. Using the induction assumption, we can apply the same argument for the induction base and get that either: (1) we add an x-y walk $W((x, y)) = W((x, v)) \circ W((v, y))$. Note that by definition (x, v) and (y, v) are edges between blocks in level $i+1$. Thus if these edges are virtual, they must have been added in level $i-1$ (since all virtual edges added in level i connect vertices in the same $(i+1)$-level blocks). We have by induction assumption that $|W((x, v))|, |W((v, y))| \leq 2^{i-1}$. Thus $|(W((x, y))| \leq 2^i$. (2) Otherwise, if E' already contained the edge (x, y) when considering this matched pair, the algorithm adds a cycle $C = W((v, x)) \circ W((x, y)) \circ W((y, v))$. Note that the edge (x, y) could potentially be added in level-i (since it is inside an $(i+1)$-level block). Thus $|W((x, y))| \leq 2^i$ (by the previous case), and $|W((x, v))|, |W((v, y))| \leq 2^{i-1}$. Overall, $|C| \leq 2^{i+1}$. The claim follows. ◁

Since the algorithm has $\ell = O(1/\epsilon)$ levels, overall all cycles have length $2^{O(1/\epsilon)}$ as required.

Missing proofs appear in the full version of the paper.

▷ Claim 15. [Number of Uncovered Edges] $|E(H)| = 1/\epsilon \cdot 2^{1/\epsilon} \cdot n^{1+O(\epsilon)}$.

Proof. We bound the number of edges added to the leftover subgraph H due to a fixed vertex v. Since the blocks are vertex-disjoint at every recursion level, a vertex belongs to at most $O(1/\epsilon)$ blocks: at most one block, in each level in the recursion tree (once a vertex becomes a singleton block, we stop sub-dividing it). Consider level i, and let B_v be the unique block containing v. Recall that in this level, the input edge set E_i contains only edges whose both endpoints are in the same i-level block.

Now, the algorithm subdivides B_v into n^ϵ disjoint blocks: $B_1, \ldots, B_k$. W.l.o.g., let B_1 be the block containing the vertex v. For every other block B_j for $j \in \{2, \ldots, k\}$, we omit at most one edge $e_j \in E_i$ and add a walk $W(e_j)$ to H. By Claim 14, every walk $W(e_j)$ is of length at most $2^{O(1/\epsilon)}$. Therefore, there is a total of $k \cdot 2^{O(1/\epsilon)}$ edges on the walks $W(e_1), \ldots, W(e_k)$ that are added to H when considering v. Summing over all the vertices, and over all $O(1/\epsilon)$ recursion levels, we get that $|H| = 1/\epsilon \cdot 2^{1/\epsilon} \cdot n^{1+O(\epsilon)}$. ◁

Finally, we show that all the edges that are not in H are covered by the cycles in $\mathcal{C}$.

▷ Claim 16 (Cover). Every edge is either in H or covered by the cycles in $\mathcal{C}$.

Proof. We claim by induction on i, that every edge $e \in G \setminus H$, either has a walk $W(e')$ such that $e' \in E_i$ and $e \in W(e')$, or that e is covered by a cycle in $\mathcal{C}$. The base of the induction holds vacuously. Assume it holds for $i-1$ and consider level i. It remains to take care for edges $e \in G$ such that (i) there exists $e' \in E_{i-1}$ satisfying that $e \in W(e')$ and (ii) $e' \notin E_i$. To see this observe that if (i) does not hold, then the statement holds by induction assumption. If (i) holds but (ii) does not hold, then $e \in W(e')$ for an $e' \in E_i$ and the statement holds.

Consider then such an edge e that satisfies the above two conditions. Since e' was omitted from E_i in phase $i-1$, it implies that $e' = (u, v)$ was an edge between two i-level (brother) blocks B_a and B_b. W.l.o.g., $u \in B_a$ and $v \in B_b$. We first observe that if u has an odd

number of edges in E_i with a second endpoint of B_b, then the unique edges e'' omitted from consideration cannot be e'. This holds since when omitting e'', we add $W(e'')$ to H. Since $e \in W(e')$ but $e \notin H$, it must hold that $e' \neq e''$. From now on, we know that e' was matched to another u-edge (u, v'). In this case either an v-v' walk $W(\widehat{e})$ for $\widehat{e} = (v, v')$ is added to the walk collection, or that a cycle containing $W(e')$ is added to $\mathcal{C}$. In either case, since $e \in W(e')$, it is indeed covered by either the walks or the cycles in level-i. ◁

Setting $\epsilon = 1/\sqrt{\log n}$, yields cycles of length $2^{\sqrt{\log n}}$ and at most $|H| = 2^{O(\sqrt{\log n})} \cdot n$ edges. All other edges not in H are covered by a cycle.

Edge-Disjoint Cycles. We prove (1) every edge belongs to at most one walk in $\mathcal{W}_i$ for every i, and (2) cycles are made by gluing a disjoint set of walks. Claim (1) can be shown by induction on the set of walks $\mathcal{W}_i$. The base of the induction holds vacuously. Assume that it holds up to i and consider the walks added in level i. The walks are formed one by one, where walks of level i formed by gluing together walks in $\mathcal{W}_i$. Whenever a walk $W(e) = W(e') \circ W(e'')$ is formed, the walks $W(e'), W(e'')$ are omitted from the walk collection and would not be considered again. The claim follows by combining with the induction assumption. To see (2) observe that whenever we form a cycle, all its walks are omitted from the walk collection. The proof follows from the fact that all walks are edge-disjoint.

Time Complexity. We claim that each all recursion calls of level i can be implemented in $O(m)$ time, for every $i = 1, \ldots, \ell$. We claim that all operations are linear in m. We keep the block ID of each vertex v (the maximum vertex ID in its block) in each level $i \geq 1$. Then by traversing over the edges in E_i, we can compute the edges $E_{a,b}$ between each pair of bothering blocks B_a, B_b in level i. We traverse the edges in $E_{a,b}$ for each vertex v in B_a. All operations of gluing walks due to an addition of virtual edges are linear in the length of the walks. Since all walks are edge-disjoint, we touch each edge $e \in E$ at most $O(1)$ many times in each phase.

3 Shorter Cycles in Almost Linear Time

We next turn to consider the high-level idea of our main algorithm which computes a decomposition with almost optimal quality in almost-linear time. Thus establishing Theorem 2. Specifically, here the cycle length will be bounded by $O(\log^2 n)$, and we will omit at most $O(n \log n)$ edges. Note that the simple algorithm described above omits $2^{1/\epsilon} \cdot n^{1+\epsilon}$ edges which is at least $2^{\sqrt{\log n}} n$ for any value of ϵ.

One option to improve this bound is by setting $\epsilon = 1/\log\log n$ to get cycles of length $O(\log n)$, while omitting $n^{1+o(1)}$ edges. Then by applying the algorithm of [8] on the remaining edges with $\delta = 1/c$ for some constant c we get an algorithm for covering all but $O(n)$ edges with $\mathsf{polylog}(n)$ length cycles in total time $m + n^{1+\delta}$. Since δ is constant, this algorithm is not an almost linear algorithm for graphs with $m = n^{1+o(1)}$ edges. Therefore, in order to obtain a truely almost linear algorithm that that omits $\widetilde{O}(n)$ edges, runs in time $m^{1+o(1)}$ and produces cycles of length $\mathsf{polylog}(n)$, we must come up with some new ideas.

Our alternative algorithm is recursive and has $O(1/\epsilon)$ recursion levels. It is also based on a balanced partitioning into blocks, only that the blocks in this context are more involved. Instead of computing edge-disjoint cycles directly, we compute short cycles that have a small amount of overlap. This notion is captured by low-congestion cycle cover defined as follows.

Key Task:

- **Input:** Parameter $\epsilon \in (0,1)$, an n-vertex graph G of diameter $O(\log n)$ with m edges, and a BFS tree $T \subseteq G$.
- **Goal:** Cover *all* non-tree edges with cycles of length at most $\mathtt{d} = O(2^{1/\epsilon} \cdot \log n)$, such that each edge appears on at most $\mathtt{c} = 1/\epsilon \cdot n^{O(\epsilon)}$ cycles. That is, compute a $(\mathtt{d}, \mathtt{c})$ cycle cover for the non tree edges.

Figure 2 The key sub-problem for short cycle decomposition.

For a bridgeless graph $G = (V, E)$, a $(\mathtt{d}, \mathtt{c})$ cycle cover is a collection of cycles of length at most $\mathtt{d}$, such that each edge in G appears on at least one cycle and on at most $\mathtt{c}$ cycles[11]. Intuitively, if each edge appears on few cycles, then one can greedily pick a subset of edge-disjoint cycles which covers a large enough fraction of the edges, and then repeat again (after removing all edges that are currently covered). Moreover, we also show that computing the decomposition boils down into an even easier variant of the low-congestion cycle cover problem.

Note that the key task considers all graphs of diameter $O(\log n)$, whereas in our case the input graph G might have a large diameter. To add more insult to the injury, the low-congestion cover computation is computed repeatedly on the subset of yet uncovered edges (i.e., by the current subset of edge-disjoint cycles). Thus, even if the original input graph has a small diameter, already in its second application, the input graph to the algorithm might not be even connected. In the full version, we show how to settle down this mystery using the notion of neighborhood covers. From now on, we focus on the key task.

Solving the Key Task. Given a tree T and an $\epsilon \in (0,1]$, our goal now is to cover all non-tree edges $E \setminus T$ with cycles of length at most $\mathtt{d} = 2^{1/\epsilon} \cdot O(\log n)$, such that each edge appears on at most $\mathtt{c} = 1/\epsilon \cdot n^{\epsilon}$ cycles.

The algorithm is recursive with $\ell = O(1/\epsilon)$ recursion levels. In each independent level $i \in \{1, \ldots, \ell\}$ of the recursion, we are given a subtree T' and a collection of at most $m/n^{\epsilon \cdot (i-1)}$ edges E' with both endpoints in T' that should be covered. Some of the edges in E' (in level $i \geq 2$) might be virtual, and in such a case it implies that the algorithm has already computed a partial cycle (i.e., a walk) that covers them. We are also given a set of walks $\mathcal{W}$ that contains a walk $W(e)$ for each $e \in E'$. Initially, T' is simply T, E' contains all edges in $E(G) \setminus T$ that we want to cover, and $\mathcal{W} = \{e : e \in G\}$ contains the trivial walks for each edge.

Step (1): Balanced Block Partitioning. The first step of the algorithm is to partition T' into $k = \Theta(n^{\epsilon})$ edge-disjoint subtrees $T_1, \ldots, T_k$ that are *balanced* with respect to their degrees in E'. We call such balanced subtrees *blocks*. Any vertex whose degree in E' is too high defines a singleton block.

As in the previous algorithm, we will distinguish between two types of E'-edges: edges inside a block and edges between blocks. We next describe how to replace edges between blocks with virtual edges that are *inside a block*, in a way that covering the virtual edges by

[11] Our constructions do not require the graph to be bridgeless, it covers edges by cycles provided that such a cycle exists.

cycles will, later on, be translated back to a covering of the original inter-block edges. Unlike the previous algorithm, here we do not have the privilege to throw away edges to leftover subgraphs. Thus, we will have to make sure that all virtual edges are eventually completed into a cycle.

Step (2): Handling Edges Between Blocks. Let $E_{a,b}$ be all the edges in E' with one endpoint in T_a and the other in T_b. Our goal now is the following: we want to find a matching of the edges in $E_{a,b}$ in a way such that if $e = (x, y)$ and $e' = (x', y')$ where $x, x' \in T_a$ and $y, y' \in T_b$ are matched then there is a path $\pi(x, x')$ in T_a such that these paths for all pairs are edge disjoint. Then, we add the virtual edge $(y, y') \in T_b \times T_b$ and remove e and e'. Furthermore, we maintain the set of walks $\mathcal{W}$ connecting the endpoints of each virtual edge $\widehat{e} = (y, y')$ where $W(\widehat{e}) = e \circ \pi(x, x') \circ e'$.

Intuitively, the addition of a virtual edge (y, y') indicates to the algorithm that a cycle covering the edges e and e' is "under construction": there is currently an y-y' walk which will become a cycle when covering the virtual edge (y, y'). Importantly, all the virtual edges are internal to T_b and thus will be covered recursively by a path inside T_b. This path, along with $W(e)$ will complete the cycle. This procedure is applied for each pair of subtrees T_a, T_b separately, eventually converting all inter-block edges to internal block edges. Then, we apply the algorithm recursively on each block.

Notice that when the algorithm is applied recursively, then the inter-block edges e, e' might be virtual. Thus, we define the walks as follows. Initially, we set $W(e) = e$ for all edges of the graph and let $\mathcal{W} = \{W(e), e \in E'\}$. Then, we update $W(\widehat{e}) = W(e) \circ \pi(x, x') \circ W(e')$. That is, if e is a virtual edge then instead of adding it to the path we added its path $W(e)$ that contain "real" edges of the graph. This, of course, makes the walk longer and we bound their length later on.

We are left to describe how the matching is performed. As long as there is a vertex x in either T_a or T_b that is adjacent to at least two edges $E_{a,b}$, then we can match these two edges. That is, in this case, we have that $x = x'$ and thus the path $\pi(x, x') = x$ is the trivial path and is, of course, edge-disjoint from any other path. Thus, we can now assume that each vertex in T_a and T_b is adjacent to at most one edge in $E_{a,b}$. Note in the previous algorithm, when we got to a point that a vertex in a block is incident to one vertex in another block, we simply got rid of this edge by adding it to the leftover subgraph H. Here, we do not have this option, as we really need to cover *all* non-tree edges. This is exactly the point where congestion kicks in: covering all edges will come with the cost of producing cycles with some overlap, rather than edge-disjoint cycles as in the previous algorithm.

Let $M_{a,b} \subset T_a$ be all the vertices in T_i that are adjacent to an endpoint of an edge in $E_{a,b}$. If $|M_{a,b}|$ is odd, then we omit a single vertex y from this set, and cover the edge (x, y) by adding the "fundamental" cycle $C = W((x, y)) \circ \pi(x, y, T)$ to the cycle collection[12].

From now on, we assume that $M_{a,b}$ is even, and each $y \in M_{a,b}$ is incident to a unique edge in $E_{a,b}$ to be covered. The key tool used in this context is the following lemma:

▷ Fact 17. [12] Given a tree T and an even subset of marked vertices $M \subseteq V(T)$, one can compute a matching the vertices of M into pairs such that the collection of tree paths $\pi(x, y, T)$ over all the matched pairs are edge-disjoint.

We apply Algorithm DisjointMatching to the instance $T_a, M_{a,b}$. The output of this algorithm is a matching of the marked vertices $M_{a,b}$ into pairs $\langle x_i, y_i \rangle$, along with a collection of edge-disjoint paths in T_a connecting the matched pairs. The matched vertices naturally

[12] If e is an edge in G it is indeed a fundamental cycle.

define the matching between the remaining edges, along with edge-disjoint paths. This completes the high-level description of level i, while omitting some minor technical subtleties.

Why It Works. We give an overview of the analysis of this algorithm.

1. **Cycle lengths:** Let d_i be the maximum length of all walks at the beginning of level i of the recursion. In level i, the walks $W(\widehat{e}) = W(e) \circ \pi(x, x') \circ W(e')$ contain two $(i-1)$-level walks and a tree segment. Since the depth of the tree is $O(\log n)$, we get $d_{i+1} \leq 2d_i + O(\log n)$. Solving for $i = O(1/\epsilon)$, gives the desired cycle length of $2^{O(1/\epsilon)} \cdot \log n$.
2. **Congestion:** We first consider the congestion added when creating walks via the routing edge-disjoint matching algorithm. We claim that in every recursion level, the congestion on the tree edges is increased by (at most) an additive term of n^ϵ. The non-tree edges, in contrast, will belong to exactly one cycle (using a similar argument to the previous algorithm).
   The congestion argument works by induction as well. Assume by induction that every tree edge e appears at most c_{i-1} many times on all walks computed up to level i. In level i, every walk is of the form: $W(\widehat{e}) = W(e) \circ \pi(x, x') \circ W(e')$. That is, it has a tree segment $\pi(x, x')$ and two segments of an $(i-1)$-level walks. Since each $(i-1)$-level walk is added to at most one i-level walk (due to the matching step), the total congestion on the non-tree part of the i-level walks is kept the same. The increase in the congestion is then due to the tree segment $\pi(x, x')$. Recall that this tree segment is the outcome of applying the routing disjoint algorithm in some block T_a. For each application of this algorithm in T_a w.r.t to the edges of a fixed block T_b, the tree segments are disjoint. However, since the algorithm is applied in T_a for n^ϵ many times – per other block T_b, the total congestion on its tree edges is n^ϵ. Overall, we get that $c_i \leq c_{i-1} + n^\epsilon$. Solving for $i = O(1/\epsilon)$, gives the desired bound.
   Finally, we bound the congestion due to the fundamental cycles. Recall that whenever the number of edges between blocks T_a and T_b is odd, we cover a single edge with its fundamental cycle. Let T' be an i-level block and $T_1, \ldots, T_k$ be its children. The tree segment of the fundamental cycle of each edge between T_a and T_b is contained in T'. Since T' has $k = O(n^\epsilon)$ children, the tree edge appears on $n^{2\epsilon}$ cycles. Next, observe that since the blocks of each level are edge-disjoint, an edge appears on $O(1/\epsilon)$ many blocks over all, thus the total congestion added due to the fundamental cycles of the child components is $1/\epsilon n^{2\epsilon}$.
3. **Time Complexity:** In each phase, for each of the blocks T_a we have $k = O(n^\epsilon)$ applications of the routing disjoint matching algorithm. This algorithm can be implemented in linear time. Since the blocks are edge-disjoint, overall it takes $k \cdot \sum_a O(|T_a|) = m \cdot n^\epsilon$. Computing the walks defined by these matching outcome can be done in linear time in the total length of all i-level walks. Since each tree edge appears at most n^ϵ times on this walks (in each recursion level), the total length of the walks is $O(m + n^\epsilon \cdot n)$. Summing overall $O(1/\epsilon)$ recursion levels gives the desired bound. The pseudocode of the algorithm appears, illustrations, the complete analysis and the distributed implementation all appear in the full version of the paper.

References

1. Alexandr Andoni, Jiecao Chen, Robert Krauthgamer, Bo Qin, David P Woodruff, and Qin Zhang. On sketching quadratic forms. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 311–319. ACM, 2016.
2. Joshua Batson, Daniel A Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *SIAM Journal on Computing*, 41(6):1704–1721, 2012.
3. Timothy Chu, Yu Gao, Richard Peng, Sushant Sachdeva, Saurabh Sawlani, and Junxing Wang. Graph Sparsification, Spectral Sketches, and Faster Resistance Computation, via Short Cycle Decompositions. In *59th Annual Symposium on Foundations of Computer Science, FOCS*. IEEE Computer Society, 2018.
4. Michael Dinitz, Robert Krauthgamer, and Tal Wagner. Towards Resistance Sparsifiers. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24-26, 2015, Princeton, NJ, USA*, pages 738–755, 2015.
5. David Durfee, John Peebles, Richard Peng, and Anup B Rao. Determinant-preserving sparsification of SDDM matrices with applications to counting and sampling spanning trees. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 926–937. IEEE, 2017.
6. Alon Itai and Michael Rodeh. Covering a graph by circuits. In *International Colloquium on Automata, Languages, and Programming*, pages 289–299. Springer, 1978.
7. Arun Jambulapati and Aaron Sidford. Efficient n/ϵ Spectral Sketches for the Laplacian and its Pseudoinverse. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2487–2503. SIAM, 2018.
8. Yang P. Liu, Sushant Sachdeva, and Zejun Yu. Short Cycles via Low-Diameter Decompositions. *SODA*, 2019.
9. Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM journal on computing*, 15(4):1036–1053, 1986.
10. Merav Parter and Eylon Yogev. Distributed Computing Made Secure: A Graph Theoreric Approach. *SODA*, 2019.
11. Merav Parter and Eylon Yogev. Low Congestion Cycle Covers and Their Applications. *SODA*, 2019.
12. David Peleg. *Distributed Computing: A Locality-sensitive Approach*. SIAM, 2000.
13. Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 81–90. ACM, 2004.

Satisfiability Thresholds for Regular Occupation Problems

Konstantinos Panagiotou
LMU München, Germany
kpanagio@math.lmu.de

Matija Pasch
LMU München, Germany
pasch@math.lmu.de

Abstract

In the last two decades the study of random instances of constraint satisfaction problems (CSPs) has flourished across several disciplines, including computer science, mathematics and physics. The diversity of the developed methods, on the rigorous and non-rigorous side, has led to major advances regarding both the theoretical as well as the applied viewpoints. The two most popular types of such CSPs are the Erdős-Rényi and the random regular CSPs.

Based on a ceteris paribus approach in terms of the density evolution equations known from statistical physics, we focus on a specific prominent class of problems of the latter type, the so-called *occupation problems*. The regular r-in-k occupation problems resemble a basis of this class. By now, out of these CSPs only the satisfiability threshold – the largest degree for which the problem admits asymptotically a solution – for the 1-in-k occupation problem has been rigorously established. In the present work we take a general approach towards a systematic analysis of occupation problems. In particular, we discover a surprising and explicit connection between the 2-in-k occupation problem satisfiability threshold and the determination of *contraction coefficients*, an important quantity in information theory measuring the loss of information that occurs when communicating through a noisy channel. We present methods to facilitate the computation of these coefficients and use them to establish explicitly the threshold for the 2-in-k occupation problem for $k = 4$. Based on this result, for general $k \geq 5$ we formulate a conjecture that pins down the exact value of the corresponding coefficient, which, if true, is shown to determine the threshold in all these cases.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures; Mathematics of computing → Discrete mathematics; Mathematics of computing → Graph theory; Mathematics of computing → Probability and statistics; Mathematics of computing → Probabilistic inference problems; Mathematics of computing → Probabilistic reasoning algorithms; Mathematics of computing → Information theory

Keywords and phrases Constraint satisfaction problem, replica symmetric, contraction coefficient, first moment, second moment, small subgraph conditioning

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.90

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1811.00991`.

Funding *Konstantinos Panagiotou*: The research leading to these results has received funding from the European Research Council, ERC Grant Agreement 772606–PTRCSP.

1 Introduction

Inspired by the pioneering work [22] of Erdős and Rényi in 1960, random discrete structures have been systematically studied in literally thousands of contributions. The initial motivation of this research was to study open problems in graph theory and combinatorics. In the following decades, however, the application of such models proved useful as a unified approach

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 90; pp. 90:1–90:14

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

to treat a variety of problems in several fields. To mention just a few, random graphs turned out to be valuable in solving fundametal theoretical and practical problems, such as the development of error correcting codes [34], the study of statistical inference through the stochastic block model [1], and the establishment of lower bounds in complexity theory [27, 24].

The results of the past years of research suggest the existence of *phase transitions* in many classes of random discrete structures, i.e. a specific value of a given model parameter at which the properties of the system in question change dramatically. Constraint satisfaction problems are one specific type of such structures that tend to exhibit this remarkable property and that are of particular interest in too many areas to mention, covering complexity theory, combinatorics, statistical mechanics, artificial intelligence, biology, engineering and economics. An instance of a CSP is defined by a set of variables that take values in – typically finite – domains and a set of constraints, where each constraint is satisfied for specific assignments of the subset of variables it involves. A major computational challenge is to determine whether such an instance is satisfiable, i.e. to determine if there is an assignment of all variables that satisfies all constraints.

Since the 1980s non-rigorous methods have been introduced in statistical physics that are targeted at the analysis of phase transitions in *random* CSPs [37, 36, 33]. Within this line of research, a variety of exciting and unexpected phenomena were discovered, as for example the existence of multiple phase transitions with respect to the structure of the solution space in random CSPs; these transitions may have a significant impact on the hardness of the underlying instances. Since then these methods and the description of the conjectured regimes have been heavily supported by several findings, including the astounding empirical success of randomized algorithms like *belief* and *survey propagation* [9], as well as rigorous verifications, most prominently the phase transition in k-SAT [19] (for sufficiently large k) and the condensation phase transition in many important models [14]. However, a complete rigorous study is still a big challenge for computer science and mathematics.

Usually, the relevant model parameter of a random CSP is a certain problem specific *density* as illustrated below. The main focus of research is to study the occurrence of phase transitions in the solution space structure and in particular the existence of *(sharp) satisfiability thresholds*, i.e. critical values of the density such that the probability that a random CSP admits a solution tends to one as the number of variables tends to infinity for densities below the threshold, while this limiting probability tends to zero for densities above the threshold.

Random CSPs. The two most popular types of random CSPs are Erdős Rényi (ER) type CSPs and random regular CSPs. In both cases the number $n = |V|$ of variables and the number k of variables involved in each constraint is fixed. In ER type CSPs we further fix the number $m = |F|$ of constraints and thereby the *density* $\alpha = m/n$, i.e. the average number of constraints that a variable is involved in. In random regular CSPs we only consider instances where each variable is involved in the same number d of constraints, which fixes the *density* d as well as the number $m = dn/k$ of constraints. In a second step we randomly choose the sets of satisfying assignments for each constraint depending on the problem. For example, in the prominent k-SAT one forbidden assignment is chosen uniformly at random from all possible assignments of the involved binary variables for each constraint independently. Another famous example is the coloring of hypergraphs, where the constraints are attached to the hyperedges and the variables to the vertices of the hypergraph, i.e. the variables involved in a constraint correspond to the vertices incident to a hyperedge. In this case the satisfying assignments are determined since each constraint is violated iff all involved vertices take the same color.

In our work we focus on the class of random regular CSPs where the choice of satisfying assignments per constraint is determined, i.e. a class that covers the regular occupation problems and the coloring of (d-regular k-uniform) hypergraphs amongst others, sparing problems with random constraints like k-SAT and XORSAT. A unique feature of this class is, intuitively speaking, that the local structure of almost all instances is fixed almost everywhere for sufficiently large n. The lack of randomness makes this class particularly accessible for an analysis of the asymptotic solution space structure and significantly simplifies simulations based on the well-known *population dynamics*. Using such simulations, non-rigorous results for this class have been mostly established for the case where the variables are binary valued, so called occupation problems, or restricted to variants of hypergraph coloring for non-binary variables. Besides the extensive studies on the coloring of simple graphs, i.e. $k = 2$, the only rigorous results derived so far consider the arguably most simple type of occupation problems where each constraint is satisfied if exactly one involved variable evaluates to true, which we refer to as d-regular 1-in-k occupation problem. In our current work we strive to extend these results to general d-regular r-in-k occupation problems, i.e. problems where each constraint is satisfied if r out of the k involved variables evaluate to true.

1.1 Occupation Problems

We continue with the formal definition of the class of problems we consider. Let $k, d \in \mathbb{Z}_{>1}$ and $r \in [k-1] := \{1, \dots, k-1\}$ be fixed. Additionally, we are given non-empty sets V of variables and constraints F. We will use the convention to index elements of V with the letter i and elements of F with the letter a (and subsequent letters) in the remainder. Then an instance o of the d-regular r-in-k occupation problem is specified by a sequence $o = (v(a))_{a \in F}$ of $m = |F|$ subsets $v(a) \subseteq V$ of size k such that each of the $n = |V|$ variables is contained in d of the subsets. In graph theory the instance o has a natural interpretation as a (d, k)-biregular graph (or d-regular k-factor graph) with node sets $V \dot\cup F$ and edges $\{i, a\} \in E$ if $i \in v(a)$.

Given an instance o as just described, we say that an assignment $x \in \{0,1\}^V$ *satisfies* a constraint $a \in F$ if $\sum_{i \in v(a)} x_i = r$, otherwise x *violates* a. If x satisfies all constraints $a \in F$, then x is a *solution* of o. We write $z(o)$ for the number of solutions of o. An example of a 4-regular 2-in-3 occupation problem is shown in Figure 1a.

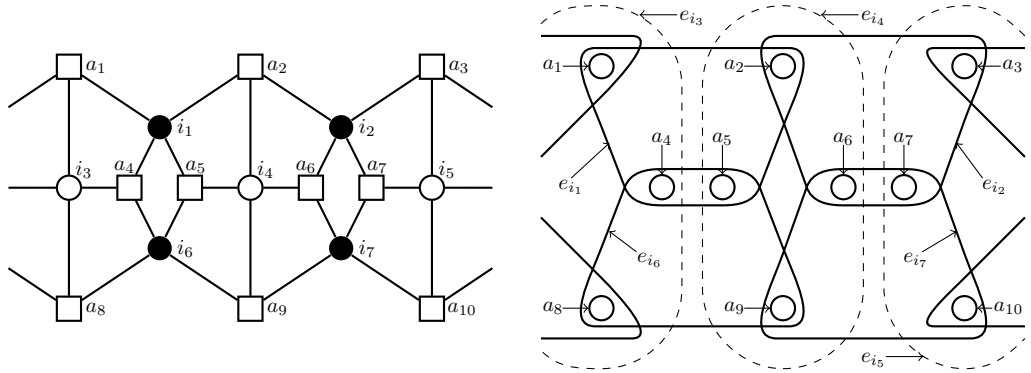
Further, for given $m, n \in \mathbb{Z}_{>0}$ let $\mathcal{O} = \mathcal{O}(k, d, n, m)$ denote the set of all instances o with variables $V = [n]$ and constraints $F = [m]$. If $\mathcal{O}$ is not empty, then the random d-regular r-in-k occupation problem O is the random variable O equipped with the uniform distribution $P = P_{\mathcal{O}}$ on $\mathcal{O}$ and $Z = z(O)$ the number of solutions of O.

1.2 Examples and Related Problems

A problem that is closely related and can be reduced to the d-regular r-in-k occupation problem is the d-regular positive r-in-k SAT problem, a variant of k-SAT introduced above. In this case, we consider a boolean formula

$$f = \bigwedge_{a \in F} c_a, \quad c_a = \bigvee_{i \in v(a)} i,\ a \in F,$$

in conjunctive normal form with m clauses over n variables $i \in V$, such that no literal appears negated (hence *positive* r-in-k SAT), and where each clause c_a is the disjunction of k literals and each variable appears in exactly d clauses (hence d-regular). The decision problem is to determine if there exists an assignment x such that exactly r literals in each



(a) Solution of the 2-occupation problem. **(b)** A 2-factor in a hypergraph.

Figure 1 On the left we see a solution of the 4-regular 2-in-3 occupation problem on a 4-regular 3-factor graph, where the rectangles and circles depict the constraints (factors) and variables (filled if they take the value one in the solution). The figure on the right shows a 2-factor in a 3-regular 4-uniform hypergraph, where the circles, solid and dashed shapes represent the vertices, hyperedges in the 2-factor and the other hyperedges respectively.

clause evaluate to true (hence r-in-k SAT). In [39] the satisfiability threshold for this problem was determined for $r = 1$, i.e. the case where exactly one literal in each clause evaluates to true. One of our main results, Theorem 1.1, solves this problem when $r = 2$ and $k = 4$.

Our second example deals with a prominent problem related to graph theory. A k-regular d-uniform hypergraph h is a pair $h = (F, E)$ with $m = |F|$ vertices and $n = |E|$ (hyper-)edges such that each edge contains d vertices and the degree of each vertex is k. An *r-factor* E' is a subset of the hyperedges such that each vertex $a \in F$ is incident to r hyperedges $e_i \in E'$. In this case the problem is to determine if h has an r-factor. For example, the case $r = 1$ is the well-known perfect matching problem and the threshold was determined in [16]. An example of a 2-factor in a hypergraph is shown in Figure 1b. Theorem 1.1 solves also this problem for $r = 2$ and $k = 4$.

There are several other problems in complexity and graph theory that are closely related to the examples above. The satisfiability threshold in Theorem 1.1 also applies to a variant of the vertex cover problem (or hitting set problem from set theory perspective), where we choose a subset of the vertices (variables with value one) in a d-regular 4-uniform hypergraph such that each hyperedge is incident to exactly two vertices in the subset. Analogously, Theorem 1.1 also establishes the threshold for a variant of the set cover problem in set theory corresponding to 2-factors in hypergraphs, i.e. given a family of d-subsets (hyperedges) and a universe (vertices) with each element contained in four subsets, the problem is to find a subfamily of the subsets such that each element of the universe is contained in exactly two subsets of the subfamily. Further, Theorem 1.1 can e.g. also be used to give sufficient conditions for the (asymptotic) existence of Euler families in regular uniform hypergraphs as discussed in [6].

1.3 Main Results

The d-regular 1-in-k occupation problem has been completely solved in [39, 16], which also covers the d-regular 2-in-3 occupation problem due to color symmetry. Our first result addresses the next non-trivial case, namely the location of the satisfiability threshold of the

random d-regular 2-in-4 occupation problem. For $k \in \mathbb{Z}_{>3}$ let

$$w_1^* = w_1^*(k) = \frac{2}{k},\ w_2^* = w_2^*(k) = \binom{k}{2}^{-1} \text{ and } d^* = d^*(k) = \frac{kH(w_1^*)}{kH(w_1^*) + \ln(w_2^*)}, \tag{1}$$

where $H(p) = -p\ln(p) - (1-p)\ln(1-p)$ is the binary entropy of $p \in [0,1]$. The following theorem establishes the location of the threshold at $d^*(4) \approx 2.83$ for $k = 4$.

▶ **Theorem 1.1** (2-in-4 Occupation Satisfiability Threshold). *Let $k = 4$, $d \in \mathbb{Z}_{>1}$, and $\mathcal{O} = \mathcal{O}(k,d,n,m)$, $Z = Z_{k,d,n,m}$ be as in Section 1.1.*

(a) *The set $\mathcal{O}$ is non-empty iff $m = m(n) = \frac{dn}{k}$. Then, the number Z of solutions is zero almost surely if k does not divide $2n$, i.e. $\mathbb{P}[Z = 0] = 1$. Further, the threshold d^* is not an integer.*

(b) *There exists a sharp satisfiability threshold at d^*, i.e. for any increasing sequence $(n_i)_{i\in\mathbb{Z}_{>0}} \subseteq \mathcal{N} = \{n : dk^{-1}n,\ 2k^{-1}n \in \mathbb{Z}_{>0},\}$ and $m_i = m(n_i)$ we have*

$$\lim_{i\to\infty} \mathbb{P}[Z > 0] = \begin{cases} 1 & , d < d^* \\ 0 & , d > d^* \end{cases}.$$

We prove Theorem 1.1 using the second moment method for Z and the small subgraph conditioning method to boost the probability asymptotically to one below the threshold d^*. However, an important question remains at this point, namely what happens when $k > 4$ or $r > 2$.

Our second main result in this paper addresses the behavior for $k > 4$, which can be directly extended to $r > 2$. In particular, a main technical contribution in proving Theorem 1.1 is the optimization of a certain multivariate function that appears in the computation of the second moment, which encodes the interplay between the "similarity" of various assignments and the change in the corresponding probability of being satisfying that they induce. A similar but more complex function appears in the computation of the second moment for $k > 4$, but there we are unfortunately not able to pin down the maximizer. However, apart from that, we discover a surprising connection between this optimization problem and a seemingly unrelated fundamental problem in information theory. In particular, we find that the optimization problem is equivalent to developing a so-called *strong data processing inequality (SDPI)*, which, roughly speaking, encodes the minimum amount of loss in the process of communication through a noisy channel. Such inequalities are of particular importance in the analysis of noisy channels.

We postpone the formal definitions and more relevant background to the next section. Further, we show that our anticipated forms of the corresponding SDPIs directly yield the locations of the global extrema required for our satisfiability threshold proof and thereby imply the following theorem based on Conjecture 1.3.

▶ **Theorem 1.2** (2-in-k Occupation Satisfiability Threshold). *Assume that Conjecture 1.3 is true. Then Theorem 1.1 holds for any $k \in \mathbb{Z}_{>3}$.*

We are confident that this surprising connection does not only apply to 2-in-k occupation problems, but to all r-in-k occupation problems and we believe that it also extends to other classes of random CSPs. Hence, this bridge facilitates the combination of the methods that have been devised in information theory and the study of random graphs, ultimately relating the second moment method to the hypercontractivity ribbon.

1.4 Contraction Coefficients

One central concept in information theory [28, 17] is the notion of a communication channel. Let us assume for concreteness that we have sets $[m]$ and $[n]$ of input and output symbols respectively. We consider the communication through a noisy channel, that is, for a given input $x \in [m]$ the output is $y \in [n]$ with a certain fixed probability $W_{y,x}$. Thus, the channel is completely characterized by its column stochastic transition probability matrix $W = (W_{y,x})_{y\in[n],x\in[m]} \in [0,1]^{n\times m}$.

In a second step, let us consider a distribution P on $[m]$ with probability mass function (pmf) $p \in [0,1]^m$, i.e. a distribution on the inputs. Then the corresponding distribution Q on the received outputs is given by the pmf $q = Wp \in [0,1]^n$. The study of the properties of such channels involves the quantification of the communicated information and further a channel capacity, i.e. the maximum amount of transmittable information. The *data processing inequality (DPI)* is a fundamental result stating that information can only *decrease* when communicated through a noisy channel.

The version of the DPI, see e.g. Lemma 3.11 in [17], relevant here is as follows. Fix a reference input distribution P^* with pmf $p^* \in \mathbb{R}^m$, i.e. the reference output distribution Q^* has the pmf $q^* = Wp^* \in \mathbb{R}^n$. If we then consider an input distribution P with pmf p and the corresponding output distribution Q with pmf $q = Wp$, it is easier to distinguish the distributions P and P^* *before* the transmission. This suggests a loss of information in the process of communication; formally, this means that

$$D_{\mathrm{KL}}(P \parallel P^*) \geq D_{\mathrm{KL}}(Q \parallel Q^*), \quad \text{where} \quad D_{\mathrm{KL}}(P \parallel P^*) = \sum_{x\in[m]} p_x \ln\left(\frac{p_x}{p_x^*}\right).$$

The quantity $D_{\mathrm{KL}}(\cdot \parallel \cdot)$ is the well-known K(ullback)–L(eibler) divergence and one of the most important means of measuring the similarity between given distributions.

This fundamental DPI can be further improved by introducing the optimal ratio $d_* = d_*(P^*, W)$ of $D_{\mathrm{KL}}(Q \parallel Q^*)$ and $D_{\mathrm{KL}}(P \parallel P^*)$ and deriving the tight bound

$$d_* D_{\mathrm{KL}}(P \parallel P^*) \geq D_{\mathrm{KL}}(Q \parallel Q^*) \quad \text{with} \quad d_* = \sup_{P\neq P^*} \frac{D_{\mathrm{KL}}(Q \parallel Q^*)}{D_{\mathrm{KL}}(P \parallel P^*)}.$$

In particular d_* is independent of the input distribution P and the output distribution $Q = Q(P)$. A data processing inequality of this type is referred to as a strong data processing inequality (SDPI) with *contraction coefficient* d_* [2, 4, 3]. In this sense the contraction coefficient $d^*(P^*, W) = d^*(X;Y)$ can be regarded as an alternative measure for the mutual information $I(X;Y)$, i.e. the KL divergence of the distribution of (X,Y) with respect to the distribution of X and Y assuming independence, where the distribution of (X,Y) has the pmf $(W_{yx}p_x^*)_{x,y} \in \mathbb{R}^{n\times m}$. This quantity is of great importance in the analysis of noisy channels and hence not only of interest in theory building, but also in many applications covering image and audio processing, biology, economics and engineering.

1.5 The Conjecture

Let $k \in \mathbb{Z}_{>3}$, $w_1^* = \frac{2}{k}$, $w_2^* = \binom{k}{2}^{-1}$ as defined in (1),

$$W = (W_{(s-1),(t-1)})_{s,t\in[3]} = \begin{pmatrix} 1-2w_1^* & 1-\frac{3}{2}w_1^* & 1-w_1^* \\ 2w_1^* & w_1^* & 0 \\ 0 & \frac{1}{2}w_1^* & w_1^* \end{pmatrix}, \tag{2}$$

and for $w \in \mathcal{W} = \{w \in [0,1]^2 : 2w_1 - 1 \leq w_2 \leq w_1\}$ let

$$p = (p_{s-1})_{s\in[3]} = \begin{pmatrix} 1 - 2w_1 + w_2 \\ 2(w_1 - w_2) \\ w_2 \end{pmatrix}, \quad q = Wp = (q_{s-1})_{s\in[3]} = \begin{pmatrix} 1 - 2w_1^* + w_1^* w_1 \\ 2w_1^*(1 - w_1) \\ w_1^* w_1 \end{pmatrix}. \quad (3)$$

Notice that W is the transition probability matrix of a (fixed) channel for fixed k, that p, q are pmfs for all $w \in \mathcal{W}$ and further any pmf on $\{0,1,2\}$ can be attained by p. As discussed in Section 1.3, $w \in \mathcal{W}$, p and q quantify the similarity of two random satisfying assignments in the following sense. Intuitively and due to symmetry, a given variable i involved in any given constraint takes the value one with probability $w_1^* = \mathbb{P}[X_i = 1]$, while two given variables i, j involved in the constraint both take the value one with probability $w_2^* = \mathbb{P}[X_i + X_j = 2]$ under a random satisfying assignment X. The parameter $w_1 = \mathbb{P}[Y_i = 1 | X_i = 1]$ gives the conditional probability that i takes the value one under a second satisfying assignment Y given that i takes the value one under X, while $w_2 = \mathbb{P}[Y_i + Y_j = 2 | X_i + X_j = 2]$ gives the conditional probability that both i, j take the value one under Y assuming that they both take the value one under X. Further, p is the pmf of $(Y_i + Y_j)|(X_i + X_j) = 2$, i.e. of the distribution of the number $(Y_i + Y_j)$ of ones taken by i and j under Y given that i, j take one under X, while q is the pmf of the distribution of the number $(X_i + Y_i)$ of ones taken by i under X and Y. In this sense, w_1 and w_2 quantify the similarity of two satisfying assignments X and Y. For example, the choice $w_1 = w_2 = 1$ of parameters implies that Y is determined by X and hence, intuitively, corresponds to a minimum loss of information.

Let P^* be the reference input distribution with pmf $p^* = p(w^*)$, then by the discussion above and in Section 1.4 we can employ the contraction coefficient $d_* = d_*(k) = d_*(P^*, W)$ to quantify the loss of information in a communication through the channel W, and further expect that d_* is attained at $w = (1,1)$.

▶ **Conjecture 1.3** (Contraction Coefficient Conjecture). *The contraction coefficient d_* is attained for the degenerate input pmf p at two, that is,*

$$w_1 = w_2 = 1 \quad \textit{and} \quad d_* = \frac{H(w_1^*)}{-\ln(w_2^*)}.$$

In our contribution, we do not only show that the computation of d_* is equivalent to the optimization problem in the second moment method, but that Conjecture 1.3 is actually equivalent to the applicability of the second moment method.

1.6 Related Work

The regular version of the random 1-in-k occupation problem (and related problems) has been completely solved in [16, 39] using the first and second moment method with small subgraph conditioning. The paper [41] shows that $d^*(k) \geq 2$ for $k \in \mathbb{Z}_{>1}$ in the d-regular 2-in-k occupation problem, i.e. the existence of 2-factors in k-regular simple graphs. A recent discussion of 2-factors (and the related Euler families) that does not rely on the probabilistic method is presented in [6]. Further, randomized polynomial time algorithms for the generation and approximate counting of 2-factors in random regular simple graphs have been introduced in [25].

The study of Erdős Rényi (hyper-)graphs was initiated by the ground breaking publication [22] in 1960 and turned into a fruitful field of research with many applications, including early results on 1-factors in simple graphs [23]. On the contrary, results for the random d-regular k-uniform (hyper-)graph ensemble were rare before the introduction of the configuration

(or pairing) model by Bollobás [8] and the development of the small subgraph conditioning method [30, 31] thereafter, see also [44]. While the derived proof scheme facilitated rigorous arguments to establish the existence and location of satisfiability thresholds of random regular CSPs [38, 7, 32, 12, 15, 20, 21, 5], the problems are treated on a case by case basis, while results on entire classes of random regular CSPs are still outstanding.

One of the main reasons responsible for the complexity of a rigorous analysis of random (regular) CSPs seems to be a conjectured structural change of the solution space for increasing densities. This hypothesis has been put forward by physicists, verified in parts and mostly for ER ensembles, further led to new rigorous proof techniques [19, 15, 13] and to randomized algorithms [9, 35] for NP-hard problems that are not only of great value in practice, but can also be employed for precise numerical (though non-rigorous) estimates of satisfiability thresholds An excellent introduction to this *replica theory* can be found in [36, 33, 43]. Specifically, numerical results indicating the satisfiability thresholds for d-regular r-in-k occupation problems (more general variants, and for ER type hypergraphs) based on this conjecture were discussed in various publications [10, 18, 42, 26, 29, 46, 45], where *occupation problems* were introduced for the first time in [40].

Another fundamental obstacle in the rigorous analysis is of a very technical nature and directly related to the second moment method as discussed in detail in our current presentation. In the case of regular r-in-k occupation problems (amongst others) this optimization problem is closely related to the computation of the *contraction coefficient* (for fixed channels and reference distributions) known from information theory. For a general introduction to information theory we recommend [17], while profound discussions and applications of contraction coefficients can be found in [3, 4] and references therein.

1.7 Open Problems

As mentioned in Section 1.2, we focus on the analysis of random regular CSPs with determined constraints. The starting point for this systematic study are r-in-k occupation problems, where we rigorously established the threshold for $r = 2$ and $k = 4$. However, apart from the optimization step in the second moment calculation our proof canonically extends to the general case. A rigorous proof of this step for general r and k is involved, but further assumptions may significantly simplify the analysis. For example, as an extension of the current work one may focus on r-in-$2r$ occupation problems, where the constraints are symmetric in the colors. As can be seen from our proof, this yields useful symmetry properties of the objective function $\frac{D_{\mathrm{KL}}(Q \parallel Q^*)}{D_{\mathrm{KL}}(P \parallel P^*)}$. Further, as suggested by the literature [11, 13, 14] such *balanced* problems [45, 46] are usually more accessible to a rigorous study. On the other hand, the optimization usually also significantly simplifies if only carried out for $k \geq k_0(r)$ for some large $k_0(r)$, as this pushes the minimum to the boundary of the function domain.

Apart from the generalizations discussed above, results for the r-in-k occupation problems are also still outstanding for Erdős-Rényi type CSPs. An analysis of this related problem might allow to tackle the crucial optimization step from a different perspective and thereby also help to establish the thresholds for the regular version.

From the algorithmic perspective, although some methods have been developed for simple graphs [25], we are not aware of algorithms designed specifically to identify solutions of the regular occupation problems (like WalkSAT for the k-SAT problem), only general methods like belief propagation based decimiation. However, problem specific obstacles for the design of such algorithms were discussed in [46].

2 Proof Techniques

In this section we give a high-level overview of our proof, in particular we present the major steps that lead to the main results. We make heavy use of the so-called *configuration model* for the generation of random instances in the form used by Moore [39].

2.1 The Configuration Model

Working with the uniform distribution on d-regular k-uniform hypergraphs directly is challenging. Instead, we show Theorems 1.2 and 1.1 for occupation problems on configurations. A d-regular k-configuration is simply a bijection $g : [n] \times [d] \to [m] \times [k]$, where the v-edges $(i, h) \in \mathrm{dom}(g)$ represent pairs of variables $i \in [n]$ and i-edges, i.e. half-edge indices $h \in [d]$. The image $(a, h') = g(i, h)$ is an f-edge, i.e. a pair of a constraint (factor) $a \in [m]$ and an a-edge (or half-edge) $h' \in [k]$, indicating that the i-edge h of the variable i is wired to the a-edge h' of a and thereby suggesting that i is connected to a in the corresponding d-regular k-factor graph. The number of such d-regular k-configurations on n variables can be easily determined and is given by $(dn)! = (km)!$, hence the uniform distribution on configurations is suitable for combinatorial arguments. Further, the occupation problem on factor graphs directly translates to configurations, which allows to introduce the number Z of solutions of the occupation problem on the random configuration G. In the following we discuss the proof of the analogues to Theorems 1.2 and 1.1 for configurations and further the translation of these results back to factor graphs and hypergraphs.

2.2 The First Moment Method

In the first step we apply the first moment method to the occupation problem on configurations, yielding the following result.

▶ **Lemma 2.1** (First Moment Method). *Let $k \in \mathbb{Z}_{>3}$, $d \in \mathbb{Z}_{>1}$. For $n \in \mathcal{N}$ tending to infinity we have*

$$\mathbb{E}[Z] \sim \sqrt{d} e^{n\phi_1}, \text{ where } \phi_1 = \frac{d}{k}(-\ln(w_2^*)) - (d-1)H(w_1^*).$$

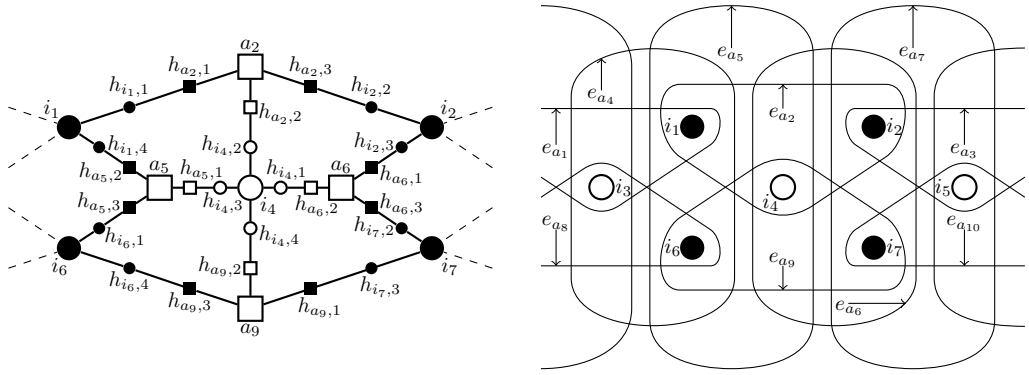
In particular this implies that $\mathbb{E}[Z] \to \infty$ for $d < d^*$ and $\mathbb{E}[Z] \to 0$ for $d > d^*$ with d^* as defined in (1). With an application of Markov's inequality we see that $\mathbb{P}[Z > 0] \to 0$ for $d > d^*$. The map ϕ_1 is known as annealed free entropy density. While the domain of ϕ_1 is trivial in this case (and further in any r-in-k occupation problem), it is non-trivial for the vast majority of CSPs, also covering the general occupation problem.

2.3 The Second Moment Method

Let $k \in \mathbb{Z}_{>3}$, $d \in \mathbb{Z}_{>1}$, further let p, q and $\mathcal{W}$ be the notions from Section 1.5, the distributions P, Q be given by the pmfs p, q and let $\phi_2 : \mathcal{W} \to \mathbb{R}$ be given by

$$\phi_2(w) = \frac{d}{k} D_{\mathrm{KL}}(P \parallel P^*) - (d-1) D_{\mathrm{KL}}(Q \parallel Q^*) \text{ for } w \in \mathcal{W}. \tag{4}$$

Conjecture 1.3 can be used to show that ϕ_2 attains its global minimum at zero iff $w = w^*$ and $d < d^*$. The proof for the specific case $k = 4$ will be presented later in this work. This conclusion then allows to derive the following result using Laplace's method for sums.



(a) occupation problem on configurations. **(b)** 4-regular 2-in-3 vertex cover.

Figure 2 The figure on the left shows the solution on a configuration corresponding to the solution in Figure 1. We only denoted a-edges (small boxes, filled if they the a-edge takes the value one) and i-edges (small circles, filled if the i-edge takes the value one) instead of f-edges and v-edges for brevity (e.g. $h_{a_1,1}$ instead of $(a_1, h_{a_1,1})$). The figure on the right illustrates the corresponding 2-in-3 vertex cover (given by the filled circles).

▶ **Lemma 2.2** (Second Moment Method). *Assume that Conjecture 1.3 holds. Then we have*

$$\frac{\mathbb{E}[Z^2]}{\mathbb{E}[Z]^2} \sim \sqrt{\frac{2}{(2\pi)^2 \prod_{i=0}^{2} p_i(w^*)}} \sqrt{\frac{(2\pi)^2}{\det\left(\frac{k}{\sqrt{2d}} H\right)}} = \sqrt{\frac{k-1}{k-d}},$$

for $n \in \mathcal{N}$ tending to infinity and where H denotes the Hessian of ϕ_2 at $w = w^*$.

Using Lemma 2.2 and Chebyshev's inequality we see that $\mathbb{P}[Z = 0] \leq \sqrt{\frac{k-1}{k-d}} - 1$. While this bound suggests a threshold exists, we need to show that the threshold at d^* is *sharp*.

2.4 Small Subgraph Conditioning

We conclude the proof of the theorem (for configurations) by applying the small subgraph conditioning method to establish that the satisfiability threshold d^* is *sharp*.

▶ **Theorem 2.3** (Small Subgraph Conditioning). *Let Z and $X_1, X_2, \ldots$ be non-negative integer-valued random variables. Suppose that $\mathbb{E}[Z] > 0$ and that for each $\ell \in \mathbb{Z}_{>0}$ there are constants $\lambda_\ell \in \mathbb{R}_{>0}$, $\delta_\ell \in \mathbb{R}_{>-1}$ such that*

(a) *for any $\bar{\ell}$ the variables $X_\ell, \ldots, X_{\bar{\ell}}$ are asymptotically independent and Poisson distributed with $\mathbb{E}[X_\ell] \sim \lambda_\ell$,*

(b) *for any sequence $r_1, \ldots, r_{\bar{\ell}}$ of non-negative integers,*

$$\frac{\mathbb{E}\left[Z \prod_{\ell=1}^{\bar{\ell}} (X_\ell)_{r_\ell}\right]}{\mathbb{E}[Z]} \sim \prod_{\ell=1}^{\bar{\ell}} \mu_\ell^{r_\ell},\ \mu_\ell = \lambda_\ell(1+\delta_\ell),$$

(c) *we explain the variance, i.e.*

$$\frac{\mathbb{E}[Z^2]}{\mathbb{E}[Z]^2} \sim \exp\left(\sum_{\ell=1}^{\infty} \lambda_\ell \delta_\ell^2\right),\ \left|\sum_{\ell=1}^{\infty} \lambda_\ell \delta_\ell^2\right| < \infty.$$

Then we have $\lim_{n\to\infty} \mathbb{P}[Z > 0] = 1$.

The discussion of the factorial moments in Theorem 2.3 (b) is performed in detail, which requires additional concepts and complex combinatorial arguments. To facilitate the presentation we also give a self-contained proof of the following well-known theorem on the expected number of small cycles (the variables X_ℓ in Theorem 2.3), which can then be extended to a proof of Theorem 2.3 (b). In order to understand what a cycle in a configuration is, we notice that we can represent a configuration g by an equivalent graph with (disjoint) vertex sets given by the variables $V = [n]$, constraints (factors) $F = [m]$, v-edges $H_1 = [n] \times [d]$ and f-edges $H_2 = [m] \times [k]$, where each variable $i \in [n]$ connects to all its v-edges $(i, h_1) \in H_1$, each constraint $a \in [m]$ to all its f-edges $(a, h_2) \in H_2$ and a v-edge (i, h_1) connects to an f-edge (a, h_2) if $g(i, h_1) = (a, h_2)$. Since we are mostly interested in the factor graph associated with a configuration we divide lengths of paths by three, e.g. a cycle of length four in a configuration is actually a cycle of length twelve in its equivalent graph representation. Figures 1a and 2a show an example of a factor graph and the corresponding configuration in its graph representation.

▶ **Theorem 2.4** (Number of Small Cycles). *For $\ell \in \mathbb{Z}_{>0}$ let X_ℓ be the number of 2ℓ-cycles in G, further*

$$\lambda_\ell = \frac{[(k-1)(d-1)]^\ell}{2\ell},$$

and $Z_\ell \sim \mathrm{Po}(\lambda_\ell)$ be independent Poisson distributed random variables. Then the random variables X_ℓ converge in distribution to Z_ℓ for $n \to \infty$, jointly for all $\ell \in \mathbb{Z}_{>0}$.

Using Theorem 2.4 we determine μ_ℓ, δ_ℓ for $\ell \in \mathbb{Z}_{>0}$ and use these results to establish the remaining parts of Theorem 2.3.

▶ **Lemma 2.5.** *The constants μ_ℓ and δ_ℓ for $\ell \in \mathbb{Z}_{>0}$ in Theorem 2.3 are given by*

$$\delta_\ell = \left(-\frac{1}{k-1}\right)^\ell.$$

2.5 Translation of the Results

We first translate the results for configurations to factor graphs using Theorem 2.4, i.e. the contiguity of the factor graph model with respect to the configuration model. For completeness we then also provide self-contained proofs to establish the application to hypergraphs with labeled and unlabeled hyperedges (where the constraints may be attached to either the vertices or to the hyperedges). This establishes our claims in Sections 1.2 and 1.3 except for the verification of Conjecture 1.3 for $k = 4$.

2.6 Contraction Coefficient for $k = 4$

Finally, we prove Conjecture 1.3 for $k = 4$, i.e. we derive Theorem 1.1 from Theorem 1.2. Using a slightly different parametrization and simplifying the KL divergence in the nominator yields

$$\begin{aligned}
d_*(4) &= \sup_{w \in \mathcal{W} \setminus \{w^*\}} R(w),\ R(w) = \frac{D_2(w_1)}{D_1(w)},\\
D_1(w) &= (w_1 - w_2)\ln(6(w_1 - w_2)) + 2w_2\ln(3w_2) + (1 - w_1 - w_2)\ln(6(1 - w_1 - w_2)),\\
D_2(w_1) &= w_1\ln(2w_1) + (1 - w_1)\ln(2(1 - w_1)),\\
\mathcal{W} &= \{w \in [0,1]^2 : w_2 \le w_1,\ w_2 \le 1 - w_1\}.
\end{aligned}$$

We focus on suitable lower bounds for D_1, therefore we minimize D_1 with respect to w_2, yielding

$$D_{\min}(w_1) = w_1 \ln(6(w_1 - w_2)) + (1 - w_1) \ln(6(1 - w_1 - w_2)),$$
$$w_2 = w_2(w_1) = \frac{1}{3}\left(2 - \sqrt{12\left(w_1 - \frac{1}{2}\right)^2 + 1}\right),\ w_1 \in [0, 1].$$

Since R is symmetric to $w_1 = \frac{1}{2}$, it is sufficient to show that $R \leq d_*$ for $w_1 \leq \frac{1}{2}$. On this interval we lower bound $D_{\min}$ using the functions

$$D_-(w_1) = 2w_1 \ln\left(\frac{12}{5}w_1\right) + (1 - 2w_1)\ln(6(1 - 2w_1)),\ w_1 \in [0, \bar{w}_1],\ \text{and}$$
$$D_+(w_1) = 6\left(\frac{1}{2} - w_1\right)^2,\ w_1 \in [\bar{w}_1, 0.5],$$

where $\bar{w}_1 \approx 0.10831$ is an intersection point of D_- and D_+ that we determined numerically. Finally, we use monotonicity arguments for the corresponding upper bounds of R to derive $R \leq d^*$.

References

1 E. Abbe. Community Detection and Stochastic Block Models: Recent Developments. *Journal of Machine Learning Research*, 18:1–86, 2018.

2 R. Ahlswede, P. Gács, and J. Körner. Bounds on conditional probabilities with applications in multi-user communication. *Z. Wahrscheinlichkeitstheorie und Verw. Gebiete*, 34(2):157–177, 1976. doi:10.1007/BF00535682.

3 V. Anantharam, A. Gohari, S. Kamath, and C. Nair. On hypercontractivity and the mutual information between Boolean functions. In *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 13–19, October 2013. doi:10.1109/Allerton.2013.6736499.

4 V. Anantharam, A. Gohari, S. Kamath, and C. Nair. On hypercontractivity and a data processing inequality. In *2014 IEEE International Symposium on Information Theory*, pages 3022–3026, June 2014. doi:10.1109/ISIT.2014.6875389.

5 H. Assiyatun and N. C. Wormald. 3-star factors in random d-regular graphs. *European J. Combin.*, 27(8):1249–1262, 2006. doi:10.1016/j.ejc.2006.05.003.

6 M. A. Bahmanian and M. Šajna. Quasi-Eulerian hypergraphs. *Electron. J. Combin.*, 24(3):Paper 3.30, 12, 2017.

7 V. Bapst, A. Coja-Oghlan, and C. Efthymiou. Planting colourings silently. *Combin. Probab. Comput.*, 26(3):338–366, 2017. doi:10.1017/S0963548316000390.

8 B. Bollobás. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European J. Combin.*, 1(4):311–316, 1980. doi:10.1016/S0195-6698(80)80030-8.

9 A. Braunstein, M. Mézard, and R. Zecchina. Survey propagation: an algorithm for satisfiability. *Random Structures Algorithms*, 27(2):201–226, 2005. doi:10.1002/rsa.20057.

10 T. Castellani, V. Napolano, F. Ricci-Tersenghi, and R. Zecchina. Bicolouring random hypergraphs. *J. Phys. A*, 36(43):11037–11053, 2003. doi:10.1088/0305-4470/36/43/026.

11 A. Coja-Oghla, T. Kapetanopoulos, and N. Müller. The replica symmetric phase of random constraint satisfaction problems. *arXiv e-prints*, page arXiv:1802.09311, February 2018. arXiv:1802.09311.

12 A. Coja-Oghlan. Constraint satisfaction: random regular k-SAT. In *Statistical physics, optimization, inference, and message-passing algorithms*, pages 231–251. Oxford Univ. Press, Oxford, 2016.

13 A. Coja-Oghlan, C. Efthymiou, N. Jaafari, M. Kang, and T. Kapetanopoulos. Charting the replica symmetric phase. *Comm. Math. Phys.*, 359(2):603–698, 2018. doi:10.1007/s00220-018-3096-x.

14 A. Coja-Oghlan, F. Krzakala, W. Perkins, and L. Zdeborova. Information-theoretic thresholds from the cavity method. *Adv. Math.*, 333:694–795, 2018.

15 A. Coja-Oghlan and K. Panagiotou. The asymptotic k-SAT threshold. *Adv. Math.*, 288:985–1068, 2016. doi:10.1016/j.aim.2015.11.007.

16 C. Cooper, A. Frieze, M. Molloy, and B. Reed. Perfect matchings in random r-regular, s-uniform hypergraphs. *Combin. Probab. Comput.*, 5(1):1–14, 1996. doi:10.1017/S0963548300001796.

17 I. Csiszár and J. Körner. *Information theory: coding theorems for discrete memoryless systems*. Probability and mathematical statistics. Academic Press, 1981. URL: https://books.google.de/books?id=wiTvAAAAMAAJ.

18 L. Dall'Asta, A. Ramezanpour, and R. Zecchina. Entropy landscape and non-Gibbs solutions in constraint satisfaction problems. *Phys. Rev. E (3)*, 77(3):031118, 16, 2008. doi:10.1103/PhysRevE.77.031118.

19 J. Ding, A. Sly, and N. Sun. Proof of the satisfiability conjecture for large k. In *STOC'15—Proceedings of the 2015 ACM Symposium on Theory of Computing*, pages 59–68. ACM, New York, 2015.

20 J. Ding, A. Sly, and N. Sun. Maximum independent sets on random regular graphs. *Acta Math.*, 217(2):263–340, 2016. doi:10.1007/s11511-017-0145-9.

21 J. Ding, A. Sly, and N. Sun. Satisfiability threshold for random regular NAE-SAT. *Comm. Math. Phys.*, 341(2):435–489, 2016. doi:10.1007/s00220-015-2492-8.

22 P. Erdős and A. Rényi. On the evolution of random graphs. *Magyar Tud. Akad. Mat. Kutató Int. Közl.*, 5:17–61, 1960.

23 P. Erdős and A. Rényi. On the existence of a factor of degree one of a connected random graph. *Acta Math. Acad. Sci. Hungar.*, 17:359–368, 1966. doi:10.1007/BF01894879.

24 U. Feige. Relations Between Average Case Complexity and Approximation Complexity. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 534–543, New York, NY, USA, 2002. ACM.

25 A. Frieze, M. Jerrum, M. Molloy, R. W. Robinson, and N. C. Wormald. Generating and counting Hamilton cycles in random regular graphs. *J. Algorithms*, 21(1):176–198, 1996. doi:10.1006/jagm.1996.0042.

26 M. Gabrié, V. Dani, G. Semerjian, and L. Zdeborová. Phase transitions in the q-coloring of random hypergraphs. *J. Phys. A*, 50(50):505002, 44, 2017. doi:10.1088/1751-8121/aa9529.

27 A. Galanis, D. Štefankovič, and E. Vigoda. Inapproximability for Antiferromagnetic Spin Systems in the Tree Non-uniqueness Region. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 823–831, New York, NY, USA, 2014.

28 R. G. Gallager. *Information theory and reliable communication*. Wiley, 1968.

29 S. Higuchi and M. Mézard. Correlation-based decimation in constraint satisfaction problems. *Journal of Physics: Conference Series*, 233(1):012003, 2010. URL: http://stacks.iop.org/1742-6596/233/i=1/a=012003.

30 S. Janson. Random regular graphs: asymptotic distributions and contiguity. *Combin. Probab. Comput.*, 4(4):369–405, 1995. doi:10.1017/S0963548300001735.

31 S. Janson, T. Łuczak, and A. Rucinski. *Random graphs*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley-Interscience, New York, 2000. doi:10.1002/9781118032718.

32 G. Kemkes, X. Pérez-Giménez, and N. C. Wormald. On the chromatic number of random d-regular graphs. *Adv. Math.*, 223(1):300–328, 2010. doi:10.1016/j.aim.2009.08.006.

33 F. Krzakala, F. Ricci-Tersenghi, L. Zdeborová, R. Zecchina, E. W. Tramel, and L. F. Cugliandolo, editors. *Statistical physics, optimization, inference, and message-passing algorithms*. Oxford University Press, Oxford, 2016.

34 S. Kudekar, T. Richardson, and R. Urbanke. Spatially coupled ensembles universally achieve capacity under belief propagation. *IEEE Trans. Inform. Theory*, 59:7761–7813, 2013.

35 E. Maneva, E. Mossel, and M. J. Wainwright. A new look at survey propagation and its generalizations. *J. ACM*, 54(4):Art. 17, 41, 2007. doi:10.1145/1255443.1255445.

36 M. Mézard and A. Montanari. *Information, physics, and computation*. Oxford Graduate Texts. Oxford University Press, Oxford, 2009. doi:10.1093/acprof:oso/9780198570837.001.0001.

37 M. Mézard, G. Parisi, and M. A. Virasoro. *Spin glass theory and beyond*, volume 9 of *World Scientific Lecture Notes in Physics*. World Scientific Publishing Co., Inc., Teaneck, NJ, 1987.

38 M. Molloy, H. D. Robalewska, R. W. Robinson, and N. C. Wormald. 1-factorizations of random regular graphs. *Random Structures Algorithms*, 10(3):305–321, 1997. doi:10.1002/(SICI)1098-2418(199705)10:3<305::AID-RSA1>3.0.CO;2-#.

39 C. Moore. The phase transition in random regular exact cover. *Ann. Inst. Henri Poincaré D*, 3(3):349–362, 2016. doi:10.4171/AIHPD/31.

40 T. Mora. *Geometry and Inference in Optimization and in Information Theory*. Theses, Université Paris Sud - Paris XI, September 2007. URL: https://tel.archives-ouvertes.fr/tel-00175221.

41 H. D. Robalewska. 2-factors in random regular graphs. *J. Graph Theory*, 23(3):215–224, 1996. doi:10.1002/(SICI)1097-0118(199611)23:3<215::AID-JGT1>3.3.CO;2-K.

42 C. Schmidt, N. Guenther, and L. Zdeborová. Circular coloring of random graphs: statistical physics investigation. *J. Stat. Mech. Theory Exp.*, 2016(8):083303, 28, 2016. doi:10.1088/1742-5468/2016/08/083303.

43 M. Talagrand. *Spin glasses: a challenge for mathematicians*, volume 46 of *Results in Mathematics and Related Areas. 3rd Series. A Series of Modern Surveys in Mathematics*. Springer-Verlag, Berlin, 2003.

44 N. C. Wormald. Models of random regular graphs. In *Surveys in combinatorics, 1999 (Canterbury)*, volume 267 of *London Math. Soc. Lecture Note Ser.*, pages 239–298. Cambridge Univ. Press, Cambridge, 1999.

45 L. Zdeborová and F. Krzakala. Quiet planting in the locked constraint satisfaction problems. *SIAM J. Discrete Math.*, 25(2):750–770, 2011. doi:10.1137/090750755.

46 L. Zdeborová and M. Mézard. Constraint satisfaction problems with isolated solutions are hard. *J. Stat. Mech. Theory Exp.*, 2008(12):P12004, 2008. URL: http://stacks.iop.org/1742-5468/2008/i=12/a=P12004.

Toward a Dichotomy for Approximation of H-Coloring

Akbar Rafiey
Department of Computing Science, Simon Fraser University, Burnaby, Canada
arafiey@sfu.ca

Arash Rafiey
Indiana State University, Terre Haute, IN, USA
Simon Fraser University, Burnaby, Canada
arash.rafiey@indstate.edu

Thiago Santos
Indiana State University, Terre Haute, IN, USA
tsantos2@sycamores.indstate.edu

Abstract

Given two (di)graphs G, H and a cost function $c : V(G) \times V(H) \to \mathbb{Q}_{\geq 0} \cup \{+\infty\}$, in the minimum cost homomorphism problem, MinHOM(H), we are interested in finding a homomorphism $f : V(G) \to V(H)$ (a.k.a H-coloring) that minimizes $\sum_{v \in V(G)} c(v, f(v))$. The complexity of *exact minimization* of this problem is well understood [35], and the class of digraphs H, for which the MinHOM(H) is polynomial time solvable is a small subset of all digraphs.

In this paper, we consider the approximation of MinHOM within a constant factor. In terms of digraphs, MinHOM(H) is not approximable if H contains a *digraph asteroidal triple (DAT)*. We take a major step toward a dichotomy classification of approximable cases. We give a dichotomy classification for approximating the MinHOM(H) when H is a graph (i.e. symmetric digraph). For digraphs, we provide constant factor approximation algorithms for two important classes of digraphs, namely bi-arc digraphs (digraphs with a *conservative semi-lattice polymorphism* or *min-ordering*), and k-arc digraphs (digraphs with an *extended min-ordering*). Specifically, we show that:

- **Dichotomy for Graphs:** MinHOM(H) has a $2|V(H)|$-approximation algorithm if graph H admits a *conservative majority polymorphims* (i.e. H is a *bi-arc graph*), otherwise, it is inapproximable;
- MinHOM(H) has a $|V(H)|^2$-approximation algorithm if H is a bi-arc digraph;
- MinHOM(H) has a $|V(H)|^2$-approximation algorithm if H is a k-arc digraph.

In conclusion, we show the importance of these results and provide insights for achieving a dichotomy classification of approximable cases. Our constant factors depend on the size of H. However, the implementation of our algorithms provides a much better approximation ratio. It leaves open to investigate a classification of digraphs H, where MinHOM(H) admits a constant factor approximation algorithm that is independent of $|V(H)|$.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Approximation algorithms, minimum cost homomorphism, randomized rounding

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.91

Category Track A: Algorithms, Complexity and Games

Related Version Full version hosted on `https://arxiv.org/abs/1902.02201`.

Funding *Arash Rafiey*: Supported by NSF 1751765.

Acknowledgements We are thankful to Andrei Bulatov for proofreading several drafts of the work and many valuable discussions that significantly improved the paper and its presentation.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 91; pp. 91:1–91:16

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

1 Introduction

For a digraph D, let $V(D)$ denote the vertex set of D, and let $A(D)$ denote the arcs of D. We denote the number of vertices of D by $|D|$. Instead of $(u,v) \in A(D)$, we use the shorthand $uv \in A(D)$ or simply $uv \in D$. A graph G is a symmetric digraph, that is, $xy \in A(G)$ if and only if (iff) $yx \in A(G)$. An edge is just a symmetric arc.

A *homomorphism* of a digraph D to a digraph H (a.k.a H-COLORING) is a mapping $f : V(D) \to V(H)$ such that for each arc xy of D, $f(x)f(y)$ is an arc of H. We say mapping f does not satisfy arc xy, if $f(x)f(y)$ is not an arc of H. The homomorphism problem for a fixed target digraph H, HOM(H), takes a digraph D as input and asks whether there is a homomorphism from D to H. Therefore, by fixing the digraph H we obtain a class of problems, one problem for each digraph D. For example, HOM(H), when H is an edge, is exactly the problem of determining whether the input graph G is bipartite (i.e., the 2-COLORING problem). Similarly, if $V(H) = \{u,v,x\}, A(H) = \{uv, vu, vx, xv, ux, xu\}$, then HOM($H$) is exactly the classical 3-COLORING problem. More generally, if H is a clique on k vertices, then HOM(H) is the k-COLORING problem. The H-COLORING problem can be considered within a more general framework, the constraint satisfaction problem (CSP). In the CSP associated with a finite relational structure $\mathbb{H}$, CSP($\mathbb{H}$), the question is whether there exists a homomorphism of a given finite relational structure to $\mathbb{H}$. Thus, the H-COLORING problem is a particular case of the CSP in which the involved relational structures are digraphs. A celebrated result due to Hell and Nesetril [31], states that, for graph H, HOM(H) is in P if H is bipartite or contains a looped vertex, and that it is NP-complete for all other graphs H. See [9] for an algebraic proof of the same result, and [12, 55] for a dichotomy for CSP($\mathbb{H}$).

There are several natural optimization versions of the HOM(H) problem. One is to find a mapping $f : V(D) \to V(H)$ that maximizes (minimizes) number of satisfied (unsatisfied) arcs in D. This problem is known under the name of MAX 2-CSP (MIN 2-CSP). For example, the most basic BOOLEAN MAX 2-CSP problem is MAX CUT where the target graph H is an edge. This line of research has received a lot of attention in the literature and there are very strong results concerning various aspects of approximability MAX 2-CSP and MIN 2-CSP [2, 22, 28, 41, 45]. See [47] for a recent survey on this and approximation of MAX k-CSP and MIN k-CSP. We consider another natural optimization version of the HOM(H) problem, i.e., we are not only interested in the existence of a homomorphism, but want to find the "best homomorphism". The *minimum cost homomorphism problem* to H, denoted by MinHOM(H), for a given input digraph D, and a cost function $c(x,i), x \in V(D), i \in V(H)$, seeks a homomorphism f of D to H that minimizes the total cost $\sum_{x \in V(D)} c(x, f(x))$. The cost function c can take non-negative rational values and positive infinity, that is $c : V(D) \times V(H) \to \mathbb{Q}_{\geq 0} \cup \{+\infty\}$. The MinHOM was introduced in [25], where it was motivated by a real-world problem in defence logistics. The MinHOM problem offers a natural and practical way to model and generalizes many optimization problems.

▶ **Example 1** ((WEIGHTED) MINIMUM VERTEX COVER)**.** This problem can be seen as MinHOM(H) where $V(H) = \{0,1\}, E(H) = \{11, 01\}$ and $c(u,0) = 0$, $c(u,1) > 0$ for every $u \in V(G)$. Note that G and H are graphs.

▶ **Example 2** (LIST HOMOMORPHISM (LHOM))**.** LHOM(H), seeks, for a given input digraph D and lists $L(x) \subseteq V(H), x \in V(D)$, a homomorphism f from D to H such that $f(x) \in L(x)$ for all $x \in V(D)$. This is equivalent to MinHOM(H) with $c(u,i) = 0$ if $i \in L(u)$, otherwise $c(u,i) = +\infty$. This problem is also known as LIST H-COLORING and its complexity is fully understood due to series of results [5, 8, 10, 11, 18, 33].

The MinHOM problem generalizes many other problems such as (Weighted) Min Ones [1, 15, 40], Min Sol [39, 53], a large class of bounded integer linear programs, retraction problems [19], Minimum Sum Coloring [4, 21, 44], and various optimum cost chromatic partition problems [27, 37, 38, 43].

A special case of MinHOM problem is where the cost function c is choosen from a fixed set Δ. This problem is denoted by MinHOM(H, Δ) [14, 53, 54]. The Valued Constrained Satisfaction Problems (VCSPs) is a generalization of this special case of the MinHOM problem. An instance of the VCSP is given by a collection of variables that must be assigned labels from a given domain with the goal to minimize the objective function that is given by the sum of cost functions, each depending on some subset of the variables [13]. Interestingly, a recent work by Cohen *et al.* [14] proved that VCSPs over a fixed valued constraint language are polynomial-time equivalent to MinHOM(H, Δ) over a fixed digraph and a proper choice of Δ.

Exact Minimization. The complexity of *exact minimization* of MinHOM(H) was studied in a series of papers, and complete complexity classifications were given in [23] for undirected graphs, in [35] for digraphs, and in [51] for more general structures. Certain minimum cost homomorphism problems have polynomial time algorithms [23, 24, 25, 35], but most are NP-hard. We remark that, the complexity of *exact minimization* of VCSPs is well understood [42, 52].

Approximation. For a minimization problem, an *α-approximation* algorithm is a (randomized) polynomial-time algorithm that finds an approximate solution of cost at most α times the minimum cost. A constant ratio approximation algorithm is an α-approximation algorithm for some constant α. We say a problem is not approximable if there is no polynomial time approximation algorithm with a multiplicative guarantee unless P = NP. The approximability of MinHOM is fairly understood when we restrict the cost function to a fixed set Δ, and further, we restrict it to take only finite values (not ∞). This setting is a special case of finite VCSPs, and there are strong approximation results on finite VCSPs. For finite VCSPs, Raghavendra [50] showed how to use the basic SDP relaxation to obtain a constant approximation. Moreover, he proved that the approximation ratio cannot be improved under Unique Game Conjecture (UGC). This constant is not explicit, but there is an algorithm that can compute it with any given accuracy in doubly exponential time. In another line of research, the power of so-called *basic linear program (BLP)* concerning constant factor approximation of finite VCSPs has been recently studied in [16, 17]. However, the approximability of VCSPs for constraint languages that are not finite-valued remains poorly understood, and [30, 39] are the only results on approximation of VCSP for languages that have cost functions that can take infinite values.

Hell *et al.,* [30] proved a dichotomy for approximating MinHOM(H) when H is a bipartite graph by transforming the MinHOM(H) to a linear program, and rounding the fractional values to get a homomorphism to H.

▶ **Theorem 3** (Dichotomy for bipartite graphs [30]). *For a fixed bipartite graph H, MinHOM(H) admits a constant factor approximation algorithm if H admits a* min-ordering *(complement of H is a* circular arc *graph), otherwise MinHOM(H) is not approximable unless $P = NP$.*

Beyond this, there is no result concerning the approximation of MinHOM(H). We go beyond bipartite case and present a constant factor approximation algorithm for *bi-arc* graphs (graphs with a *conservative majority* polymorphism). Designing an approximation

algorithm for MinHOM(H) when H is a digraph is much more complex than when H is a graph. We improve state-of-the-art by providing constant factor approximation algorithms for MinHOM(H) where H belongs to these two important cases of digraphs, namely *bi-arc* digraphs (digraphs with a *conservative semi-lattice* polymorphism a.k.a min-ordering), and *k-arc* digraphs (digraphs with a k-min-ordering). To do so, we introduce new LPs that reflect the structural properties of the target (di)graph H as well as new methods to round the fractional solutions and obtain homomorphisms to H. We will show our randomized rounding procedure can be de-randomized, and hence, we get a deterministic polynomial algorithm. Furthermore, we argue that our techniques can be used towards finding a dichotomy for the approximation of MinHOM(H).

1.1 Our Contributions

Most of the minimum cost homomorphism problems are NP-hard, therefore we investigate the approximation of MinHoM(H).

Approximating Minimum Cost Homomorphism to Digraph H.
Input: A digraph D and a vertex-mapping costs $c(x,u), x \in V(D), u \in V(H)$,
Output: A homomorphism f of D to H with the total cost of $\sum_{x \in V(D)} c(x, f(x)) \leq \alpha \cdot OPT$, where α is a constant.

Here, OPT denotes the cost of a minimum cost homomorphism of D to H. Moreover, we assume size of H is constant. Recall that we approximate the cost over real homomorphisms, rather than approximating the maximum weight of satisfied constraints, as in, say, Max Csp. One can show that if LHOM(H) is not polynomial time solvable then there is no approximation algorithm for MinHOM(H) [30, 48].

▶ **Observation 4.** *If LHOM(H) is not polynomial time solvable, then there is no approximation algorithm for MinHOM(H).*

The complexity of the LHOM problems for graphs, digraphs, and relational structures (with arity two and higher) have been classified in [18, 33, 10] respectively. LHOM(H) is polynomial time solvable if the digraph H does not contain a *digraph asteroidal triple (DAT)*[1] as an induced sub-digraph, and NP-complete when H contains a DAT [33].

MinHOM(H) is polynomial time solvable when digraph H admits a k-min-max-ordering, a subclass of DAT-free digraphs, and otherwise, NP-complete [35, 34]. Here, in this paper, we take an important step towards closing the gap between DAT-free digraphs and the one that admit a k-min-max-ordering. First, we consider digraphs that admit a min-ordering. Digraphs that admit a min-ordering have been studied under the name of *bi-arc* digraphs [36] and *signed interval* digraphs [29]. Deciding if digraph H has a min-ordering and finding a min-ordering of H is in P [36]. We provide a constant factor approximation algorithm for MinHOM(H) where H admits a min-ordering.

▶ **Theorem 5** (Digraphs with a min-ordering)**.** *If digraph H admits a min-ordering, then MinHOM(H) has a constant factor approximation algorithm.*

Sections 4,5 are dedicated to the proof of Theorem 5. In section 6, we turn our attention to digraphs with k-min-orderings, for integer $k > 1$. They are also called digraphs with *extended X-underbar* [3, 26, 46]. It was shown in [26] that if H has the X-underbar property,

[1] The definition of DAT is rather technical and it is not necessary to fully understand it in this paper.

then the HOM(H) problem is polynomial time solvable. In Lemma 21, we show that if H admits a k-min-ordering, then H is a DAT-free digraph, and provide a simple proof that LHOM(H) is polynomial time solvable. Finally, we have the following theorem.

▶ **Theorem 6** (Digraphs with a k-min-ordering)**.** *If digraph H admits a k-min-ordering for some integer $k > 1$, then MinHOM(H) has a constant factor approximation algorithm.*

Considering graphs, Feder *et al.,* [18] proved that LHOM(H) is polynomial time solvable if H is a *bi-arc* graph, and is NP-complete otherwise. In the same paper, they showed graph H is a bi-arc graph iff it admits a conservative majority polymorphism. In Section 7, we show that the same dichotomy classification holds in terms of approximation.

▶ **Theorem 7** (Dichotomy for graphs)**.** *There exists a constant factor approximation algorithm for MinHOM(H) if H is a bi-arc graph, otherwise MinHOM(H) is inapproximable.*

In section 8, we give a concrete plan of how to solve the general case. By combining the approach for obtaining the dichotomy in the graph case, together with the idea of getting an approximation algorithm for digraphs admitting a min-ordering, we might be able to achieve a constant factor approximation algorithm for MinHOM(H) when H is DAT-free.

Our constant factors depend on the size of H. However, the implementation of the LP and the ILP would yield a small integrality gap (details in the full version [49]). This indicates perhaps a better analysis of the performance of our algorithm is possible.

▶ **Open Problem 8.** *For which digraphs MinHOM(H) is approximable within a constant factor independent of size of H?*

2 Preliminaries and Definitions

Complexity and approximation of the minimum cost homomorphism problems, and in general the constraint satisfaction problems, are often studied under the existence of *polymorphisms* [6]. A polymorphism of H of arity k is a mapping f from the set of k-tuples over $V(H)$ to $V(H)$ such that if $x_iy_i \in A(H)$ for $i = 1, 2, \ldots, k$, then $f(x_1, x_2, \ldots, x_k)f(y_1, y_2, \ldots, y_k) \in A(H)$. If f is a polymorphism of H we also say that H admits f. A polymorphism f is *idempotent* if it satisfies $f(x, x, \ldots, x) = x$ for all $x \in V(H)$, and is *conservative* if $f(x_1, x_2, \ldots, x_k) \in \{x_1, x_2, \ldots, x_k\}$. A conservative *semi-lattice* polymorphism is a conservative binary polymorphism that is associative, idempotent, commutative. A conservative *majority* polymorphism μ of H is a conservative ternary polymorphism such that $\mu(x, x, y) = \mu(x, y, x) = \mu(y, x, x) = x$ for all $x, y \in V(H)$.

A conservative semi-lattice polymorphism of H naturally defines a binary relation $x \leq y$ on the vertices of H by $x \leq y$ iff $f(x, y) = x$; by associative, the relation $\leq$ is a linear order on $V(H)$, which we call a *min-ordering* of H.

▶ **Definition 9.** *The ordering $v_1 < v_2 < \cdots < v_n$ of $V(H)$ is a*

- min-ordering *iff $uv \in A(H), u'v' \in A(H)$ and $u < u', v' < v$ implies that $uv' \in A(H)$;*
- max-ordering *iff $uv \in A(H), u'v' \in A(H)$ and $u < u', v' < v$ implies that $u'v \in A(H)$;*
- min-max-ordering *iff $uv \in A(H), u'v' \in A(H)$ and $u < u', v' < v$ implies that $uv', u'v \in A(H)$.*

For bipartite graph $H = (B, W)$ let $\overrightarrow{H}$ be the digraph obtained by orienting all the edges of H from B to W. If $\overrightarrow{H}$ admits a min-ordering then we say H admits a min-ordering. It is worth mentioning that, a bipartite graph H admits a conservative majority, iff it admits a min-ordering [30]. Moreover, the complement of H is a circular arc graphs with clique cover two [18].

▶ **Definition 10.** *Let $H = (V, E)$ be a digraph that admits a homomorphism $f : V(H) \to \overrightarrow{C_k}$ (here $\overrightarrow{C_k}$ is the induced directed cycle on $\{0, 1, 2, \ldots, k-1\}$(i.e., arc set $\{(01, 12, 23, ..., (k-2)(k-1), (k-1)0\}$). Let $V_i = f^{-1}(i)$, $0 \leq i \leq k-1$.*

- *A* k-min-ordering *of H is a linear ordering $<$ of the vertices of H, so that $<$ is a min-ordering on the subgraph induced by any two circularly consecutive V_i, V_{i+1} (subscript addition modulo k).*
- *A* k-min-max-ordering *of H is a linear ordering $<$ of the vertices of H, so that $<$ is a min-max-ordering on the subgraph induced by any two circularly consecutive V_i, V_{i+1} (subscript addition modulo k).*

3 LP for Digraphs with a min-max-ordering

Before presenting the LP, we give a procedure to modify lists associated to the vertices of D. To each vertex $x \in D$, associate a list $L(x)$ that initially contains $V(H)$. Think of $L(x)$ as the set of possible images for x in a homomorphism from D to H. Apply the *arc consistency* procedure as follows. Take an arbitrary arc $xy \in A(D)$ ($yx \in A(D)$) and let $a \in L(x)$. If there is no out-neighbor (in-neighbor) of a in $L(y)$ then remove a from $L(x)$. Repeat this until a list becomes empty or no more changes can be made. Note that if we end up with an empty list after arc consistency then there is no homomorphism of D to H.

Let $a_1, a_2, a_3, \ldots, a_p$ be a min-max-ordering $<$ of the target digraph H. Define $\ell^+(i)$ to be the smallest subscript j such that a_j is an out-neighbor of a_i (and $\ell^-(i)$ to be the smallest subscript j such that a_j is an in-neighbor of a_i).

Consider the following linear program. For every vertex v of D and every vertex a_i of H define variable v_i. Moreover, define variable v_{p+1} for every $v \in D$ whose value is set to zero.

$$\begin{array}{rll}
\min & \sum_{v,i} c(v, a_i)(v_i - v_{i+1}) & \\
\text{subject to:} & v_i \geq 0 & \text{(C1)} \\
& v_1 = 1 & \text{(C2)} \\
& v_{p+1} = 0 & \text{(C3)} \\
& v_{i+1} \leq v_i & \text{(C4)} \\
& v_{i+1} = v_i \quad \text{if } a_i \notin L(v) & \text{(C5)} \\
& u_i \leq v_{l^+(i)} \quad \forall uv \in A(D) & \text{(C6)} \\
& v_i \leq u_{l^-(i)} \quad \forall uv \in A(D) & \text{(C7)}
\end{array}$$

Let $\mathcal{S}$ denote the set of constraints of the above LP, then:

▶ **Theorem 11.** *If digraph H admits a min-max-ordering, then there is a one-to-one correspondence between homomorphisms of D to H and integer solutions of $\mathcal{S}$.*

Proof. For homomorphism $f : D \to H$, if $f(v) = a_t$ we set $v_i = 1$ for all $i \leq t$, otherwise we set $v_i = 0$. We set $v_1 = 1$ and $v_{p+1} = 0$ for all $v \in V(D)$. Now all the variables are nonnegative and we have $v_{i+1} \leq v_i$. Note that if $a_i \notin L(v)$ then $f(v) \neq a_i$ and we have $v_i - v_{i+1} = 0$. It remains to show that $u_i \leq v_{l^+(i)}$ for every uv arc in D. Suppose for contradiction that $u_i = 1$ and $v_{l^+(i)} = 0$ and let $f(u) = a_r$ and $f(v) = a_s$. This implies that $u_r = 1$, whence $i \leq r$; and $v_s = 1$, whence $s < l^+(i)$. Since $a_i a_{l^+(i)}$ and $a_r a_s$ both are arcs of H with $i \leq r$ and $s < l^+(i)$, the fact that H has a min-ordering implies that $a_i a_s$ must also be an arc of H, contradicting the definition of $l^+(i)$. The proof for $v_i \leq u_{l^-(i)}$ is analogous.

Conversely, if there is an integer solution for $\mathcal{S}$, we define a homomorphism f as follows: we let $f(v) = a_i$ when i is the largest subscript with $v_i = 1$. We prove that this is indeed a homomorphism by showing that every arc of D is mapped to an arc of H. Let uv be an arc

of D and assume $f(u) = a_r$, $f(v) = a_s$. We show that $a_r a_s$ is an arc in H. Observe that $1 = u_r \leq v_{l^+(r)} \leq 1$ and $1 = v_s \leq u_{l^-(s)} \leq 1$, therefore we must have $v_{l^+(r)} = u_{l^-(s)} = 1$. Since r and s are the largest subscripts such that $u_r = v_s = 1$ then $l^+(r) \leq s$ and $l^-(s) \leq r$. Since $a_r a_{l^+(r)}$ and $a_{l^-(s)} a_s$ are arcs of H, we must have the arc $a_r a_s$, as H admits a max-ordering. Furthermore, $f(v) = a_i$ iff $v_i = 1$ and $v_{i+1} = 0$, so, $c(v, a_i)$ contributes to the sum iff $f(v) = a_i$. ◀

We have translated the minimum cost homomorphism problem to a linear program. In fact, this linear program corresponds to a minimum cut problem in an auxiliary network, and can be solved by network flow algorithms [23, 48]. In [30], a similar result to Theorem 11 was proved for the MinHOM(H) problem on undirected graphs when target graph H is bipartite and admits a min-max-ordering. We shall enhance the above system $\mathcal{S}$ to obtain an approximation algorithm for the case where H is only assumed to admit a min-ordering.

4 LP for Digraphs with a min-ordering

In the rest of the paper assume lists are not empty. Moreover, non-empty lists guarantee a homomorphism when H admits a min-ordering.

▶ **Lemma 12.** *[32] Let H be a digraph that admits a min-ordering. If all the lists are non-empty after arc consistency, then there exists a homomorphism from D to H.*

Suppose $a_1, a_2, \ldots, a_p$ is a min-ordering of H. Let E' denote the set of all the pairs (a_i, a_j) such that $a_i a_j$ is not an arc of H, but there is an arc $a_i a_{j'}$ of H with $j' < j$ and an arc $a_{i'} a_j$ of H with $i' < i$. Let $E = A(H)$ and define H' to be the digraph with vertex set $V(H)$ and arc set $E \cup E'$. Note that E and E' are disjoint sets.

▶ **Observation 13.** *The ordering $a_1, a_2, \cdots, a_p$ is a min-max-ordering of H'.*

▶ **Observation 14.** *Let $e = a_i a_j \in E'$. Then a_i does not have any out-neighbor in H after a_j, or a_j does not have any in-neighbor in H after a_i.*

Observation 14 easily follows from the fact that H has a min-ordering. Since H' has a min-max-ordering, we can form system of linear inequalities $\mathcal{S}$, for H' as described in Section 3. Homomorphisms of D to H' are in a one-to-one correspondence with integer solutions of $\mathcal{S}$, by Theorem 11. However, we are interested in homomorphisms of D to H, not H'. Therefore, we shall add further inequalities to $\mathcal{S}$ to ensure that we only admit homomorphisms from D to H, i.e., avoid mapping arcs of D to the arcs in E'.

For every arc $e = a_i a_j \in E'$ and every arc $uv \in A(D)$, by Observation 14, two of the following set of inequalities will be added to $\mathcal{S}$ (i.e. either (C8), (C11) or (C9), (C10) or (C9), (C11)).

$$v_j \leq u_s + \sum_{t<i, a_t a_j \in E, a_t \in L(u)} (u_t - u_{t+1}) \quad \text{if } a_s \in L(u) \text{ is the first in-neighbour of } a_j \text{ after } a_i \qquad \text{(C8)}$$

$$v_j \leq v_{j+1} + \sum_{t<i, a_t a_j \in E, a_t \in L(u)} (u_t - u_{t+1}) \quad \text{if } a_j \text{ has no in-neighbour after } a_i \qquad \text{(C9)}$$

$$u_i \leq v_s + \sum_{t<j, a_i a_t \in E, a_t \in L(v)} (v_t - v_{t+1}) \quad \text{if } a_s \in L(v) \text{ is the first out-neighbour of } a_i \text{ after } a_j \qquad \text{(C10)}$$

$$u_i \leq u_{i+1} + \sum_{t<j, a_i a_t \in E, a_t \in L(v)} (v_t - v_{t+1}) \quad \text{if } a_i \text{ has no out-neighbour after } a_j \qquad \text{(C11)}$$

Additionally, for every pair $(x, y) \in V(D) \times V(D)$ consider a list $L(x, y)$ of possible pairs (a, b), $a \in L(x)$ and $b \in L(y)$. Perform *pair consistency* procedure as follows. Consider three vertices $x, y, z \in V(D)$. For $(a, b) \in L(x, y)$ if there is no $c \in L(z)$ such that $(a, c) \in L(x, z)$

and $(c, b) \in L(z, y)$ then remove (a, b) from $L(x, y)$. Repeat this until a pair list becomes empty or no more changes can be made. Here, we assume that after pair consistency procedure no pair list is empty, as otherwise there is no homomorphism of D to H. Therefore, by pair consistency, add the following constraints for every u, v in $V(D)$ and $a_i \in L(u)$:

$$u_i - u_{i+1} \quad \leq \quad \sum_{\substack{j: \\ (a_i, a_j) \in L(u,v)}} (v_j - v_{j+1}) \qquad \text{(C12)}$$

▶ **Lemma 15.** *If H admits a min-ordering, then there is a one-to-one correspondence between homomorphisms of D to H and integer solutions of the extended system $\mathcal{S}$.*

5 Approximation for Digraphs with a min-ordering

In what follows, we describe an overview of our approximation algorithm for MinHOM(H) where the fixed digraph H has a min-ordering. We encourage the reader to see Algorithm 1 while reading this section. An overview of the proofs of the correctness and approximation bound are postponed for the later subsections (further details in the full version [49]).

Let D be the input digraph together with a cost function c. Let $a_1, \ldots, a_p$ be a min ordering of the vertices of H. The algorithm, first constructs digraph H' from H as in Section 4. By Observation 13, $a_1, \ldots, a_p$ is a min-max-ordering for H'. By Lemma 15, the integral solutions of the extended LP are in one-to-one correspondence to homomorphisms from D to H. At this point, our algorithm will minimize the cost function over extended $\mathcal{S}$ in polynomial time using a linear programming algorithm. This will generally result in a fractional solution (Even though the original system $\mathcal{S}$ is known to be totally unimodular [23, 48] and hence have integral optima, we have added inequalities, and hence lost this advantage). We will obtain an integer solution by a randomized procedure called *rounding*. Choose, uniformly at random, a random variable $X \in [0, 1]$, and define the rounded values $u'_i = 1$ when $u_i \geq X$ (u_i is the returned value by the LP), and $u'_i = 0$ otherwise. It is easy to check that the rounded values satisfy the original inequalities, i.e., correspond to a homomorphism f of D to H'.

Now the algorithm will once more modify the solution f to become a homomorphism from D to H, i.e., to avoid mapping the arcs of D to the arcs in E'. This will be accomplished by another randomized procedure, which we call Shift. We choose, uniformly at random, another random variable $Y \in [0, 1]$, which will guide the shifting. Let F denote the set of all arcs in E' to which some arcs of D are mapped by f. If F is empty, we need no shifting. Otherwise, let $a_i a_j$ be an arc of F. Since $F \subseteq E'$, Observation 14 implies that either a_j has no in-neighbor after a_i or a_i has no out-neighbor after a_j. Suppose the first case happens (the shifting process is similar in the other case).

Consider a vertex v in D such that $f(v) = a_j$ (i.e. $v'_j = 1$ and $v'_{j+1} = 0$) and v has an in-neighbor u in D with $f(u) = a_i$ (i.e. $u'_i = 1$ and $u'_{i+1} = 0$). For such a vertex v, let $S_v = \{a_{t_1}, a_{t_2}, \ldots, a_{t_k}\}$ be the set of all vertices a_t with $t < j$ such that $a_i a_t \in E$ and $a_t \in L(v)$. Suppose S_v consists of a_t with subscripts t ordered as $t_1 < t_2 < \cdots < t_k$.

▶ **Lemma 16.** *During procedure Shift, the set of indices $t_1 < \cdots < t_k$ considered in Line 6 of the Algorithm 1 is non-empty.*

By Lemma 16, S_v is not empty. The algorithm now selects one vertex from this set as follows. Let $P_{v,t} = \frac{v_t - v_{t+1}}{P_v}$, where $P_v = \sum_{\substack{t<j \\ a_i a_t \in E, a_t \in L(v)}} (v_t - v_{t+1})$.

Note that $P_v > 0$ because of constraints (C9) and (C10). Then a_{t_q} is selected if $\sum_{p=1}^{q} P_{v,t_p} < Y \leq \sum_{p=1}^{q+1} P_{v,t_p}$. Thus a concrete a_t is selected with probability $P_{v,t}$, which is

proportional to the difference of the fractional values $v_t - v_{t+1}$. When the selected vertex is a_t, we shift the image of the vertex v from a_j to a_t, and set $v'_r = 1$ if $r \leq t$, else set $v'_r = 0$. Note that a_t is before a_j in the min-ordering. Now we might need to shift images of the neighbors of v. In this case, repeat the shifting procedure for neighbors of v. This processes continues in a Breadth-first search (BFS) like manner, until no more shift is required (Figure 1 gives an illustration). Note that a vertex might be visited multiple times in procedure shift while a pair $(v, a_i) \in V(D) \times V(H)$ is considered at most one time.

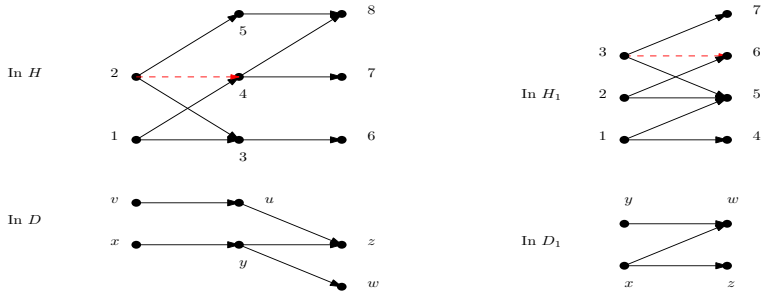


Figure 1 Two examples. In the right example, the target digraph is H_1 and input is D_1. Digraphs D_1 and H_1 both can be view as bipartite graphs and $1, 2, 3, 4, 5, 6, 7$ is a min-ordering of H. When x, y are mapped to 3 and w is mapped to 6 then the algorithm should shift the image of w from 6 to 5 and since 35 is an arc there is no need to shift the image of y. In the left example, the target digraph is H and the input is D. $1, 2, 3, 4, 5, 6, 7, 8$ is a min-ordering of H and 24 is a missing arc. Suppose x is mapped to 2, y to 4, w to 7, z to 8, u to 5 and v to 2. Then we should shift the image of y to 3 and then w to 6 and z to 6 and then u to 3 and v to one of the $1, 2$.

We remark that the images of vertices in D are always shifted towards smaller elements in their lists. Lemma 17 shows that this shifting modifies the homomorphism f, and hence, the corresponding values of the variables. Namely, $v'_{t+1}, \ldots, v'_j$ are reset to 0, keeping all other values the same. Note that these modified values still satisfy the original set of constraints $\mathcal{S}$, i.e., the modified mapping is still a homomorphism.

▶ **Lemma 17.** *Procedure* SHIFT*, in polynomial time, returns a homomorphism of D to H'.*

We repeat the same process for the next v with these properties, until no edge of D is mapped to an edge in E'. Each iteration involves at most $|V(H)| \cdot |V(D)|$ shifts. After at most $|E'|$ iterations, no edge of D is mapped to an edge in F and we no longer need to shift. Next theorem follows from Lemma 16 and 17.

▶ **Theorem 18.** *Our algorithm, in polynomial time, returns a homomorphism of D to H.*

5.1 Analyzing the Approximation Ratio

We now claim that the cost of this homomorphism is at most $|V(H)|^2$ times the minimum cost of a homomorphism. Let w denote the value of the objective function with the fractional optimum u_i, v_j, and w' denote the value of the objective function with the final values u'_i, v'_j, after the rounding and all the shifting. Also, let w^* be the minimum cost of a homomorphism of D to H. Obviously, $w \leq w^* \leq w'$.

We now show that the expected value of w' is at most a constant times w. Let us focus on the contribution of one summand, say $v'_t - v'_{t+1}$, to the calculation of the cost. In any integer solution, $v'_t - v'_{t+1}$ is either 0 or 1. The probability that $v'_t - v'_{t+1}$ contributes to w' is the probability of the event that $v'_t = 1$ and $v'_{t+1} = 0$. This can happen in the following:

Algorithm 1 Approximation MinHOM(H).

procedure Approx–MinHOM(D, H)
  Construct H' from H (as in Section 3)
  Let u_is be the (fractional) values returned by the extended LP
  Choose a random variable $X \in [0,1]$, and $\forall u_i$s : if $X \leq u_i$ let $u'_i = 1$, else let $u'_i = 0$
  Let $f(u) = a_i$ where i is the largest subscript with $u'_i = 1$ ▷ f is a homomorphism from D to H'
  Choose a random variable $Y \in [0,1]$
  while $\exists uv \in A(D)$ such that $f(u)f(v) \in A(H') \setminus A(H)$ **do**
    if $f(v)$ does not have an in-neighbor after $f(u)$ **then** Shift(f, v)
    else if $f(u)$ does not have an out-neighbor after $f(v)$ **then** Shift(f, u)
  return f ▷ f is a homomorphism from D to H

Algorithm 2 The Shifting Procedure.

procedure Shift(f, x)
  Let Q be a Queue, $Q.enqueue(x)$
  while Q is not empty **do**
    $v \leftarrow Q.dequeue()$
    for $uv \in A(D)$ with $f(u)f(v) \notin A(H)$ **or** $vu \in A(D)$ with $f(v)f(u) \notin A(H)$ **do**
      ▷ Here we assume the first condition holds, the other case is similar
      ▷ Further, we assume $f(v)$ does not have an in-neighbor after $f(u)$
      Let $t_1 < \cdots < t_k$ be indices so that $a_{t_j} < f(v), a_{t_j} \in L(v), f(u)a_{t_j} \in A(H)$
      Let $P_v \leftarrow \sum_{j=1}^{j=k} (v_{t_j} - v_{t_j+1})$ and $P_{v,t} \leftarrow (v_t - v_{t+1}) \,/\, P_v$
      if $\sum_{p=1}^{q} P_{v,t_p} < Y \leq \sum_{p=1}^{q+1} P_{v,t_p}$ **then**
        $f(v) \leftarrow a_{t_q}$, set $v'_i = 1$ for $1 \leq i \leq t_q$, and set $v'_i = 0$ for $t_p < i$
      for $vz \in A(D)$ $(zv \in A(D))$ with $f(v)f(z) \notin A(H)$ $(f(z)f(v) \notin A(H))$ **do**
        $Q.enqueue(z)$
  return f ▷ f is a homomorphism from D to H'

1. v is mapped to a_t by rounding, and is not shifted away. In other words, we have $v'_t = 1$ and $v'_{t+1} = 0$ after rounding, and these values don't change by procedure Shift.
2. v is first mapped to some $a_j, j > t$, by rounding, and then re-mapped to a_t by procedure Shift.

▶ **Lemma 19.** *The expected contribution of one summand, say $v'_t - v'_{t+1}$, to the expected cost of w' is at most $|V(H)|^2 c(v, a_t)(v_t - v_{t+1})$.*

▶ **Theorem 20.** *Algorithm 1 returns a homomorphism with expected cost $|V(H)|^2 \cdot OPT$. The algorithm can be de-randomized to obtain a deterministic $|V(H)|^2$-approximation algorithm.*

6 Approximation for Digraphs with a k-min-ordering

Digraphs admitting k-min-ordering ($k > 1$) do not admit a min-ordering or a conservative majority polymorphism. However, this does not rule out the possibility of a constant factor approximation algorithm. We show that they are in fact DAT-free digraphs, and they admit a nice geometric representation (see the full version [49]).

▶ **Lemma 21.** *Let H be a digraph that admits a k-min-ordering. Then H is DAT-free, and LHOM(H) is polynomial time solvable.*

Let H be a digraph with a k-min-ordering ($k > 1$) and partition $V_0, V_1, \ldots, V_{k-1}$ of its vertices, and let $<$ be a k-min-ordering of $V(H)$. It is easy to argue that the input digraph D must be homomorphic to $\overrightarrow{C_k}$ otherwise there is no homomorphism from D to H. Therefore, we assume (for some $0 \leq \ell \leq k-1$), $L(u) \subseteq V_i$ for every $u \in U_{i+\ell}$, $0 \leq i \leq k-1$. Now the LP is designed according to the lists L. Since $<$ is a min-ordering of $V_i \cup V_{i+1}$, the constraints are very similar to the ones in Section 3. The conclusion of this section is the following:

▶ **Theorem 22.** *There is a (deterministic) $|V(H)|^2$-approximation algorithm for MinHOM(H) when the target digraph H admits a k-min-ordering, $k > 1$.*

7 A Dichotomy for Graphs

Feder and Vardi [20] proved that if a graph H admits a conservative majority polymorphism, then LHOM(H) is polynomial time solvable. Later, Feder *et al.,* [18] showed that LHOM(H) is polynomial time solvable iff H is a *bi-arc* graph. Hence, by Observation 4, the problem is inapproximable beyond bi-arc graphs. A bi-arc graph is represented by a pair of families of arcs on a circle with specific conditions (exact definition is given in the full version [49]). Note that in a bi-arc graph a vertex may have a self-loop.

▶ **Theorem 23** ([7, 18])**.** *A graph admits a conservative majority polymorphism iff it is a bi-arc graph.*

▶ **Definition 24** (G^*)**.** *Let $G = (V, E)$ be a graph. Let G^* be a bipartite graph with partite sets V, V' where V' is a copy of V. Two vertices $u \in V$, and $v' \in V'$ of G^* are adjacent in G^* iff uv is an edge of G.*

A *circular arc* graph is a graph that is the intersection graph of a family of arcs on a circle. A bipartite graph whose complement is a circular arc graph, is called a *co-circular arc* graph. Note that co-circular arc graphs are irreflexive, meaning no vertex has a loop.

▶ **Lemma 25.** *Let H^* be the bipartite graph constructed from a bi-arc graph H. Then H^* is a co-circular arc graph and H^* admits a min-ordering.*

Let H be a bi-arc graph, with vertex set I, and let $H^* = (I, I')$ be the bipartite graph constructed from H. Let $a_1, a_2, \ldots, a_p$ be an ordering of the vertices in I and $b_1, b_2, \ldots, b_p$ be an ordering of the vertices of I'. Note that each a_i has a copy $b_{\pi(i)}$ in $\{b_1, b_2, \ldots, b_n\}$ where π is a permutation on $\{1, 2, 3, \ldots, p\}$. By Lemma 25, let us assume $a_1, a_2, \ldots, a_p, b_1, b_2, \ldots, b_p$ is a min-ordering for H^*.

Let G be the input graph with vertex set V and a cost function c. Construct G^* from G with vertex set $V \cup V'$ as in Definition 24. Now construct an instance of the MinHOM(H^*) for the input graph G^* and set $c(v', b_{\pi(i)}) = c(v, a_i)$ for $v \in V$ and $v' \in V'$. Further, make H^* a digraph by orienting all its edges from I to I', and similarly make G^* a digraph by

orienting all its edges from V to V'. Note that, by construction of H^* and G^*, there exists a homomorphism $f : G \to H$ with cost $\mathfrak{C}$ iff there exists homomorphism $f^* : G^* \to H^*$ with cost $2\mathfrak{C}$ such that, if $f^*(v) = a_i$ then $f^*(v') = b_j$ with $j = \pi(i)$.

We first perform the arc consistency and pair consistency procedures for the vertices in G^*. Note that if $L(u)$ contains element a_i then $L(u')$ contains $b_{\pi(i)}$ and when $L(u')$ contains some b_j then $L(u)$ contains $a_{\pi^{-1}(j)}$. Next, we define the system of linear equations $\widehat{S^*}$ with the same construction as in Sections 3, 4. Further, we add the following constraint to $\widehat{S^*}$. The full set of constraints in $\widehat{S^*}$ is presented in the extended version.

$$u_i - u_{i+1} = u'_{\pi(i)} - u'_{\pi(i)+1} \qquad \forall u, u' \in G^*, \forall a_i, b_{\pi(i)} \in H^*$$

▶ **Lemma 26.** *If H is a bi-arc graph, then there is a one-to-one correspondence between homomorphisms from G to H and integer solutions of $\widehat{S^*}$.*

Once again we round an optimal fractional solution of $\widehat{S^*}$, using random variable $X \in [0, 1]$. Let $\mathcal{F}$ be a mapping form $V(G^*)$ to $V(H^*)$ obtained after rounding using X. We give an algorithm that modifies $\mathcal{F}$ and achieves a homomorphism $f : G \to H$ (i.e. an integral solution that satisfies $\widehat{S^*}$). The algorithm deploys a shifting procedure that first uses a random variable Y to shift the images of some of the vertices of $V(G^*)$ to obtain a homomorphism f from G^* to H^*. Second, it applies a breadth-first search function to make f consistent on V and V'; meaning that $f(u) = a_i$, $u \in V$ iff $f(u') = b_{\pi(i)}$, $u' \in V'$. The proof of the following theorems and de-randomization of the algorithm appear in the full version [49].

▶ **Theorem 27.** *There exists a randomized algorithm that modifies $\mathcal{F}$ and obtain a homomorphism $f : G \to H$. Moreover, the expected cost of the homomorphism returned by this algorithm is at most $2|V(H)| \cdot OPT$.*

▶ **Theorem 28.** *If H admits a conservative majority polymorphism, then MinHOM(H) has a (deterministic) $2|V(H)|$-approximation algorithm, otherwise it is inapproximable.*

8 Beyond majority and min-ordering (DAT-free cases)

This section offers a view of moving forward to get a dichotomy classification for constant approximability of MinHOM(H). We believe the class of DAT-free digraphs is the right boundary between the approximable cases, and the ones that do not admit any approximation.

▶ **Conjecture 29.** *MinHOM(H) admits a constant approximation polynomial time algorithm when H is a DAT-free digraph, otherwise, MinHOM(H) is not approximable.*

For digraph $D = (V, A)$, let D^* be a bipartite digraph with partite sets V, V' where V' is a copy of V. There is an arc in D^* from $u \in V$ to $v' \in V'$ iff uv is an arc of D. In what follows, we give a road map for solving the conjecture. Let us start off by making a connection between homomorphisms from D to a DAT-free target digraph H, and the homomorphisms from D^* to H^*.

▶ **Proposition 30.** *Let D, H be two digraphs and let D, H, L (here L are the lists) be an instance of the LHOM(H). Suppose H is DAT-free. Then H^* admits a min-ordering, and LHOM(H^*) is polynomial time solvable for instance D^*, H^* where $L^*(v') = \{a' | a \in L(v)\}$ and $L^*(v) = L(v)$ for every $v \in V(D)$.*

Similar to Lemmas 15 and 26, we can obtain set of constraints $\widehat{S^*}$ such that there is a one-to-one correspondence between homomorphisms from D to H and integer solutions of $\widehat{S^*}$ (details in the full version [49]). Our primary challenge would be finding a rounding

procedure to obtain a homomorphism from D to H. We believe there is a need to deploy the shift procedure in min-ordering case (Section 3), as well as, the shifting procedure in the majority case (Section 7). This essentially means obtaining a new way of solving a list homomorphism from D to H when H is a DAT-free, using the bi-partition method. The calculation should work out; yielding a constant bound between the fractional value of the LP and the integral value obtained by rounding. Notice that in the majority case the symmetry of the arcs is heavily used in our argument, whereas in the digraph case we no longer have this property in hand.

References

1. Sanjeev Arora, László Babai, Jacques Stern, and Z. Sweedyk. The Hardness of Approximate Optima in Lattices, Codes, and Systems of Linear Equations. *J. Comput. Syst. Sci.*, 54(2):317–331, 1997. doi:10.1006/jcss.1997.1472.
2. Per Austrin. Towards Sharp Inapproximability For Any 2-CSP. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 307–317, 2007. doi:10.1109/FOCS.2007.73.
3. Guillaume Bagan, Arnaud Durand, Emmanuel Filiot, and Olivier Gauwin. Efficient Enumeration for Conjunctive Queries over X-underbar Structures. In *Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings*, pages 80–94, 2010. doi:10.1007/978-3-642-15205-4_10.
4. Amotz Bar-Noy, Mihir Bellare, Magnús M. Halldórsson, Hadas Shachnai, and Tami Tamir. On Chromatic Sums and Distributed Resource Allocation. *Inf. Comput.*, 140(2):183–202, 1998. doi:10.1006/inco.1997.2677.
5. Libor Barto. The Dichotomy for Conservative Constraint Satisfaction Problems Revisited. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada*, pages 301–310, 2011. doi:10.1109/LICS.2011.25.
6. Libor Barto, Andrei A. Krokhin, and Ross Willard. Polymorphisms, and How to Use Them. In Andrei A. Krokhin and Stanislav Zivny, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 1–44. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/DFU.Vol7.15301.1.
7. Richard C. Brewster, Tomás Feder, Pavol Hell, Jing Huang, and Gary MacGillivray. Near-Unanimity Functions and Varieties of Reflexive Graphs. *SIAM J. Discrete Math.*, 22(3):938–960, 2008. doi:10.1137/S0895480103436748.
8. Andrei A. Bulatov. Tractable conservative Constraint Satisfaction Problems. In *18th IEEE Symposium on Logic in Computer Science (LICS 2003), 22-25 June 2003, Ottawa, Canada, Proceedings*, page 321, 2003. doi:10.1109/LICS.2003.1210072.
9. Andrei A. Bulatov. H-Coloring dichotomy revisited. *Theor. Comput. Sci.*, 349(1):31–39, 2005. doi:10.1016/j.tcs.2005.09.028.
10. Andrei A. Bulatov. Complexity of conservative constraint satisfaction problems. *ACM Trans. Comput. Log.*, 12(4):24:1–24:66, 2011. doi:10.1145/1970398.1970400.
11. Andrei A. Bulatov. Conservative constraint satisfaction re-revisited. *J. Comput. Syst. Sci.*, 82(2):347–356, 2016. doi:10.1016/j.jcss.2015.07.004.
12. Andrei A. Bulatov. A Dichotomy Theorem for Nonuniform CSPs. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 319–330, 2017. doi:10.1109/FOCS.2017.37.
13. David A. Cohen, Martin C. Cooper, Peter Jeavons, and Andrei A. Krokhin. The complexity of soft constraint satisfaction. *Artif. Intell.*, 170(11):983–1016, 2006. doi:10.1016/j.artint.2006.04.002.

14 David A. Cohen, Martin C. Cooper, Peter G. Jeavons, Andrei A. Krokhin, Robert Powell, and Stanislav Zivny. Binarisation for Valued Constraint Satisfaction Problems. *SIAM J. Discrete Math.*, 31(4):2279–2300, 2017. doi:10.1137/16M1088107.

15 Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity classifications of boolean constraint satisfaction problems*, volume 7. SIAM, 2001.

16 Víctor Dalmau, Andrei A. Krokhin, and Rajsekar Manokaran. Towards a characterization of constant-factor approximable finite-valued CSPs. *J. Comput. Syst. Sci.*, 97:14–27, 2018. doi:10.1016/j.jcss.2018.03.003.

17 Alina Ene, Jan Vondrák, and Yi Wu. Local Distribution and the Symmetry Gap: Approximability of Multiway Partitioning Problems. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 306–325, 2013. doi:10.1137/1.9781611973105.23.

18 Tomás Feder, Pavol Hell, and Jing Huang. Bi-arc graphs and the complexity of list homomorphisms. *Journal of Graph Theory*, 42(1):61–80, 2003. doi:10.1002/jgt.10073.

19 Tomás Feder, Pavol Hell, Peter Jonsson, Andrei A. Krokhin, and Gustav Nordh. Retractions to Pseudoforests. *SIAM J. Discrete Math.*, 24(1):101–112, 2010. doi:10.1137/080738866.

20 Tomás Feder and Moshe Y. Vardi. The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study through Datalog and Group Theory. *SIAM J. Comput.*, 28(1):57–104, 1998. doi:10.1137/S0097539794266766.

21 Krzysztof Giaro, Robert Janczewski, Marek Kubale, and Michal Malafiejski. A 27/26-Approximation Algorithm for the Chromatic Sum Coloring of Bipartite Graphs. In *Approximation Algorithms for Combinatorial Optimization, 5th International Workshop, APPROX 2002, Rome, Italy, September 17-21, 2002, Proceedings*, pages 135–145, 2002. doi:10.1007/3-540-45753-4_13.

22 Michel X. Goemans and David P. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. ACM*, 42(6):1115–1145, 1995. doi:10.1145/227683.227684.

23 Gregory Z. Gutin, Pavol Hell, Arash Rafiey, and Anders Yeo. A dichotomy for minimum cost graph homomorphisms. *Eur. J. Comb.*, 29(4):900–911, 2008. doi:10.1016/j.ejc.2007.11.012.

24 Gregory Z. Gutin, Arash Rafiey, and Anders Yeo. Minimum Cost Homomorphisms to Semicomplete Bipartite Digraphs. *SIAM J. Discrete Math.*, 22(4):1624–1639, 2008. doi:10.1137/060668316.

25 Gregory Z. Gutin, Arash Rafiey, Anders Yeo, and Michael Tso. Level of repair analysis and minimum cost homomorphisms of graphs. *Discrete Applied Mathematics*, 154(6):881–889, 2006. doi:10.1016/j.dam.2005.06.012.

26 Wolfgang Gutjahr, Emo Welzl, and Gerhard J. Woeginger. Polynomial graph-colorings. *Discrete Applied Mathematics*, 35(1):29–45, 1992. doi:10.1016/0166-218X(92)90294-K.

27 Magnús M. Halldórsson, Guy Kortsarz, and Hadas Shachnai. Minimizing Average Completion of Dedicated Tasks and Interval Graphs. In *Approximation, Randomization and Combinatorial Optimization: Algorithms and Techniques, 4th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2001 and 5th International Workshop on Randomization and Approximation Techniques in Computer Science, RANDOM 2001 Berkeley, CA, USA, August 18-20, 2001, Proceedings*, pages 114–126, 2001. doi:10.1007/3-540-44666-4_15.

28 Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001. doi:10.1145/502090.502098.

29 Pavol Hell, Jing Huang, Ross M. McConnell, and Arash Rafiey. Interval-Like Graphs and Digraphs. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, pages 68:1–68:13, 2018. doi:10.4230/LIPIcs.MFCS.2018.68.

30 Pavol Hell, Monaldo Mastrolilli, Mayssam Mohammadi Nevisi, and Arash Rafiey. Approximation of Minimum Cost Homomorphisms. In *Algorithms - ESA 2012 - 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings*, pages 587–598, 2012. doi:10.1007/978-3-642-33090-2_51.

31 Pavol Hell and Jaroslav Nesetril. On the complexity of H-coloring. *J. Comb. Theory, Ser. B*, 48(1):92–110, 1990. doi:10.1016/0095-8956(90)90132-J.

32 Pavol Hell and Jaroslav Nesetril. *Graphs and homomorphisms*. Oxford University Press, 2004.

33 Pavol Hell and Arash Rafiey. The Dichotomy of List Homomorphisms for Digraphs. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1703–1713, 2011. doi:10.1137/1.9781611973082.131.

34 Pavol Hell and Arash Rafiey. Monotone Proper Interval Digraphs and Min-Max Orderings. *SIAM J. Discrete Math.*, 26(4):1576–1596, 2012. doi:10.1137/100783844.

35 Pavol Hell and Arash Rafiey. The Dichotomy of Minimum Cost Homomorphism Problems for Digraphs. *SIAM J. Discrete Math.*, 26(4):1597–1608, 2012. doi:10.1137/100783856.

36 Pavol Hell and Arash Rafiey. Bi-Arc Digraphs and Conservative Polymorphisms. *CoRR*, abs/1608.03368, 2016. arXiv:1608.03368.

37 Klaus Jansen. Approximation Results for the Optimum Cost Chromatic Partition Problem. *J. Algorithms*, 34(1):54–89, 2000. doi:10.1006/jagm.1999.1022.

38 Tao Jiang and Douglas B West. Coloring of trees with minimum sum of colors. *Journal of Graph Theory*, 32(4):354–358, 1999.

39 Peter Jonsson and Gustav Nordh. Introduction to the Maximum SolutionProblem. In *Complexity of Constraints - An Overview of Current Research Themes [Result of a Dagstuhl Seminar]*., pages 255–282, 2008. doi:10.1007/978-3-540-92800-3_10.

40 Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The Approximability of Constraint Satisfaction Problems. *SIAM J. Comput.*, 30(6):1863–1920, 2000. doi:10.1137/S0097539799349948.

41 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal Inapproximability Results for Max-Cut and Other 2-Variable CSPs? In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 146–154, 2004. doi:10.1109/FOCS.2004.49.

42 Vladimir Kolmogorov, Andrei A. Krokhin, and Michal Rolinek. The Complexity of General-Valued CSPs. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1246–1258, 2015. doi:10.1109/FOCS.2015.80.

43 Leo G. Kroon, Arunabha Sen, Haiyong Deng, and Asim Roy. The Optimal Cost Chromatic Partition Problem for Trees and Interval Graphs. In *Graph-Theoretic Concepts in Computer Science, 22nd International Workshop, WG '96, Cadenabbia (Como), Italy, June 12-14, 1996, Proceedings*, pages 279–292, 1996. doi:10.1007/3-540-62559-3_23.

44 Ewa Kubicka and Allen J. Schwenk. An Introduction to Chromatic Sums. In *Computer Trends in the 1990s - Proceedings of the 1989 ACM 17th Annual Computer Science Conference, Louisville, Kentucky, USA, February 21-23, 1989*, pages 39–45, 1989. doi:10.1145/75427.75430.

45 Michael Lewin, Dror Livnat, and Uri Zwick. Improved Rounding Techniques for the MAX 2-SAT and MAX DI-CUT Problems. In *Integer Programming and Combinatorial Optimization, 9th International IPCO Conference, Cambridge, MA, USA, May 27-29, 2002, Proceedings*, pages 67–82, 2002. doi:10.1007/3-540-47867-1_6.

46 Gary MacGillivray and Jacobus Swarts. The C_k-extended graft construction. *Discrete Applied Mathematics*, 159(12):1293–1301, 2011. doi:10.1016/j.dam.2011.04.006.

47 Konstantin Makarychev and Yury Makarychev. Approximation Algorithms for CSPs. In Andrei A. Krokhin and Stanislav Zivny, editors, *The Constraint Satisfaction Problem: Complexity*

and Approximability, volume 7 of *Dagstuhl Follow-Ups*, pages 287–325. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/DFU.Vol7.15301.11.

48 Monaldo Mastrolilli and Arash Rafiey. On the approximation of minimum cost homomorphism to bipartite graphs. *Discrete Applied Mathematics*, 161(4-5):670–676, 2013. doi:10.1016/j.dam.2011.05.002.

49 Akbar Rafiey, Arash Rafiey, and Thiago Santos. Toward a Dichotomy for Approximation of H-coloring. *CoRR*, abs/1902.02201, 2019. arXiv:1902.02201.

50 Prasad Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 245–254, 2008. doi:10.1145/1374376.1374414.

51 Rustem Takhanov. A Dichotomy Theorem for the General Minimum Cost Homomorphism Problem. In *27th International Symposium on Theoretical Aspects of Computer Science, STACS 2010, March 4-6, 2010, Nancy, France*, pages 657–668, 2010. doi:10.4230/LIPIcs.STACS.2010.2493.

52 Johan Thapper and Stanislav Zivny. The complexity of finite-valued CSPs. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 695–704, 2013. doi:10.1145/2488608.2488697.

53 Hannes Uppman. The Complexity of Three-Element Min-Sol and Conservative Min-Cost-Hom. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, pages 804–815, 2013. doi:10.1007/978-3-642-39206-1_68.

54 Hannes Uppman. Computational Complexity of the Extended Minimum Cost Homomorphism Problem on Three-Element Domains. In *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014), STACS 2014, March 5-8, 2014, Lyon, France*, pages 651–662, 2014. doi:10.4230/LIPIcs.STACS.2014.651.

55 Dmitriy Zhuk. A Proof of CSP Dichotomy Conjecture. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 331–342, 2017. doi:10.1109/FOCS.2017.38.

Beating Fredman-Komlós for Perfect k-Hashing

Venkatesan Guruswami
Computer Science Department, Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA, USA, 15213
venkatg@cs.cmu.edu

Andrii Riazanov
Computer Science Department, Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA, USA, 15213
riazanov@cs.cmu.edu

Abstract

We say a subset $C \subseteq \{1, 2, \ldots, k\}^n$ is a k-hash code (also called k-separated) if for every subset of k codewords from C, there exists a coordinate where all these codewords have distinct values. Understanding the largest possible rate (in bits), defined as $(\log_2 |C|)/n$, of a k-hash code is a classical problem. It arises in two equivalent contexts: (i) the smallest size possible for a perfect hash family that maps a universe of N elements into $\{1, 2, \ldots, k\}$, and (ii) the zero-error capacity for decoding with lists of size less than k for a certain combinatorial channel.

A general upper bound of $k!/k^{k-1}$ on the rate of a k-hash code (in the limit of large n) was obtained by Fredman and Komlós in 1984 for any $k \geq 4$. While better bounds have been obtained for $k = 4$, their original bound has remained the best known for each $k \geq 5$. In this work, we present a method to obtain the first improvement to the Fredman-Komlós bound for every $k \geq 5$, and we apply this method to give explicit numerical bounds for $k = 5, 6$.

2012 ACM Subject Classification Theory of computation → Error-correcting codes; Mathematics of computing → Coding theory

Keywords and phrases Coding theory, perfect hashing, hash family, graph entropy, zero-error information theory

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.92

Category Track A: Algorithms, Complexity and Games

Related Version A full version is available at `https://eccc.weizmann.ac.il/report/2018/096/`.

Funding *Venkatesan Guruswami*: Research supported in part by NSF grants CCF-1422045 and CCF-1563742.

Acknowledgements Some of this work was done when the first author was visiting the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore and the Center of Mathematical Sciences and Applications, Harvard University.

1 Introduction

A code of length n over an alphabet of size k is a subset $C \subseteq \{1, 2, \ldots, k\}^n$. We say such a code C is a *k-hash code* (also called k-separated in the literature), if for every subset of k distinct codewords $\{c^{(1)}, c^{(2)}, \ldots, c^{(k)}\}$ from C, there exists a coordinate j such that all these codewords differ in this coordinate, i.e. $\{c_j^{(1)}, c_j^{(2)}, \ldots, c_j^{(k)}\} = \{1, 2, \ldots, k\}$. The rate (in bits) of the code is defined as $R = \frac{\log_2 |C|}{n}$. Then for each fixed integer k, let R_k be the limit superior ($\limsup$), as $n \to \infty$, of the rate of the largest k-hash code of length n.

The study of the quantity R_k is a fundamental problem in combinatorics, information theory, and computer science. As the name suggests, k-hash codes have strong connections to the hashing problem. A family of functions mapping a universe of size N to the set

$\{1, 2, \ldots, k\}$ is called a perfect k-hash family if any k elements of the universe are mapped in one-to-one fashion by at least one hash function from this family. If C is a k-hash code, then a perfect k-hash family for universe C with n functions is just the family of coordinate projections. Therefore, R_k gives the growth rate of the size of universes for which perfect k-hash families of a given size exist. Equivalently, an upper bound on R_k is equivalent to a lower bound on the size of a perfect k-hash family as a function of the universe size.

An equivalent information-theoretic context in which k-hash codes arise concerns zero-error list decoding on certain channels. A channel can be thought of as a bipartite graph (V, W, E), where V is the set of channel inputs, W is the set of channel outputs, and $(v, w) \in E$ if on input v the channel can output w. The $k/(k-1)$ channel then is the channel with $V = W = \{1, 2, \ldots, k\}$, and $(v, w) \in E$ iff $v \neq w$. In this context, R_k is the largest asymptotic rate at which one can communicate using n repeated uses of the channel (as n grows), if we want to ensure that the receiver can identify a subset of at most $k-1$ sequences that is guaranteed to contain the transmitted sequence. See [4, 3] for more details.

Studying the rates of the codes and hashing family sizes in the above settings is a longstanding problem. A probabilistic argument shows the existence of k-hash codes with rate at least $\frac{1}{k-1} \log \frac{1}{1-k!/k^k} - o(1)$ [5, 10], and better bounds are known for some small values of k. Our focus here is on *upper bounds* on R_k, that is limitations on the size of k-hash codes. Here the best-known general upper bound on the rate R_k dates all the way back to the 1984 paper of Fredman and Komlós [5]:

$$R_k \leq \frac{k!}{k^{k-1}} =: \alpha_k \ . \tag{1}$$

For large k the multiplicative discrepancy between the probabilistic lower bound on R_k and the above Fredman-Komlós upper bound (1) grows approximately as k^2, so the current bounds on the rate require tightening to obtain better estimations of R_k. There is another trivial upper bound, $R_k \leq \log_2 \left(\frac{k}{k-1}\right)$, that follows from a simple double-counting or first moment method. The above bound (1) is much better than this bound for $k \geq 4$. For $k = 3$ (which is called the trifference problem by Körner), however, $R_3 \leq \log_2(3/2) \approx 0.585$ remains the best upper bound, and improving it (or showing it can be achieved!) is a major combinatorial challenge. For the case $k = 4$, the bound (1) which states $R_4 \leq 0.375$ has been improved, first by Arikan to 0.3512 [1], and recently by Dalai, Guruswami, and Radhakrishnan [3] to $6/19 \leq 0.3158$.

However, the above quantity α_k remained the best known upper bound on R_k for each $k > 4$. Our main result gives the first improvement to the Fredman-Komlós bound (1) for $k \geq 5$, proving that R_k is strictly smaller than α_k for every k.

▶ **Theorem 1.** *For all $k \geq 4$ there exists β_k such that $R_k \leq \beta_k < \alpha_k$. For $k = 5, 6$, we have the explicit upper bounds $R_5 < \beta_5 = 0.190826 < 0.192 = \alpha_5$, and $R_6 < \beta_6 = 0.0922789 < 0.0\overline{925} = \frac{5}{54} = \alpha_6$.*

Our approach provides a method to compute the explicit bound β_k for any $k \geq 5$ by finding a root of a degree-$O(k)$ polynomial, which lies within a specific interval. Moreover, we present a technical conjecture on the optimum values of certain polynomial optimization problems over the simplex, assuming which even stronger upper bounds on R_k can be obtained. Using the tools of numerical optimization in *Mathematica*, we verify this conjecture for the cases $k = 5, 6$, which gives us better values of β_5 and β_6, claimed in the Theorem above.

Our approach is also applicable to the (b, k)-hashing problem for $b \geq k$, where one considers codes $C \subseteq \{1, 2, \ldots, b\}^n$ with the property that for any k distinct codewords $\{c^{(1)}, c^{(2)}, \ldots, c^{(k)}\}$ from C there exists a coordinate j such that all these codewords differ

in this coordinate. Using exactly the same arguments, we obtain an improvement on the Körner-Marton upper bound [9] on the rate of such codes. When $b = k$, this latter bound is identical to the Fredman-Komlós bound, but can be better than the corresponding bound in [5] when $b > k$. For some small pairs of values (b, k) with $b > k$, the Körner-Marton bound was further improved by Arikan [1]. In those cases, the bounds we get are probably weaker than Arikan's. For this reason and for the sake of simplicity, in this paper we analyze only the case $b = k$, which corresponds to k-hashing, but all our proofs generalize in a straightforward way for (b, k)-hashing as well.

2 Background and approach

The previous general upper bounds on the rates of k-hash codes by Fredman and Komlós [5], Körner and Marton [9], and Arikan [1] are all based on information-theoretic inequalities for graph covering, related to the Hansel lemma [6]. Körner [10] cast the Fredman-Komlós proof in the language of graph entropy, which he had introduced in [8] (see [12] for a nice survey on graph entropy). Körner and Marton [9] generalized this approach to the hypergraph case, which led to improvements to the Fredman-Komlós bound for the (b, k)-hashing problem in certain cases when $b > k$, but not for R_k. In this paper we use the following version of the Hansel lemma, which is also proved in [11] via a simple probabilistic argument:

▶ **Lemma 2** (Hansel). *Let K_m be a complete graph on m vertices. Let also $G_1, G_2, \ldots, G_t$ be bipartite graphs, such that $E(K_m) = \bigcup_{i=1}^{t} E(G_i)$. Denote by $\tau(G_i)$ the fraction of non-isolated vertices in G_i. Then the following holds:*

$$\log m \leq \sum_{i=1}^{t} \tau(G_i). \tag{2}$$

To relate this lemma to the context of the paper, consider a k-hash code $C \subseteq [k]^n$. Take a subset of this code $\{x_1, x_2, \ldots, x_{k-2}\} \subseteq C$, and define bipartite graphs $G_i^{x_1,\ldots,x_{k-2}}$, for $i \in [n]$, as follows:

$$V(G_i^{x_1,\ldots,x_{k-2}}) = C \setminus \{x_1, x_2, \ldots, x_{k-2}\},$$
$$E(G_i^{x_1,\ldots,x_{k-2}}) = \left\{ \{y_1, y_2\} \ : \ (y_1)_i, (y_2)_i, (x_1)_i, (x_2)_i, \ldots, (x_{k-2})_i \text{ are distinct} \right\}.$$

Note that since C is a k-hash code, for any pair $\{y_1, y_2\} \subseteq C \setminus \{x_1, x_2, \ldots, x_{k-2}\}$, there exists some coordinate i, such that all the k codewords $y_1, y_2, x_1, x_2, \ldots, x_{k-2}$ differ in the i^{th} coordinate. In other words, $\{y_1, y_2\} \in E(G_i^{x_1,\ldots,x_{k-2}})$ for this i. Therefore, $E\left(K_{|C|-(k-2)}\right) = \bigcup_{i=1}^{n} E(G_i^{x_1,\ldots,x_{k-2}})$. Then Hansel lemma 2 applies directly, and denoting $\tau_i(x_1, x_2, \ldots, x_{k-2}) = \tau\left(G_i^{x_1,\ldots,x_{k-2}}\right)$, we obtain

$$\log\left(|C| - k + 2\right) \leq \sum_{i=1}^{n} \tau_i(x_1, x_2, \ldots, x_{k-2}). \tag{3}$$

Taking the expectation over the choice of $x_1, x_2, \ldots, x_{k-2}$, we get

$$\log\left(|C| - k + 2\right) \leq \sum_{i=1}^{n} \mathbb{E}[\tau_i(x_1, x_2, \ldots, x_{k-2})]. \tag{4}$$

By bounding the RHS of the above inequality one might obtain an upper bound on $\log |C|$, and thus on the rate of this code. Different strategies to pick the codewords $\{x_1, x_2, \ldots, x_{k-2}\}$ from C lead to different approaches to bound the RHS of (4). Here we briefly present the ideas underlying the previous works and then outline our approach.

In the original bound by Fredman and Komlós [5] the codewords $x_1, x_2, \ldots, x_{k-2}$ are picked independently at random from the code C. Then one can use symmetry arguments (or Muirhead's inequality) to bound the RHS of (4), which leads to the inequality

$$R_k \leq \frac{k!}{k^{k-1}}. \tag{5}$$

Due to the symmetry arguments involved, this bound is actually tight only in the case when the frequencies of the symbols of the code C in each coordinate are exactly uniform.

Arikan [1, 2] used rate versus distance results (the Plotkin bound) from coding theory to ensure that it is possible to pick $x_1, x_2, \ldots, x_{k-2}$ which agree on many coordinates. Note that this already guarantees that many terms in the RHS of (3) equal 0. Together with an argument which allows to modify the code so that it doesn't have any coordinate where the symbols have an overly skewed (far from uniform) frequency, Arikan was able to improve the bound (5) for $k = 4$. However, no improvement was gained for larger k.

Dalai, Guruswami, and Radhakrishnan [3] combine aspects of the above two approaches for the case $k = 4$. As in Arikan's work, they pick x_1, x_2 to agree on the first several coordinates. However, instead of a fixed such choice, they pick such a pair at random from a rich subcode of C with a common prefix. Considering such subcodes with common prefixes is a standard approach that leads to the Plotkin bound. The technical crux of the argument in [3] is a concavity claim for some quadratic form which says that despite conditioning on a common prefix, which might greatly alter the frequency vector of symbols in any coordinate in the suffix, the Fredman-Komlós bound for completely random x_1, x_2 is still valid on those coordinates. (In some sense, only the average frequency vector over all prefixes matters, not the individual ones.) Actually this holds modulo a technical condition that there are no coordinates with very skewed symbol distribution, which can be ensured by some pre-processing of the code similar to [2]. Thus some terms in (4) are equal to 0 and the other are bounded by 3/8, and balancing these appropriately, a bound of $R_4 \leq \frac{6}{19}$ is obtained in [3].

In this work, we follow the strategy of [3] for general k by picking $x_1, x_2, \ldots, x_{k-2}$ randomly so that they all lie in a rich subcode of C. However, rather than taking Plotkin-type subcode with a common prefix, we consider a subcode C which takes at most $(k-3)$ values on each coordinate from some large set T. This again implies that the coordinates from T contribute 0 to the RHS of (4). In this case, however, the analogous concavity claim seems out of reach, as one has to argue about degree-$(k-2)$ polynomials rather than quadratics. We instead take a different approach that works directly with the arbitrary symbol frequencies that may arise upon conditioning within a subcode, avoiding the averaging or concavity step. (This leads to worse bounds, but still allows to beat the Fredman-Komlós bound for $k > 4$.) However, another problem arises in that the constraint on the code to have non-skewed frequencies in each coordinate cannot be dealt with using Arikan's argument for large k. To cope with this issue, we differentiate two separate cases: (i) where C has only a few coordinates with skewed distributions of symbols, and (ii) where there are a lot of such coordinates.

- In the first case, we pick the coordinates T (where $x_1, x_2, \ldots, x_{k-2}$ are chosen to collide) to include all these skewed coordinates. Note that this is unlike [1, 3] where any choice of T of prescribed size works. Our choice of T ensures that in the remaining coordinates the frequency vector is not too far from uniform, and we apply the approach of [3] to get an improvement upon the Fredman-Komlós bound.

- In the second case, we use the original random strategy of picking $x_1, x_2, \ldots, x_{k-2}$ as in [5]. The idea here is that the bound (5) is tight only when all the frequencies of symbols are exactly uniform. Then, in the case when there are a lot of far-from-uniform frequencies, it is possible to improve the bound (5).

By picking the correct way to differentiate between skewed and non-skewed distributions, we then obtain an improvement on the Fredman-Komlós bound (5) for every $k \geq 4$ in section 3.3. As mentioned earlier, this is the *first* such improvement for $k \geq 5$. For $k = 5$ and $k = 6$ we also use numerical optimization tools to provide slightly stronger explicit bounds on R_k.

3 Upper bound on the rate of k-hash codes

Let $\Sigma = \{1, 2, \ldots, k\} = [k]$, and let $C \subseteq \Sigma^n$ be a k-hash code with rate $R = \frac{\log |C|}{n}$ (all logarithm are to the base 2). Let $f_i \in \mathbb{R}^k$ be the frequency vector of symbols of the code for each coordinate $i \in [n]$, namely:

$$f_i[a] = \frac{1}{|C|}|\{x \in C : x_i = a\}|.$$

Throughout the analysis, we will be interested in two cases: when for most of the coordinates the distribution of codeword symbols is close to uniform (non-skewed), or when this doesn't hold. To define the term "close to uniform" formally, we consider a *threshold* γ, that satisfies $\frac{1}{2k-3} \leq \gamma \leq \frac{1}{k}$, and say that $f \in \mathbb{R}^k$ is close to uniform when $f[a] \geq \gamma$ for all $a \in [k]$. Denote then $P_\gamma = \{i \in [n] : \min\limits_{a \in \Sigma} f_i[a] \geq \gamma\}$ – the set of all the coordinates for which the distribution of codeword symbols is close to uniform. Denote also $\ell := \left\lfloor \frac{nR - \log n}{\log\left(\frac{k}{k-3}\right)} \right\rfloor$. We then consider two cases:

1. *Unbalanced:* $|P_\gamma| < n - \ell$, so there is a decent fraction of coordinates where the distribution of codeword symbols is skewed. For this case, we apply a random strategy to pick $x_1, x_2, \ldots, x_{k-2}$ in (4).
2. *Almost balanced:* $|P_\gamma| \geq n - \ell$, so for almost all coordinates, the distribution of codeword symbols is close to uniform. Then we follow the approach from [3] to pick $x_1, x_2, \ldots, x_{k-2}$ which collide on many coordinates.

For both of these cases, we will obtain some bounds on the rate of C, which depend on the threshold γ. It then will remain to choose γ in a manner ensuring that both these bounds beat (5). Then, since for any code C exactly one of the cases holds, we can obtain a general upper bound on the rate.

Before we continue with studying the two cases separately, let's look at how we can estimate $\tau_i(x_1, x_2, \ldots, x_{k-2})$. Clearly, the codeword $y \in C$ appears non-isolated in the graph $G_i^{x_1, x_2, \ldots, x_{k-2}}$ only if all the codewords $x_1, x_2, \ldots, x_{k-2}$ and y differ in the i^{th} coordinate. Therefore, the fraction of non-isolated vertices in $G_i^{x_1, x_2, \ldots, x_{k-2}}$ satisfies

$$\tau_i(x_1, \ldots, x_{k-2}) \leq \left(\frac{|C|}{|C| - (k-2)}\right)\left(1 - f_i[(x_1)_i] - f_i[(x_2)_i] - \cdots - f_i[(x_{k-2})_i]\right) \times \\ \times \mathbf{1}\Big[(x_1)_i, (x_2)_i, \ldots, (x_{k-2})_i \text{ distinct}\Big], \tag{6}$$

where $\mathbf{1}[E]$ is the indicator variable for an event/condition E.

3.1 Unbalanced case

We will pick $x_1, x_2, \ldots, x_{k-2}$ uniformly at random without replacement from C to obtain an upper bound on the rate of C from (4). Taking the expectations of the both sides in (6) gives

$$\begin{aligned}
&\mathbb{E}[\tau_i(x_1, \ldots, x_{k-2})] \\
&\quad \le \frac{|C|}{|C|-k+2} \sum_{\substack{a_1,\ldots,a_{k-2} \in \Sigma \\ \{a_s\} \text{ distinct}}} \left(1 - \sum_{s=1}^{k-2} f_i[a_s]\right) \cdot \mathbb{P}\big[(x_s)_i = a_s,\ s = 1, \ldots, (k-2)\big] \\
&\quad = \frac{|C|}{|C|-k+2} \prod_{j=0}^{k-3} \frac{|C|}{|C|-j} \sum_{\substack{a_1,a_2,\ldots,a_{k-2} \in \Sigma \\ \{a_s\} \text{ distinct}}} \left(1 - \sum_{s=1}^{k-2} f_i[a_s]\right) \cdot f_i[a_1] f_i[a_2] \ldots f_i[a_{k-2}],
\end{aligned} \tag{7}$$

where the coefficients $\frac{|C|}{|C|-j}$, $j = 0, 1, \ldots, k-3$ appear because we pick elements from C without replacement. Define the following function of two probability vectors $g, f \in \mathbb{R}^k$:

$$\phi_k(g, f) := \sum_{\substack{a_1,a_2,\ldots,a_{k-2} \in \Sigma \\ \{a_s\} \text{ distinct}}} \prod_{s=1}^{k-2} g[a_s] \Big(1 - \sum_{s=1}^{k-2} f[a_s]\Big). \tag{8}$$

Using this notation, we derive from (7):

$$\mathbb{E}[\tau_i(x_1, x_2, \ldots, x_{k-2})] \le \phi_k(f_i, f_i)\big(1 + o(1)\big). \tag{9}$$

Since $\sum_{a \in \Sigma} f_i[a] = 1$, it is easy to see that $\phi_k(f_i, f_i)$ is a symmetric expression in $f_i[a]$ for all $a \in \Sigma$. Denote by $S_h^t(g)$ the h-th elementary symmetric sum of the first t coordinates of the vector $g \in \mathbb{R}^k$, i.e. the sum of all products of h distinct elements from $\{g[1], g[2], \ldots, g[t]\}$. Then we can write

$$\phi_k(f_i, f_i) = (k-2)! \cdot \binom{k-1}{k-2} S_{k-1}^k(f_i) = (k-1)! \cdot S_{k-1}^k(f_i)$$

It is not hard to show that $S_h^k(g)$ for g being a probability vector in $\mathbb{R}^k$ is maximized when g is uniform. Indeed, if there are two non-equal coordinates $g[a] \ne g[b]$, then substituting the values in these coordinates by their arithmetic average strictly increases the value of $S_h^k(g)$. Then let us denote by u the uniform distribution on k elements, i.e. $u[a] = 1/k$ for all $a \in [k]$, and so $S_h^k(g) \le S_h^k(u)$. Then in (9)

$$\mathbb{E}[\tau_i(x_1, x_2, \ldots, x_{k-2})] \le (k-1)! \cdot S_{k-1}^k(f_i) \cdot \big(1 + o(1)\big) \le (k-1)! \cdot S_{k-1}^k(u) \cdot \big(1 + o(1)\big),$$

where we compute $S_{k-1}^k(u) = \binom{k}{k-1} \cdot \left(\frac{1}{k}\right)^{k-1} = \left(\frac{1}{k}\right)^{k-2}$. Therefore, we retrieve

$$\mathbb{E}[\tau_i(x_1, x_2, \ldots, x_{k-2})] \le \frac{(k-1)!}{k^{k-2}} \cdot \big(1 + o(1)\big) = \frac{k!}{k^{k-1}} \cdot \big(1 + o(1)\big). \tag{10}$$

Substituting this inequality into (4), notice that we derive exactly the Fredman-Komlós bound (5). Denote then

$$\alpha_k := \frac{k!}{k^{k-1}},$$

the Fredman-Komlós upper bound on the rate R_k.

Now recall that we are considering the unbalanced case, in which there are a lot of coordinates with frequencies of codeword symbols being far from uniform. Take i to be any of such coordinates, and let for convenience $f = f_i$, so $\min_{a\in\Sigma} f[a] < \gamma$. Without loss of generality, say $f[k] < \gamma$. Notice the following trivial property of symmetric sums:

$$\phi_k(f,f) = (k-1)!\cdot S^k_{k-1}(f) = (k-1)!\Big(S^{k-1}_{k-1}(f) + f[k]\cdot S^{k-1}_{k-2}(f)\Big).$$

The above expression is symmetric in the first $(k-1)$ coordinates of f. Let's then fix $f[k]$, and do the same averaging operations with all the remaining coordinates of f, making in the end $f'[1] = f'[2] = \cdots = f'[k-1] = \frac{1-f[k]}{k-1}$. The value of $\phi_k(f,f)$ only increases after such operations, so

$$\phi_k(f,f) \le (k-1)!\Big(f'[1]f'[2]\cdots f'[k-1] + f[k]\cdot S^{k-1}_{k-2}(f')\Big).$$

Let $y = \frac{1-f[k]}{k-1}$, so $f[k] = 1-(k-1)y$. Since $0 \le f[k] < \gamma$ by the assumption above, it holds $\frac{1-\gamma}{k-1} \le y \le \frac{1}{k-1}$. Recall that we took the threshold $\gamma \le \frac{1}{k}$, thus $y \ge \frac{1-\gamma}{k-1} \ge \frac{1}{k}$. Then

$$\phi_k(f,f) \le (k-1)!\Big(y^{k-1} + \big(1-(k-1)y\big)\cdot(k-1)y^{k-2}\Big) = (k-1)!y^{k-2}\Big((k-1)-(k^2-2k)y\Big).$$

Denote $G_k(y) = (k-1)!y^{k-2}\Big((k-1)-(k^2-2k)y\Big)$, so $\phi_k(f,f) \le G_k(y)$. We have

$$\big(G_k(y)\big)' = (k-1)!(k-1)(k-2)y^{k-3}\big(1-ky\big), \tag{11}$$

so the derivative of G_k is negative on the interval $\frac{1}{k} \le \frac{1-\gamma}{k-1} < y \le \frac{1}{k-1}$, and it is zero at $y = \frac{1}{k}$. Therefore, we finally obtain for any such f:

$$\phi_k(f,f) \le \max_{y\in\left[\frac{1-\gamma}{k-1},\frac{1}{k-1}\right]} G_k(y) = G_k\left(\frac{1-\gamma}{k-1}\right). \tag{12}$$

Note that $G_k\left(\frac{1-\gamma}{k-1}\right) \le G_k\left(\frac{1}{k}\right) = \alpha_k$ for any $\gamma \le \frac{1}{k}$, and the strict inequality $G_k\left(\frac{1-\gamma}{k-1}\right) < G_k\left(\frac{1}{k}\right) = \alpha_k$ holds when $\gamma < \frac{1}{k}$.

So if $\min_{a\in[k]} f_i[a] < \gamma$ for some coordinate i, we retrieved the bound

$$\mathbb{E}[\tau_i(x_1,x_2,\ldots,x_{k-2})] \le G_k\left(\frac{1-\gamma}{k-1}\right)\big(1+o(1)\big). \tag{13}$$

For now we obtained two bounds for the summands in the RHS of (4): (i) the bound (10) holds for all the coordinates, and (ii) the bound (13) holds for the coordinates with codeword symbol frequencies far from uniform. As we noted above, the second bound is strictly stronger than the first bound when we take the threshold $\gamma < \frac{1}{k}$. Also recall that in the unbalanced case which we now consider, there are a lot of coordinates of the second type, so essentially the bound (13) applies many times. Let's now formalize this argument to obtain an improvement on the Fredman-Komlós bound for the unbalanced case.

Denote $\xi_k(\gamma) = G_k\left(\frac{1-\gamma}{k-1}\right)$, then

$$\xi_k(\gamma) = (k-1)!\frac{(1-\gamma)^{k-2}}{(k-1)^{k-2}}\,\frac{(k-1)^2-(k^2-2k)(1-\gamma)}{k-1} = \frac{(k-2)!(1-\gamma)^{k-2}\big((k^2-2k)\gamma+1\big)}{(k-1)^{k-2}}, \tag{14}$$

and note that $\xi_k(\gamma) \le \alpha_k$ for $\gamma \le \frac{1}{k}$. Recall that we denoted by P_γ the set of coordinates i for which $\min_{a\in\Sigma} f_i[a] \ge \gamma$. For such $i \in P_\gamma$ we directly apply the bound (10). For all the other coordinates $i \in [n] \setminus P_\gamma$ we use the inequality (13). In the unbalanced case $|P_\gamma| < n - \ell$, thus $n - |P_\gamma| > \ell$. Applying all these arguments to (4), we obtain

$$\begin{aligned}\log(|C| - k + 2) &\le \Big(|P_\gamma|\alpha_k + \big(n - |P_\gamma|\big)\xi_k(\gamma)\Big)(1 + o(1)) \\ &< \Big(n\alpha_k - \ell\left(\alpha_k - \xi_k(\gamma)\right)\Big)(1 + o(1)) \\ &\le \left(n\alpha_k - \frac{nR}{\log\left(\frac{k}{k-3}\right)}\left(\alpha_k - \xi_k(\gamma)\right) + o(n)\right)(1 + o(1)),\end{aligned}$$

where $\ell = \left\lfloor \frac{nR - \log n}{\log\left(\frac{k}{k-3}\right)} \right\rfloor$. Since $|C| = 2^{Rn}$, the above implies for $n \to \infty$:

$$R \le \alpha_k - \frac{R\left(\alpha_k - \xi_k(\gamma)\right)}{\log\left(\frac{k}{k-3}\right)} + o(1),$$

$$\boxed{R_k^{\text{unbal}}(\gamma) \le \frac{\alpha_k}{1 + \frac{\alpha_k - \xi_k(\gamma)}{\log\left(\frac{k}{k-3}\right)}}.} \tag{15}$$

Note that for $\gamma = \frac{1}{k}$ the above bound becomes equal to α_k, since $\xi_k(1/k) = G\left(\frac{1-1/k}{k-1}\right) = G\left(\frac{1}{k}\right) = \alpha_k$. Moreover, the previous analysis (11) of the function $G_k(\cdot)$ implies that the RHS of the above bound is strictly increasing as a function of γ. Thus the bound (15) is strictly better than the Fredman-Komlós bound for the unbalanced case for any threshold $\gamma < \dfrac{1}{k}$.

3.2 Almost balanced case

For this case we extend the approach used in [3] for 4-hashing. Namely, in [3] the authors considered a Plotkin-type subcode of C containing the words with the common prefix, and then picked x_1, x_2 from this subcode. For our purposes, we will consider a rich subcode of codewords which can take a restricted set of symbols on some fixed set of coordinates, and choose $x_1, x_2, \ldots, x_{k-2}$ randomly from the subcode. In the almost balanced case, we are able to ensure that the distributions of codeword symbols in all non-fixed coordinates are close to uniform, which will allow us to use some continuity argument to bound the RHS of (4).

In the almost balanced case we assume $|P_\gamma| \ge n - \ell$, so there are at most ℓ coordinates where the distribution of codeword symbols is skewed. The set of such coordinates is $\overline{P_\gamma} = [n] \setminus P_\gamma$, $|\overline{P_\gamma}| \le \ell$. Then take any subset $T \subset [n]$, such that $\overline{P_\gamma} \subseteq T$ and $|T| = \ell$, and denote $S = [n] \setminus T$.

Our goal is to find a subcode of C of sufficient size, such that any $(k-2)$ codewords $x_1, x_2, \ldots, x_{k-2}$ from this subcode collide in all the coordinates from T. In other words, for any coordinate $t \in T$ there should exist i, j such that $(x_i)_t = (x_j)_t$. This will ensure that the coordinates from T contribute 0 to the RHS of (4), which will allow us to prove a better bound on the rate of the code C. We will now define the subcodes which satisfy this property.

First, denote by $\binom{\Sigma}{p}$ the family of p-element subsets of the alphabet $\Sigma = \{1, 2, \ldots, k\}$. Then define $\Omega := \underbrace{\binom{\Sigma}{k-3} \times \binom{\Sigma}{k-3} \times \cdots \times \binom{\Sigma}{k-3}}_{\ell}$.

Now, for any $\omega \in \Omega$ and any string $s \in \Sigma^\ell$, denote $s \vdash \omega$ if $s_1 \in \omega_1, s_2 \in \omega_2, \ldots, s_\ell \in \omega_\ell$. Then, for any $\omega \in \Omega$, we define:

$$C_\omega := \{x \in C : x_{\{T\}} \vdash \omega\},$$

where $x_{\{T\}}$ is the projection of the codeword x on the set of coordinates T. Notice that C_ω has the property we discussed above. Indeed, for any pick $x_1, x_2, \ldots, x_{k-2} \in C_\omega$ and any $t \in T$, it holds $(x_1)_t, (x_2)_t, \ldots, (x_{k-2})_t \in \omega_t$, but $|\omega_t| = k-3$, and therefore $(x_1)_t, (x_2)_t, \ldots, (x_{k-2})_t$ are not all distinct.

Denote then $M_\omega = |C_\omega|$. Note that for each $x \in C$ there are exactly $\binom{k-1}{k-4}^\ell$ different elements $\omega \in \Omega$ such that $x_{\{T\}} \vdash \omega$. Therefore

$$\sum_{\omega \in \Omega} M_\omega = |C| \cdot \binom{k-1}{k-4}^\ell.$$

It suffices to prove that there exists at least one $\omega \in \Omega$ such that $M_\omega \geq n$ for our arguments further. For the sake of contradiction, suppose $M_\omega < n$ for all $\omega \in \Omega$. But then

$$2^{nR} = |C| = \sum_{\omega \in \Omega} M_\omega \frac{1}{\binom{k-1}{k-4}^\ell} < \frac{\binom{k}{k-3}^\ell}{\binom{k-1}{k-4}^\ell} \cdot n = \left(\frac{k}{k-3}\right)^\ell n = 2^{\ell \cdot \log \frac{k}{k-3} + \log n} \leq 2^{nR},$$

where $\ell = \left\lfloor \frac{nR - \log n}{\log\left(\frac{k}{k-3}\right)} \right\rfloor$. The above is a contradiction, so there exists such $\omega \in \Omega$ that $M_\omega \geq n$.

We are finally ready to describe the strategy to pick the codewords $x_1, x_2, \ldots, x_{k-2}$ in the almost balanced case. We do the following: first, deterministically choose some $\omega \in \Omega$ such that $M_\omega \geq n$, and then pick $x_1, x_2, \ldots, x_{k-2}$ uniformly at random (without replacement) from C_ω. Since all the codewords collide on the coordinates from the set T, we obtain in (4):

$$\log(|C| - k + 2) \leq \sum_{m \in [n]} \mathbb{E}[\tau_m(x_1, x_2, \ldots, x_{k-2})] = \sum_{m \in S} \mathbb{E}[\tau_m(x_1, x_2, \ldots, x_{k-2})]. \tag{16}$$

Now fix some $m \in S$, and let $f_{m|\omega}$ be the frequency vector of the m^{th} coordinate in the subcode C_ω. Taking expectation over the choice of $x_1, x_2, \ldots, x_{k-2}$ in (6) with respect to the the random strategy described above, we have

$$\begin{aligned}
&\mathbb{E}[\tau_m(x_1, x_2, \ldots, x_{k-2})] \\
&= \frac{|C|}{|C| - k + 2} \prod_{j=0}^{k-3} \frac{|C_\omega|}{|C_\omega| - j} \sum_{\substack{a_1, \ldots, a_{k-2} \in \Sigma \\ \{a_s\} \text{ distinct}}} \left(1 - \sum_{s=1}^{k-2} f_m[a_s]\right) \cdot f_{m|\omega}[a_1] f_{m|\omega}[a_2] \ldots f_{m|\omega}[a_{k-2}],
\end{aligned}$$

where the coefficients $\frac{|C_\omega|}{|C_\omega| - j}$, $j = 0, 1, \ldots, (k-3)$ appear because we pick $(k-2)$ elements from C_ω without replacement. Since we took ω such that $|C_\omega| \geq n$, it follows that $\frac{|C_\omega|}{|C_\omega| - j} \leq \frac{n}{n-j}$.

Using the function $\phi_k(g, f)$ which was defined in (8), we can rewrite the above as

$$\mathbb{E}[\tau_m(x_1, x_2, \ldots, x_{k-2})] \leq \prod_{j=0}^{k-2} \left(\frac{n}{n-j}\right) \phi_k(f_{m|\omega}, f_m) = \phi_k(f_{m|\omega}, f_m) \cdot \big(1 + o(1)\big). \tag{17}$$

Consider the following definition:

$$\boxed{\theta_k(\gamma) := \max_{g,f} \{\phi_k(g, f) : f, g \in \mathbb{R}^k \text{ are probability vectors, } \min_{a \in \Sigma} f[a] \geq \gamma\}.} \tag{18}$$

Let's first consider the bound we obtain using this definition, and then analyze $\theta_k(\gamma)$.

Since $\min_{a\in\Sigma} f_m[a] \geq \gamma$ by construction of the set S, we have $\phi_k(f_{m|\omega}, f_m) \leq \theta_k(\gamma)$ for any $m \in S$, so substituting it into (17) gives

$$\mathbb{E}[\tau_m(x_1, x_2, \ldots, x_{k-2})] \leq \theta_k(\gamma) \cdot \big(1 + o(1)\big).$$

Therefore, in (16) we derive

$$\begin{aligned}\log(|C| - k + 2) &\leq |S| \cdot \theta_k(\gamma)\big(1 + o(1)\big) = (n - \ell) \cdot \theta_k(\gamma)\big(1 + o(1)\big) \\ &\leq \left(n - \frac{nR}{\log\left(\frac{k}{k-3}\right)} + \frac{\log n}{\log\left(\frac{k}{k-3}\right)} + 1\right) \theta_k(\gamma)\big(1 + o(1)\big).\end{aligned}$$

Recall that $|C| = 2^{nR}$, thus for $n \to \infty$

$$R \leq \left(1 - \frac{R}{\log\left(\frac{k}{k-3}\right)}\right) \theta_k(\gamma) + o(1),$$

$$\boxed{R_k^{\mathrm{bal}}(\gamma) \leq \frac{\theta_k(\gamma)}{1 + \frac{\theta_k(\gamma)}{\log\left(\frac{k}{k-3}\right)}}.} \tag{19}$$

It now remains to understand how $\theta_k(\gamma)$, defined in (18), behaves as a function of γ.

Upper bound for $\theta_k(\gamma)$

First, note that for $\gamma = \frac{1}{k}$ the only probability vector f with $\min_{a\in\Sigma} f[a] \geq \gamma$ is the uniform vector u. Then $\phi_k(g, u)$ is an elementary symmetric sum of all the coordinates of g, and therefore we obtain $\phi_k(g, u) \leq \phi_k(u, u) = \alpha_k$, and so $\theta_k(1/k) = \alpha_k$.

Now take any $\gamma \leq \frac{1}{k}$, and let g, f be probability vectors in $\mathbb{R}^k$ such that $f[a] \geq \gamma$ for $a \in \Sigma$. We will further use "f_a" to refer to the a^{th} coordinate of vector f rather than "$f[a]$".

Let $t = \binom{k}{2} = k(k-1)/2$ and let $P_1, P_2, \ldots, P_t$ be an enumeration of all $(k-2)$-element subsets of $\Sigma = \{1, 2, \ldots, k\}$. Then we have from (8)

$$\phi_k(g, f) = \sum_{\substack{a_1,\ldots,a_{k-2}\in\Sigma \\ \{a_i\} \text{ distinct}}} \prod_{i=1}^{k-2} g_{a_i}\left(1 - \sum_{i=1}^{k-2} f_{a_i}\right) = (k-2)! \sum_{j=1}^{t} \left[\prod_{a\in P_j} g_a \cdot \left(1 - \sum_{a\in P_j} f_a\right)\right]. \tag{20}$$

Denote $d_j := \prod_{a\in P_j} g_a$, and let $d_{(i)}$ be the i^{th} order statistic of the set $\{d_1, d_2, \ldots, d_t\}$, i.e. $\{d_{(1)}, d_{(2)}, \ldots, d_{(t)}\} = \{d_1, d_2, \ldots, d_t\}$ and $d_{(1)} \geq d_{(2)} \geq \cdots \geq d_{(t)}$. The proof of the following claim can be found in the full version of the paper.

▷ Claim 3.

$$\phi_k(g, f) \leq (k-2)!\left[(1 - k\gamma)\sum_{j=1}^{k-1} d_{(j)} + 2\gamma \sum_{j=1}^{t} d_{(j)}\right]. \tag{21}$$

Now, $\sum_{j=1}^{t} d_{(j)} = \sum_{j=1}^{t} d_j$ is just an elementary symmetric sum of degree $k-2$ for the probability vector g, so this expression is maximized for the uniform vector, and then we get $\sum_{j=1}^{t} d_{(j)} \leq \binom{k}{2}\left(\frac{1}{k}\right)^{k-2}$.

Finally, we use $d_{(j)} \le \left(\frac{1}{k-2}\right)^{k-2}$, because each $d_{(j)}$ is a product of $(k-2)$ coordinates of some probability vector. Thus we obtain

$$\phi_k(g,f) \le (k-2)!\left[(1-k\gamma)\frac{(k-1)}{(k-2)^{k-2}} + \gamma\frac{(k-1)}{k^{k-3}}\right] = (k-1)!\left[\frac{(1-k\gamma)}{(k-2)^{k-2}} + \frac{\gamma}{k^{k-3}}\right]. \quad (22)$$

Since this holds for any g and any f such that $\min_{a\in\Sigma} f[a] \ge \gamma$, we obtain an upper bound

$$\theta_k(\gamma) \le (k-1)!\left[\frac{(1-k\gamma)}{(k-2)^{k-2}} + \frac{\gamma}{k^{k-3}}\right] =: \rho_k(\gamma). \quad (23)$$

Note that $\rho_k(\gamma)$ is linear in γ for a fixed k, and that $\rho_k(1/k) = \alpha_k = \theta_k(1/k)$, so this upper bound is tight for $\gamma = \frac{1}{k}$.

3.2.1 Conjecture on the exact value of $\theta_k(\gamma)$

We now describe the conjecture we make on the exact value of $\theta_k(\gamma)$. Consider the upper bound (21) on $\phi_k(g,f)$. Even for some fixed ordering of the coordinates of g, say (without loss of generality) $g_1 \ge g_2 \ge \cdots \ge g_k \ge 0$, there might be different cases of orderings within the set $\{d_1, d_2, \ldots, d_t\}$. Say there are q_k different ways to take the first $(k-1)$ order statistics within the set $\{d_1, d_2, \ldots, d_t\}$ (with this fixed ordering), then there would be q_k different functionals of d_i's, and thus of g_i's, in the RHS of (21), call them $\Theta_k^{(1)}(g,\gamma), \Theta_k^{(2)}(g,\gamma), \ldots, \Theta_k^{(q_k)}(g,\gamma)$. Since exactly one ordering within $\{d_1, \ldots, d_t\}$ is correct for any particular vector g, we obtain

$$\phi_k(g,f) \le \max_{i=1,2,\ldots,q_k} \Theta_k^{(i)}(g,\gamma).$$

Then define

$$\theta_k^{(i)}(\gamma) := \max_x \left\{\Theta_k^{(i)}(x,\gamma) : \sum_{j=1}^{k} x_j = 1, x \ge 0\right\}, \qquad \text{for } i = 1,2,\ldots,q_k, \quad (24)$$

and so the quantity $\theta_k(\gamma)$ defined in (18) satisfies

$$\theta_k(\gamma) \le \max_{i=1,2,\ldots,q_k} \theta_k^{(i)}(\gamma).$$

So to find an upper bound on $\theta_k(\gamma)$ it suffices to find the maximum among $\theta_k^{(i)}(\gamma)$ for $i = 1,2,\ldots,q_k$. Unfortunately, q_k grows exponentially as k increases, so it is not clear how to do this efficiently. We introduce a conjecture below, which suggests that we can determine which of the values $\theta_k^{(i)}(\gamma)$, $i = 1,2,\ldots,q_k$, is the greatest for any k.

Specifically, the conjecture is stated as follows: we assume that the maximum among all the values $\theta_k^{(i)}(\gamma)$, $i = 1,2,\ldots,q_k$, is the greatest for the functional $\Theta_k^{(i)}(x,\gamma)$ corresponding to the case, when the first $(k-1)$ order statistics of the set $\{d_1, d_2, \ldots, d_t\}$ form the set $\left\{\frac{\prod_{i=1}^{k-1} g_i}{g_a}\right\}_{a\in[k-1]}$. In other words, $\{d_{(1)}, d_{(2)}, \ldots, d_{(k-1)}\}$ correspond to the sets P_j that contain all their $(k-2)$ elements from $\{1,2,\ldots,k-1\}$ (recall $d_j = \prod_{a\in P_j} g_j$). So the first $(k-1)$ order statistics are formed as the products of only the first $(k-1)$ coordinates of g, ignoring the coordinate g_k. Our intuition behind the assumption that this functional will have the greatest maximum is based on symmetry arguments.

Recall that we denote by $S_h^t(g)$ the h-th elementary symmetric sum of the first t coordinates of g. Then the above conjecture can be formalized as follows:

▶ **Conjecture 4.**

$$\theta_k(\gamma) = \max_x \left\{ (k-2)! \left[\Big(1-(k-2)\gamma\Big) S_{k-2}^{k-1}(x) + 2\gamma \cdot x_k \cdot S_{k-3}^{k-1}(x) \right] : \quad \sum_{i=1}^{k} x_i = 1, x \geq 0 \right\}. \quad (25)$$

Indeed, the function $\Theta_k^{\gamma}(g) = (k-2)! \left[\Big(1-(k-2)\gamma\Big) S_{k-2}^{k-1}(g) + 2\gamma \cdot g_k \cdot S_{k-3}^{k-1}(g) \right]$ just corresponds to the functional in the RHS of (21) in the case we discussed above.

Simple computation of the RHS of (25) gives us another formulation of this conjecture:

$$\theta_k(\gamma) = \frac{(k-1)!(k-3)^{k-3}\gamma^{k-2}}{\big((k^2-2k)\gamma - 1\big)^{k-3}}. \quad (26)$$

3.3 Improvement of the Fredman-Komlós bound

In this section we show that it is possible to choose such a threshold γ that both bounds (15) and (19) are stronger than the Fredman-Komlós bound.

Using (23) in the bound (19), we obtain

$$R_k^{\text{bal}}(\gamma) \leq \frac{\theta_k(\gamma)}{1 + \frac{\theta_k(\gamma)}{\log\left(\frac{k}{k-3}\right)}} \leq \frac{\rho_k(\gamma)}{1 + \frac{\rho_k(\gamma)}{\log\left(\frac{k}{k-3}\right)}}. \quad (27)$$

Since for any k-hash code C either the unbalanced or the almost balanced case holds, and we get to choose the threshold γ to differentiate between these cases, the above, combined with (15), gives us the following upper bound for the rate in the general case:

$$R_k \leq \min_{\gamma \in \left(\frac{1}{2k-3}, \frac{1}{k}\right)} \max \left\{ \frac{\rho_k(\gamma)}{1 + \frac{\rho_k(\gamma)}{\log\left(\frac{k}{k-3}\right)}}, \ \frac{\alpha_k}{1 + \frac{\alpha_k - \xi_k(\gamma)}{\log\left(\frac{k}{k-3}\right)}} \right\}. \quad (28)$$

The optimal threshold γ is such that the bounds (27) and (15) are equal, since the first bound becomes stronger as γ increases, while the second bound becomes weaker. Therefore, the optimal threshold is the solution of the following equation:

$$\frac{\rho_k(\gamma)}{1 + \frac{\rho_k(\gamma)}{\log\left(\frac{k}{k-3}\right)}} = \frac{\alpha_k}{1 + \frac{\alpha_k - \xi_k(\gamma)}{\log\left(\frac{k}{k-3}\right)}} \quad (29)$$

where $\alpha_k = \dfrac{k!}{k^{k-1}}$ is the Fredman-Komlós bound, $\rho_k(\gamma)$ can be found using expression (23), and $\xi_k(\gamma)$ is found via (14). Note that $\rho_k(\gamma)$ is a linear function and $\xi_k(\gamma)$ is a rational functions with degree $O(k)$, and therefore the above equation is equivalent to finding a root of a polynomial of degree $O(k)$ in variable γ, which lies in the interval $\left(\frac{1}{2k-3}, \frac{1}{k}\right)$. Such a solution certainly exists, because at $\gamma = \frac{1}{k}$ the LHS is less than α_k, while the RHS is equal to α_k, however at $\gamma = \frac{1}{2k-3}$ the LHS is greater than α_k while the RHS is less than α_k. Therefore, there exists a point $\gamma^* \in \left(\frac{1}{2k-3}, \frac{1}{k}\right)$ where these bounds are equal, since these functions are continuous. The values of the bounds for $\gamma = \frac{1}{k}$ guarantee that both bounds will be less than α_k when we take the optimal threshold γ^*. Therefore, for each k this optimal threshold γ^*, substituted into (28), gives a new upper bound on the rate of k-hash codes, which is stronger than the Fredman-Komlós bound (5).

Assuming the conjecture 4 holds, we obtain a stronger bound by using the exact value of $\theta_k(\gamma)$ instead of its upper bound $\rho_k(\gamma)$. The optimal threshold in this case is the solution of

$$\frac{\theta_k(\gamma)}{1+\frac{\theta_k(\gamma)}{\log\left(\frac{k}{k-3}\right)}} = \frac{\alpha_k}{1+\frac{\alpha_k-\xi_k(\gamma)}{\log\left(\frac{k}{k-3}\right)}}, \tag{30}$$

where the value for $\theta_k(\gamma)$ is taken from (26). The value attained by the above expressions at the optimal threshold is the new upper bound on R_k.

New bounds for $k = 5$ and $k = 6$

Currently, the conjecture 4 is formulated in such a way that for any γ (from an appropriate range) one particular functional, parametrized with γ, has a maximum value on a simplex larger than some other set of functionals, also parametrized with γ. However, for our purposes of obtaining an upper bound on R_k, we only need the conjecture to hold specifically for the value of the optimal threshold $\gamma = \gamma^*$, found via (30). This is because if (26) holds for this γ^*, we then know that the upper bounds (15) and (19) for unbalanced and balanced cases are equal by the choice of γ^* and thus give a general upper bound on the rate R_k.

So to use the conjecture for a given k it just suffices to solve all optimization problems (24) for this value γ^*, and check if the conjecture indeed holds (namely, that the maximum of $\Theta_k^{(i)}(g,\gamma)$ is the greatest for the functional $\Theta_k^{\gamma}(g)$ described in the conjecture).

Applying (30) for $k = 5$ gives us the optimal threshold $\gamma_5^* \approx 0.136163$. We then use the numberical optimization tools in *Wolfram Mathematica* [7] to verify that the conjecture 4 indeed holds, which in this case reduces to optimizing two degree-3 multilinear polynomials with 5 variables over the simplex. After verifying the conjecture, we obtain the new general bound for 5-hashing:

$$R_5 < 0.190826 < 0.192 = \frac{24}{125} = \alpha_5.$$

For $k = 6$, the above approach gives us:

$$R_6 < 0.0922789 < 0.0\overline{925} = \frac{5}{54} = \alpha_6.$$

References

1. E. Arikan. A Bound on the Zero-Error List Coding Capacity. In *Proceedings. IEEE International Symposium on Information Theory*, pages 152–152, 1993. doi:10.1109/ISIT.1993.748467.
2. E. Arikan. An upper bound on the zero-error list-coding capacity. *IEEE Transactions on Information Theory*, 40(4):1237–1240, 1994. doi:10.1109/18.335947.
3. M. Dalai, V. Guruswami, and J. Radhakrishnan. An improved bound on the zero-error list-decoding capacity of the 4/3 channel. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 1658–1662, 2017. doi:10.1109/ISIT.2017.8006811.
4. Peter Elias. Zero error capacity under list decoding. *IEEE Trans. Information Theory*, 34(5):1070–1074, 1988. doi:10.1109/18.21233.
5. Michael L. Fredman and János Komlós. On the Size of Separating Systems and Families of Perfect Hash Functions. *SIAM Journal on Algebraic Discrete Methods*, 5(1):61–68, 1984. doi:10.1137/0605009.
6. G. Hansel. Nombre minimal de contacts de fermature nécessaires pour réaliser une fonction booléenne symétrique de n variables. *C. R. Acad. Sci. Paris*, pages 6037–6040, 1964.
7. Wolfram Research, Inc. Mathematica, Version 11.3. Champaign, IL, 2018.

8 J. Körner. Coding of an information source having ambiguous alphabet and the entropy of graphs. *6th Prague Conference on Information Theory*, pages 411–425, 1973.

9 J. Körner and K. Marton. New Bounds for Perfect Hashing via Information Theory. *European Journal of Combinatorics*, 9(6):523–530, 1988. doi:10.1016/s0195-6698(88)80048-9.

10 János Körner. Fredman–Komlós bounds and information theory. *SIAM Journal on Algebraic Discrete Methods*, 7(4):560–570, 1986. doi:10.1137/0607062.

11 A. Nilli. Perfect Hashing and Probability. *Combinatorics, Probability and Computing*, 3(03):407–409, 1994. doi:10.1017/s0963548300001280.

12 Jaikumar Radhakrishnan. Entropy and Counting, 2001. URL: http://www.tcs.tifr.res.in/~jaikumar/Papers/EntropyAndCounting.pdf.

Random Walks on Dynamic Graphs: Mixing Times, Hitting Times, and Return Probabilities

Thomas Sauerwald
Department of Computer Science and Technology, University of Cambridge, United Kingdom
thomas.sauerwald@cl.cam.ac.uk

Luca Zanetti
Department of Computer Science and Technology, University of Cambridge, United Kingdom
luca.zanetti@cl.cam.ac.uk

Abstract

We establish and generalise several bounds for various random walk quantities including the mixing time and the maximum hitting time. Unlike previous analyses, our derivations are based on rather intuitive notions of local expansion properties which allow us to capture the progress the random walk makes through t-step probabilities.

We apply our framework to dynamically changing graphs, where the set of vertices is fixed while the set of edges changes in each round. For random walks on dynamic connected graphs for which the stationary distribution does not change over time, we show that their behaviour is in a certain sense similar to static graphs. For example, we show that the mixing and hitting times of any sequence of d-regular connected graphs is $O(n^2)$, generalising a well-known result for static graphs. We also provide refined bounds depending on the isoperimetric dimension of the graph, matching again known results for static graphs. Finally, we investigate properties of random walks on dynamic graphs that are not always connected: we relate their convergence to stationarity to the spectral properties of an average of transition matrices and provide some examples that demonstrate strong discrepancies between static and dynamic graphs.

2012 ACM Subject Classification Theory of computation → Random walks and Markov chains

Keywords and phrases random walks, dynamic graphs, hitting times

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.93

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1903.01342`.

Funding *Thomas Sauerwald*: This project is partially funded by the ERC Starting Grant (DYNAMIC MARCH).
Luca Zanetti: This project is partially funded by the ERC Starting Grant (DYNAMIC MARCH).

1 Introduction

Problem and Motivation. A random walk is a stochastic process on an undirected connected graph $G = (V, E)$. A particle starts on a specified vertex, and then at each time-step $t = 1, 2, \ldots$ it moves to a neighbouring vertex chosen uniformly at random. Random walks have proven to be extremely powerful in the design of various sampling schemes, exploration strategies, and distributed algorithms [26]. They provide a simple yet robust way to explore a large network. Most of the studies on random walks, however, assume the underlying graph to be fixed. In contrast, many prevalent networks today (such as the Internet, social networks, and wireless communication networks) are subject to dramatic changes in their topology over time. Therefore, understanding the theoretical power and limitations of dynamic graphs has been identified as one of the key challenges in computer science [28].

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 93; pp. 93:1–93:15


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

Recently, several works have considered this problem and investigated the behaviour of random walks [3, 4, 34, 12, 25, 32, 33] or similar processes [6, 9, 11, 18, 23] on such dynamic graphs, and their applications to distributed networks [3, 34, 23]. Moreover, rather than a property of the underlying network itself, dynamic graphs may naturally arise in distributed algorithms when communication is performed on a changing, possibly disconnected, subgraph like a spanning-tree or a matching (see, e.g., [7]).

One very popular model is that of an evolving graph, where we consider a sequence of graphs $G^{(1)}, G^{(2)}, \ldots$ over the same set of vertices but with a varying set of edges. This model has been the subject of the majority of previous studies of random walks on dynamic graphs and will be the object of our study as well. Another important feature of dynamic networks is that, with a changing set of edges, the resulting connectivity (i.e., expansion) changes. This might be very common in communication networks, where nodes change their location in space over time and can only communicate if they are within a certain distance of each other. For example, [23] highlights the need to study such evolving graphs with relatively poor connectivity and [28] emphasises the unpredictable nature of fast-changing dynamic networks. To incorporate these features into our model, we will consider evolving graphs with relatively mild assumptions on their connectivity and will not make any restriction on how fast they are changing. Our quantitative analysis is focused on the *mixing time*, the time to converge to the equilibrium distribution, and the *hitting time*, the expected number of steps required by a random walk that starts in a vertex u to reach a vertex v. Analysing the mixing time of dynamic graphs is also useful for load balancing applications, where the mixing time represents the time it takes for all nodes to have (roughly) a load that is proportional to their stationary distribution. Most theoretical studies of load balancing so far assumed the graph to be fixed.

Our Results. The main motivation for our work comes from the results by Avin et al. [4], which describe a remarkable dichotomy with respect to the behaviour of random walks in evolving graphs: while sequences of connected graphs that share the same stationary distribution are guaranteed to have mixing and hitting times polynomial in the size of the graphs, even small incremental changes to the stationary distribution can cause hitting times to become exponential in the worst case. We focus on the first case of this dichotomy and prove that, at least regarding mixing and hitting times, there is essentially no difference in the behaviour of random walks on static and evolving graphs with a time-independent stationary distribution.

Recall that, for static graphs, it is well-known that the worst-case hitting time is $O(n^2)$ for regular graphs and $O(n^3)$ for arbitrary graphs [14, 15]. Quite surprisingly, we can show that something very similar holds in the setting of evolving graphs: our theorem below proves an upper bound of $O(n^2)$ for the mixing and hitting times of regular evolving graphs, which is optimal even for static graphs, an upper bound of $O(n^3)$ for the mixing time of non-regular evolving graphs, which is again optimal even for static graphs, and an $O(n^3 \log n)$ upper bound for the maximum hitting time, which is only a factor of $O(\log n)$ short from the optimal bound on static graphs (simply consider the Barbell graph, i.e., two cliques of size $n/3$ connected by a path of length $n/3$, which has $O(n^3)$ mixing and maximum hitting time). Below, we use $p_{u,v}^{[0,t]}$ for the probability that a random walk started in u is in v after t steps.

▶ **Main Result 1** (restated, see Theorem 3.3 on page 9). *Let $\mathcal{G} = \{G^{(t)}\}_{t=1}^{\infty}$ be a sequence of connected graphs with n vertices, the same stationary distribution π with $\pi_* = \min_u \pi(u)$, and at most m^* edges. Then:*

1. $t_{mix}(\mathcal{G}) = O(n/\pi_*)$,
2. $\left|\frac{p_{u,v}^{[0,t]}}{\pi_v} - 1\right| \lesssim \frac{m^*}{t} + \frac{1}{\pi_* \sqrt{t}}$, *simplifying to* $\left|\frac{p_{u,v}^{[0,t]}}{\pi_v} - 1\right| \lesssim \frac{n}{\sqrt{t}}$ *if all graphs in $\mathcal{G}$ are d-regular,*
3. $t_{hit}(\mathcal{G}) = O(n \log n/\pi_*)$. *Furthermore, if the graphs in $\mathcal{G}$ are d-regular, $t_{hit}(\mathcal{G}) = O(n^2)$.*

We point out that the second statement in the above result is implied by [34, Lemma 4.6] for the class of regular and bounded-degree graphs.

▶ Remark 1.1. In this work, we never explicitly derive upper bounds on the cover time (i.e., the expected time for a random walk to visit all vertices). However, analogous to Matthew's Bound for static graphs [26, Chapter 11.2], all the *stated* upper bounds on hitting times can be converted into upper bounds on cover times at the cost of an additional $O(\log n)$-factor.

▶ Remark 1.2. Unlike static graphs where the gap between cover and hitting time is $O(\log n)$ (thanks to Matthew's Bound), for evolving graphs the gap could be $\Omega(n)$ even if the sequence consists of regular connected graphs. For example, for any $t \leqslant cn \ln n$, let $G^{(t)}$ be a complete graph with n vertices, while for any $t > cn \ln n$, let $G^{(t)}$ be a cycle with n vertices. We can choose the constant c so that, with probability $1 - \Theta(n^{-1})$, every fixed vertex is visited before $cn \ln n$ steps, but with constant nonzero probability, there is at least one unvisited vertex which is at distance $\Omega(n)$ from the location of the walk at step $cn \ln n$. This yields a $\Theta(n)$ maximum hitting time, but a $\Theta(n^2)$ cover time.

▶ Remark 1.3. Since the stationary distribution of a random walk depends only on the degrees of the vertices, having the same stationary distribution means that the degrees in each graph of the sequence are the same up to scaling.

A natural question is of course under which conditions the worst-case bound on the hitting time can be improved. For static graphs, it has been observed that for many regular networks, the hitting time is indeed optimal, i.e., $O(n)$. One very general and unifying condition is the conjecture of Aldous and Fill [1, Open Problem 6.20] stating that for any bounded-degree, d-regular graph, an isoperimetric dimension of $2 + \varepsilon$ is enough for hitting times to be linear (which is as good as possible). Since the isoperimetric dimension is equal to the dimension of grids, it follows that grids of dimension 3 or higher have a linear hitting time, while grids of dimension 2 have a hitting time of $O(n \log n)$.

For static graphs, a positive answer to the above conjecture by Aldous and Fill was first given by [5], and another proof was found by [2]. Both these proofs, however, exploit the connection between hitting times and electrical resistances [8], which is not known to exist for *general* evolving graphs (however, for a special class of *randomly* evolving graphs such a connection has been used in [4, Theorem 18]). Since our techniques for bounding hitting times are more probabilistic in nature and avoid arguments based on electrical networks, we are able to show that the conjecture by Aldous and Fill is true even in a dynamic setting.

▶ **Main Result 2** (restated, see Theorem 4.2 on page 9). *Let $\mathcal{G} = \{G^{(t)}\}_{t=1}^{\infty}$ be a sequence of n-vertex graphs such that each $G^{(t)}$ is regular, has bounded degree, and satisfies the following isoperimetric condition: there exists $\varepsilon \in [0, 1/4]$ such that, for any subset of vertices A with $1 \leqslant |A| \leqslant n/2$, $|E(A, V \setminus A)| = \Omega(|A|^{\frac{1}{2}+\varepsilon})$. Then,*

1. $t_{mix}(\mathcal{G}) = O(n^{1-2\varepsilon})$,
2. $\left|\frac{p_{u,v}^{[0,t]}}{n} - 1\right| = O\left(\frac{1}{t^{1+2\varepsilon}}\right)$,
3. $t_{hit}(\mathcal{G}) = O(n)$ *if* $\varepsilon > 0$, $t_{hit}(\mathcal{G}) = O(n \log n)$ *if* $\varepsilon = 0$.

Note that the isoperimetric condition essentially says that each graph in the sequence must be at least as well-connected as a $(2 + \varepsilon)$-dimensional grid. For $\varepsilon = 0$, we recover the $O(n \log n)$ hitting time for static two-dimensional grids. Both of these cases might be relevant in certain applications of moving wireless devices or robots performing terrain exploration.

The first two results apply to settings where there is a "stable connectivity", but each graph in the sequence may have a relatively poor expansion. The next result applies to scenarios where connectivity is more intermittent, in fact some of the vertices may even be

isolated at some time steps. However, "averaging" over a sufficiently long time window, the graph will not only be connected but might also satisfy some reasonably strong expansion guarantee. In this sense, this model is somewhat related to that of [22], which stipulates the existence of a spanning subgraph over any time-interval of a certain length. More formally, in the next theorem we assume that the random walk is on a sequence $\mathcal{G}$ of graphs with transition matrices $P^{(1)}, P^{(2)}, ...$ and there exists a time-independent distribution π which is stationary for any $P^{(i)}$. We remark that we do not assume connectivity and, therefore, any individual $P^{(i)}$ might have multiple stationary distributions. We assume, however, that there exists a large enough time window t such that, for any $i \geqslant 1$, $\overline{P}^{[i,i+t]} = \frac{1}{t}(P^{(i)} + P^{(i+1)} + \cdots + P^{(i+t)})$ is ergodic with a unique stationary distribution π and spectral gap $\lambda(\overline{P}^{[i,i+t]}) \geqslant \lambda > 0$. We then can show that the distribution of a lazy random walk on $\mathcal{G}$ converges to π at a rate that depends on t and the spectral gap λ. We refer to Section 5 for details on the set-up.

▶ **Main Result 3** (restated, see Corollary 5.3 on page 11)**.** *Consider a dynamically evolving sequence $\mathcal{G} = \{G^{(t)}\}_{t=1}^{\infty}$ of graphs with transition matrices $\{P^{(i)}\}_{i=1}^{\infty}$ such that (1) there exists π which is a stationary distribution for any $P^{(i)}$; and (2) there exists a time-window $t \geqslant 0$ such that, for any $i \geqslant 0$, $\overline{P}^{[i,i+t]} = \frac{1}{t}(P^{(i)} + P^{(i+1)} + \cdots + P^{(i+t)})$ is ergodic and has spectral gap $\lambda\big(\overline{P}^{[i,i+t]}\big) \geqslant \lambda > 0$. Then, $t_{mix}(\mathcal{G}) = O\left(t^2 \log(1/\pi_*)\lambda^{-1}\right)$, where $\pi_* = \min_u \pi(u)$.*

This result is not only significant in the context of dynamically evolving graphs, but also in settings of static graphs where communication is restricted to a bounded-degree subgraph which potentially changes in each round. One prominent example are matching-based communications, where in each round a random matching is generated and only those edges can be used for averaging or exchanging information, e.g., [7].

Even when the assumptions of Main Theorem 3 are satisfied for a *small* time-window t, we cannot always guarantee that hitting and mixing times will be polynomial in the size of the graphs. Indeed, we exhibit examples of dynamic evolving graphs of n vertices that satisfy such conditions but have mixing and/or hitting times that are exponential in n and t. We show in Proposition 5.5 that, since graphs in the sequences need not be connected, it is possible to construct examples where the stationary distribution π has exponentially small probability mass on some vertices. This could result in exponential mixing and hitting times, but somewhat surprisingly also possibly in polynomial mixing time and exponential maximum hitting time. Both of the constructed graph sequences rely on the idea of simulating a directed graph by a sequence of disconnected bipartite graphs. We note that the idea of simulating directed graphs with a sequence of evolving graphs was introduced in [4], where it is shown how to simulate a directed graph by a sequence of connected evolving graphs with *varying* stationary distribution. In contrast, our result rely on simulating dynamic graphs with a sequence of disconnected evolving graphs with the *same* stationary distribution.

A natural question is whether we can relax the assumptions on the regularity or existence of a time-invariant stationary distribution. Unfortunately, we show that this is not always possible. We exhibit in Proposition 5.4 a sequence of graphs which are connected, have bounded-degree and constant spectral gap, but for which t-step probabilities are very far from the uniform distribution even for a time t which is larger than the mixing time of a random walk on any (static) graph in the sequence.

Going back to Main Result 2, the essence behind the proof is that, to achieve an optimal $O(n)$ hitting time, we do not need large sets to have high expansion. What we only need is that small sets have a "sufficiently high" expansion. We derive another result in the same spirit by upper bounding a variational characterisation of the commute time in terms of

some version of the conductance profile [27]. However, since we need to exploit a variational characterisation of the commute time, as opposed to the earlier results, this bound only holds for static graphs. Let C_{st} be the commute time from s to t. Then, we have the following.

▶ **Main Result 4** (restated, see Lemma 6.1 on page 12)**.** *For any static graph $G = (V, E)$ and $s, t \in V$, there exists a labelling of the vertices from 1 to n such that $C_{st} \leqslant 2m \sum_{j=1}^{n-1} |\partial[j]|^{-1}$, where $\partial[j]$ is the set of edges with one endpoint in $\{1, \ldots, j\}$ and one in $\{j+1, \ldots, n\}$.*

Note the relation between Main Theorem 4 and the well-known Nash-Williams' inequality [26, Proposition 9.15] which states that, for every set $\{E_1, E_2, \ldots, E_k\}$ of edge-disjoint cut-sets separating s from t, $C_{st} \geqslant 2m \sum_{j=1}^{k} |E_j|^{-1}$. Our upper bound, however, differs from the Nash-Williams' inequality in two ways: (1) the cut-sets $\partial[j]$ are in general not edge-disjoint; (2) we prove the existence of a "good" labelling, while Nash-Williams holds for *any* labelling.

As an application of this result, we consider the hitting time on d-regular graphs in terms of the edge-connectivity (i.e., the smallest number of edges that would need to be removed to make the graph disconnected), which does not impose any condition on the expansion of large sets.

▶ **Main Result 5** (restated, see Theorem 6.3 on page 13)**.** *Let $G = (V, E)$ be any static d-regular graph with edge-connectivity ρ. Then $t_{hit}(\mathcal{G}) \leqslant O(n^2 \cdot (\frac{\log d}{d} + \frac{1}{\rho}))$. In particular, since $\rho \leqslant d$, we get the simpler (but potentially slightly weaker) upper bound $t_{hit}(\mathcal{G}) = O(n^2 \log d/\rho)$.*

We remark that in Aldous and Fill [1, Proposition 6.22], it was shown that for any d-regular graph G which is ρ-edge-connected, the maximum hitting time is $O(n^2 d \cdot \rho^{-3/2})$. They also mention that if the graph is $\Omega(d)$-edge-connected, they obtain a bound of $O(n^2 \cdot d^{-1/2})$. For this case of maximal edge-connectivity, $\rho = \Theta(d)$, our bound is considerably better than the one by Aldous and Fill, and, modulo the $\log d$-factor, gives also the right dependency on d. In particular, we demonstrate in Section 6 that the dependency on the edge-connectivity ρ is as good as possible (neglecting logarithmic factors) in the sense that for any pair of ρ and d, there exists a d-regular graph with edge-connectivity ρ which matches the upper bound in Main Result 5 (Theorem 6.3) up to constant factors.

Further Related Work. While in this work we focus on standard (lazy) random walks on graphs, we should point out that previous work has established an alternative in form of the so-called *max-degree walk* [4, 12]. In this random walk variant, a large loop probability depending on the degree of the current vertex and (an estimate of) the maximum degree Δ is added. With this modification, the stationary distribution of each graph is identical (and uniform), which makes the analysis of this walk easier. However, one downside of this approach is that it either requires a good estimate of Δ (or even n), or the random walk may potentially be slowed down significantly. Also, studying standard random walks seems more natural and, as we will see later, it also helps us to uncover some of the subtle boundaries between fast mixing and polynomial hitting, and slow mixing and exponential hitting.

One of the earliest appearances of dynamic graphs is in the context of load balancing [17], where the authors assumed a uniform (i.e., time-independent) lower bound on the edge and vertex expansion. A refinement is to instead relate the balancing (mixing) time to the geometric mean of the spectral gaps, which was used in [13]. A result of a similar flavour for both the conductance and the vertex expansion was shown in [18] in the context of randomised rumour spreading, and more recently a similar result was shown for the voter model [6]. In [22], the authors analyse a sequence of graphs satisfying a T-interval

connectivity property, which asserts that for every T consecutive rounds there exists a stable connected spanning subgraph. The authors present upper bounds for several distributed computational problems.

One specific graph model that has been very popular is the so-called Markovian evolving graph. In this model every edge is associated to the same but independent two-state birth and death chain which decides whether the edge is present or not in the next step. Many aspects of this network have been studied, most notably the (dynamic) diameter [10] and the time to spread a piece of information [11]. Recently, however, Lamprou et al. [25] also considered the cover time of these graphs. In particular, suppose there exists an underlying graph G with minimum degree δ such that at each time t the graph $G^{(t)}$ contains each edge in G independently with probability p (i.e., the presence of an edge does not depend on the past). They show the cover time of such dynamic graph is at most $t_{\mathsf{cov}}(G)/(1-(1-p)^{\delta})$, where $t_{\mathsf{cov}}(G)$ is the cover time of G. They also study *random walks with a delay*, where at each step a particle chooses a random neighbour of the current vertex according to the topology of the underlying graph G, and moves there if the corresponding edge is present, otherwise waits till it becomes available. For this perhaps slightly less natural process, they give bounds on the cover time also for the case where the probability of an edge being available at time t depends on whether that edge was available at time $t-1$. We also highlight dynamical percolation, a particular type of Markovian evolving graphs that has received recent attention (see, e.g., [20, 30, 36]). Here, an "open" edge becomes "close" with probability p, while a close edge becomes open with probability $1-p$. In contrast to the literature above, however, works on random walks on dynamical percolation usually refer to continuous-time random walks.

Another class of dynamic graph models involves agents that move in some bounded space and can interact only if they are close enough [24, 29, 31]. In contrast to these works, our bounds are less tight but hold under much weaker assumptions on the graph and therefore capture a more dynamic and less "regular" setting.

Finally, we mention that Saloff-Coste and Zuniga [32, 33] have generalised spectral and geometric techniques, such as Nash and log-Sobolev inequalities, to time-inhomogeneous Markov Chains (of which random walks on dynamic graphs are a subset). In particular, in contrast to our results, they study chains where the individual transition matrices might not have the same time-independent stationary distribution. For this reason, they focus on *merging* properties of these chains, i.e., the ability of the chain to "forget" the initial distribution. They obtain bounds on merging for chains that satisfy the c-stability property, which implies (but it is not equivalent) that the stationary distributions of the individual transition matrices do not change too much over time. Unfortunately, proving that a time-inhomogeneous chain is c-stable is itself very difficult, and they are able to obtain concrete bounds on merging only for very simple time-inhomogeneous Markov chains.

2 Notation and preliminaries

Let $\mathcal{G} = \{G^{(t)}\}_{t=1}^{\infty}$ be an infinite sequence of undirected and unweighted graphs defined on the same vertex set V, with $|V| = n$. We study (lazy) random walks on $\mathcal{G}$: suppose that at a time $t \geqslant 0$ a particle occupies a vertex $u \in V$. At step $t+1$ the particle will remain at the same vertex u with probability $1/2$, or will move to a random neighbour of u in $G^{(t)}$. In other words, it will perform a single random walk step according to a transition matrix $P^{(t)}$, which is the transition matrix of a lazy random walk on $G^{(t)}$: $P^{(t)}(u,u) = 1/2$, $P^{(t)}(u,v) = 1/(2d_u)$ if there is an edge between u and v in $G^{(t)}$ (and in this case we write

$u \sim_t v$), or $P^{(t)}(u,v) = 0$ otherwise. We denote with $p_{u,v}^{[t_1,t_2]}$ the probability that a random walk that visits vertex $u \in V$ at time t_1 will visit $v \in V$ at time $t_2 \geqslant t_1$. Notice that given an initial probability distribution $p^{(0)} : V \to [0,1]$, $p^{[0,t]} = p^{(0)}P^{[0,t]} = p^{(0)}P^{(1)}P^{(2)}\cdots P^{(t)}$ is the probability distribution after t steps.

Unless stated otherwise we assume that all the graphs in $\mathcal{G}$ are connected and have the same stationary distribution π, i.e., $\pi P^{(t)} = \pi$ for any $t \geqslant 0$. We denote the smallest value assumed by π as $\pi_* = \min_{x\in V} \pi(x)$. We define the $\ell_2(\pi)$-inner product as $\langle f, g\rangle_\pi = \sum_{u\in V} f(u)g(u)\pi(u)$ for any $f, g : V \to \mathbb{R}$. Analogously, we denote with $\|f\|_{2,\pi} = \sqrt{\langle f, f\rangle_\pi}$ the $\ell_2(\pi)$-norm of $f : V \to \mathbb{R}$. Notice that since all the graphs in $\mathcal{G}$ are undirected, for any $t \geqslant 0$, $P^{(t)}$ is reversible with respect to π, i.e., $\pi(x)P^{(t)}(x,y) = \pi(y)P^{(t)}(y,x)$ for any $x, y \in V$ (this is also called the detailed balance condition). Moreover, $P^{(t)}$ is self-adjoint for the $\ell_2(\pi)$-inner product: for any $f, g : V \to \mathbb{R}$,

$$\langle P^{(t)}f, g\rangle_\pi = \langle f, P^{(t)}g\rangle_\pi. \tag{1}$$

We will often work with the likelihood ratio $\rho_{u,\cdot}^{[0,t]} = p_{u,\cdot}^{[0,t]}/\pi(\cdot)$. When it is clear from the context, we will drop the starting point u and use the shorthands $p^{(t)}$ and $\rho^{(t)}$ to indicate (respectively) the probability distribution of the random walk at time t and its likelihood ratio. We define the ℓ_2 mixing time as

$$t_{\mathsf{mix}}(\mathcal{G}) = \min\{t\colon \|\rho_{u,\cdot}^{[0,t]} - 1\|_{2,\pi} \leqslant 1/3 \text{ for any } u \in V\}.$$

Observe that, since $\mathbb{E}_\pi \rho^{(t)} = 1$, we have that $\|\rho^{(t)} - 1\|_{2,\pi}^2 = \mathrm{Var}_\pi \rho^{(t)} = \mathbb{E}_\pi \left(\rho^{(t)}\right)^2 - 1$.

Let p be a probability distribution with likelihood ratio $\rho = p/\pi$. For a reversible P,

$$P\rho(u) = \sum_{v\in V} P(u,v)\rho(v) = \sum_{v\in V} P(u,v)\frac{p(v)}{\pi(v)} = \frac{1}{\pi(u)}\sum_{v\in V} P(v,u)p(v) = \frac{pP(u)}{\pi(u)},$$

from which it follows that $P^{(t)}\cdots P^{(1)}\rho^{(0)}(u) = \rho^{(t)}(u)$.

Given a transition matrix P with stationary distribution π and a function $f\colon V \to \mathbb{R}$, we define the Dirichlet form as

$$\mathcal{E}_P(f,f) = \frac{1}{2}\sum_{u,v\in V} (f(u) - f(v))^2\,\pi(u)P(u,v).$$

When P is a transition matrix of a lazy random walk on a graph $G = (V,E)$ with $|E| = m$, $\mathcal{E}_P(f,f) = \frac{1}{4m}\sum_{u\sim v}(f(u) - f(v))^2$, where $u \sim v$ stands for $\{u,v\} \in E$. As long as P is lazy (i.e., $P(u,u) \geqslant 1/2$ for any $u \in V$), we can relate the ℓ_2^2 distance of a distribution from stationary to its Dirichlet form [16, Proposition 2.5]:

$$\mathrm{Var}_\pi \rho^{(t)} \geqslant \mathrm{Var}_\pi \rho^{(t+1)} + \mathcal{E}_{P^{(t+1)}}(\rho^{(t)}, \rho^{(t)}). \tag{2}$$

The *spectral gap* of P is defined as

$$\lambda(P) = \inf_{\substack{f:V\to\mathbb{R}\\ \mathrm{Var}_\pi f\neq 0}} \frac{\mathcal{E}_P(f,f)}{\mathrm{Var}\, f}.$$

We denote with $\Phi_P(A)$ the *conductance* of a subset of vertices $A \subset V$:

$$\Phi_P(A) = \frac{\sum_{u\in A, v\notin A}\pi(u)P(u,v)}{\min\{\pi(A), \pi(V\setminus A)\}},$$

where $\pi(A) = \sum_{u \in A} \pi(u)$. The conductance of P is then defined as $\Phi(P) = \min_{A \subset V} \Phi_P(A)$. Cheeger's inequality [35] relates $\lambda(P)$ to the conductance $\Phi(P)$ of a reversible P: $2\Phi(P) \geqslant \lambda(P) \geqslant \Phi(P)^2/2$.

Given two vertices u and v, we denote with $\tau_{u,v}$ the *hitting time* of v from u, i.e., the expected time to reach v starting from u. The maximum hitting time is defined as $t_{\mathsf{hit}}(\mathcal{G}) = \max_{u,v} \tau_{u,v}$. The *commute time* between u and v, denoted by $C_{u,v}$, is defined as the expected time for a random walk starting from u, to reach v and then return to u.

Finally, we write $A \lesssim B$, respectively $A \gtrsim B$, to mean that there exists some absolute constant $C > 0$, independent of the parameters of the sequence of graphs $\mathcal{G}$, such that $A \leqslant C \cdot B$, respectively $A \geqslant C \cdot B$.

3 Worst-case bounds for mixing and hitting times

In this section we assume that a particle performs a random walk on a sequence of graphs $\mathcal{G} = \{G^{(t)}\}_{t=1}^{\infty}$ where all the $G^{(t)}$ share the same set of n vertices V, are connected, and have a time-independent stationary distribution π with $\pi_* = \min_u \pi(u)$. In general, graphs in the sequence might have a different number of edges, but the ratio degree of a node over total number of edges remains the same. We denote with $m^* \leqslant n^2$ the maximum number of edges a graph in the sequence can have.

Our goal is to bound mixing and maximum hitting times of a random walk on $\mathcal{G}$. We start by studying the rate of convergence to stationarity. By equation (2), our goal then becomes to study $\operatorname{Var} \rho^{(t)} \leqslant \operatorname{Var} \rho^{(0)} - \sum_{i=1}^{t-1} \mathcal{E}_{P^{(i)}}(\rho^{(i-1)}, \rho^{(i-1)})$. The next lemma provides a lower bound on the Dirichlet form of graphs in $\mathcal{G}$. The main insight of this lemma is that it shows a faster decrease of the ℓ_2-distance to stationarity when this distance is large, i.e., at the beginning of the walk. This is in the same vein as, for example, bounds on mixing based on the spectral profile [19].

▶ **Lemma 3.1.** *Let P be the transition matrix of a lazy random walk on a graph $G \in \mathcal{G}$. Given a probability distribution $\sigma : V \to [0,1]$ with likelihood ratio $f = \sigma/\pi$ such that $\operatorname{Var}_\pi f = \varepsilon > 0$,*

$$\mathcal{E}_P(f,f) \gtrsim \max\left\{\frac{\varepsilon^2}{m^* + 1/(\pi_*^2(1+\varepsilon))}, \frac{\pi_* \varepsilon^2}{n}\right\}.$$

While the previous lemma will be directly used to derive bounds on mixing, to obtain a bound on the hitting time we will need to study t-step probabilities. For this reason, we prove a technical lemma that relates $2t$-step probabilities to the variance of the likelihood ratio of a t-step probability distribution, generalising a well-known result for time-homogeneous reversible Markov chains (see, e.g., [1, Lemma 3.20]). We remark, however, that while in time-homogeneous Markov chains $2t$-step transition probabilities will be as small as the variance of their t-step likelihood ratio, in our case, since the order in which transition matrices are applied can matter significantly, this might not be necessarily true: we can only relate these probabilities to the variance of the t-step likelihood ratio of a related but slightly different Markov chain.

▶ **Lemma 3.2.** *Let $t_1 < t_2$. Then, for any $u, v \in V$, it holds that*

$$\left|\rho_{v,u}^{[t_1,t_2]} - 1\right| \leqslant \max\left\{\operatorname{Var}_\pi\left(P^{(\lfloor \frac{t_1+t_2}{2} \rfloor)} \cdots P^{(t_2)} \rho_{u,\cdot}^{[t_1,t_1]}\right), \operatorname{Var}_\pi\left(P^{(\lfloor \frac{t_1+t_2}{2} - 1 \rfloor)} \cdots P^{(1)} \rho_{v,\cdot}^{[t_1,t_1]}\right)\right\}.$$

Using Lemma 3.1 and Lemma 3.2 we can obtain almost optimal worst case bounds on mixing, hitting, and t-step probabilities of a random walk on $\mathcal{G}$. In particular, when $\mathcal{G}$ comprises only regular graphs, the next theorem implies a $O(n^2)$ bound on mixing and

hitting times, which matches the well-known results for a random walk on a static undirected graph. In the general non-regular case, we prove a $O(n^3)$ bound on mixing and a $O(n^3 \log n)$ bound on hitting, which almost matches the $O(n^3)$ bound for mixing and hitting in static graphs. This improves upon [4], which presents a bound of $O(n^3 \log n)$ for hitting on regular graphs and a bound of $O(n^5 \log n)$ for hitting in the general case.

▶ **Theorem 3.3.** *Let $\mathcal{G}$ be a sequence of connected graphs with n vertices, the same stationary distribution π, and at most m^* edges in each graph. Then, for a lazy random walk on $\mathcal{G}$:*

1. $t_{mix}(\mathcal{G}) = O(n/\pi_*)$,
2. $\left|\frac{p_{u,v}^{[0,t]}}{\pi_v} - 1\right| \lesssim \frac{m^*}{t} + \frac{1}{\pi_* \sqrt{t}}$, *simplifying to* $\left|\frac{p_{u,v}^{[0,t]}}{\pi_v} - 1\right| \lesssim \frac{n}{\sqrt{t}}$ *if all the graphs in $\mathcal{G}$ are d-regular,*
3. $t_{hit}(\mathcal{G}) = O(n \log n/\pi_*)$. *Furthermore, if the graphs in $\mathcal{G}$ are d-regular, $t_{hit}(\mathcal{G}) = O(n^2)$.*

The proof, which is omitted here, proceeds roughly as follows. First we establish the bound on the mixing time based on Lemma 3.1, which readily implies that starting from a distance to stationarity equal to ε, such distance is halved in $O(n/(\varepsilon\pi_*))$ steps. We then connect the distance to stationarity to t-step probabilities with Lemma 3.2, obtaining the second result of Theorem 3.3. Finally, to bound the hitting time, we employ a probabilistic argument already exploited in, e.g., [21], and which makes use of both our bounds on mixing time and on t-step probabilities.

4 Bounds on hitting times based on the isoperimetric dimension

Aldous and Fill conjectured in their book [1, Open Problem 6.20] that whenever a regular bounded-degree graph satisfies $|E(A, A^c)| = \Omega(|A|^{\frac{1}{2}+\varepsilon})$ for any small positive ε, the maximum hitting time should be $O(n)$. Observe that this isoperimetric condition is satisfied by the torus in 3 or higher dimensions, which has indeed $O(n)$ maximum hitting time. Furthermore, to have $O(n)$ maximum hitting time, ε needs to be strictly greater than zero: take for example the 2-dimensional torus: there is a set A for which $|E(A, A^c)| = \Theta(|A|^{1/2})$ and, indeed, the maximum hitting time is $\Theta(n \log n)$.

The conjecture was first proved in [5], with a proof based on the relation between commute times and effective resistances in a graph. Since a similar relation is not known for time-inhomogeneous Markov chains, such a proof cannot be generalised to random walks on dynamic graphs. In this section we present a new proof of this result based on the "conditional expectation trick" already used in the proof of Theorem 3.3. We start by obtaining a bound on the Dirichlet form of a graph satisfying the aforementioned isoperimetric condition.

▶ **Lemma 4.1.** *Let $G = (V, E)$ be a d-regular undirected graph with $|V| = n$ and $d = O(1)$ such that, for any $A \subset V$ with $1 \leqslant |A| \leqslant n/2$, $|E(A, V \setminus A)| = \Omega(|A|^{\frac{1}{2}+\varepsilon})$ for $1/4 \geqslant \varepsilon \geqslant 0$. Consider the transition matrix P of a lazy random walk in G. Let σ be any probability distribution and $f = \sigma/\pi$, where π is the uniform distribution. If $\mathbb{E}_\pi f^2 = \beta > C$ for a large enough constant C, then*

$$\mathcal{E}_P(f, f) \gtrsim \frac{\beta^{2-2\varepsilon}}{n^{1-2\varepsilon}}.$$

We now apply the previous lemma to prove the main result of this section in an analogous way to the proof of Theorem 3.3.

▶ **Theorem 4.2.** *Let $\mathcal{G} = \{G^{(t)}\}_{t=1}^{\infty}$ be a sequence of n-vertex graphs such that each $G^{(t)}$ is regular, has bounded degree, and satisfies the following isoperimetric condition: there exists $\varepsilon \in [0, 1/4]$ such that, for any subset of vertices A with $1 \leqslant |A| \leqslant n/2$, $|E(A, V \setminus A)| = \Omega(|A|^{\frac{1}{2}+\varepsilon})$. Then,*

1. $t_{mix}(\mathcal{G}) = O(n^{1-2\varepsilon})$,
2. $\left|\frac{p_{u,v}^{[0,t]}}{n} - 1\right| = O\left(\frac{1}{t^{1+2\varepsilon}}\right)$,
3. $t_{hit}(\mathcal{G}) = O(n)$ *if* $\varepsilon > 0$, $t_{hit}(\mathcal{G}) = O(n \log n)$ *if* $\varepsilon = 0$.

5 Bounds on mixing based on average transition probabilities

Unlike in the time-homogeneous case, eigenvalues of the individual transition matrices of a time-inhomogenous Markov chain are not necessarily indicative of its mixing time, even when there exists a unique time-independent stationary distribution. An emblematic example is the following: consider a sequence of graphs $\mathcal{G} = \{G^{(t)}\}_{t=1}^{\infty}$ defined over a vertex set $V = \{1, \ldots, 2n\}$ such that, at odd t, $G^{(t)}$ is the union of two expanders (graphs with constant spectral gap), one over $\{1, \ldots, n\}$, the other over $\{n+1, \ldots, 2n\}$, while at even t, $G^{(t)}$ is a perfect matching between $\{1, \ldots, n\}$ and $\{n+1, \ldots, 2n\}$. Since all the graphs are disconnected, each transition matrix has spectral gap equals to 0, and eigenvalue bounds are, in this case, useless to analyse convergence to stationarity. On the other hand, it is quite clear that a lazy random walk on $\mathcal{G}$ mixes in $\Theta(\log n)$ time.

A more precise way to study mixing in time-inhomogeneous random walks would be to consider the spectral gap of the product of the transition matrices $P^{(1)} \cdots P^{(t)}$. Unfortunately, spectral bounds for the product of matrices are notoriously hard to come by. What is significantly easier is to study the *average* transition matrix $\overline{P} = \frac{1}{t}\left(P^{(1)} + P^{(2)} + \cdots + P^{(t)}\right)$, which at least does not depend on the order in which the transition matrices appear. For this reason, in this section we give bounds on mixing on $\mathcal{G}$ that depend on the Dirichlet form of $\overline{P}$. In particular, consider the aforementioned example where $G^{(t)}$ is two disjoint expanders at odd times, and a perfect matching between the two sets at even times. Consider the average transition matrix $\overline{P} = \frac{1}{2}\left(P^{(\ell)} + P^{(\ell+1)}\right)$ for any two consecutive steps $\ell, \ell+1$: $\overline{P}$ is just the transition matrix of a random walk on an expander graph defined over the entire set of vertices. Our results, then, make us easily derive the correct bound $t_{\mathsf{mix}}(\mathcal{G}) = O(\log n)$.

Throughout this section we assume that $\mathcal{G} = \{G^{(t)}\}_{t=1}^{\infty}$ is a sequence of undirected graphs over a vertex set V with $|V| = n$. The graphs are not necessarily connected, which means they might have multiple stationary distributions. We require, however, that there exists a time-independent distribution π which is a stationary distribution for all the graphs in $\mathcal{G}$. Fixing a time interval $[t_1, t_2]$, we consider $\overline{P} = \frac{1}{t_2 - t_1}\left(P^{(t_1)} + \cdots + P^{(t_2)}\right)$. We consider time intervals for which $\overline{P}$ is irreducible. Note since the transition matrices $\{P^{(i)}\}_i$ are strongly aperiodic and reversible with respect to π, so is $\overline{P}$. Therefore, we can always assume that $\overline{P}$ is ergodic and has a unique stationary distribution π, unlike the individual matrices $P^{(i)}$.

For simplicity, we assume in our proofs that each graph in $\mathcal{G}$ has the same number of edges m. Our results, however, also hold for sequences of graphs with different edges densities.

Notice that, by the detailed balance condition, if $u \sim_i v$ for some step i, $\pi(u)/\pi(v) = d_u/d_v$, where d_u and d_v are, respectively, the (time-independent) degrees of u and v[1]. In particular, this means there exists some $\alpha_u \geqslant 0$, which is independent from t, such that $\pi(u) = \alpha_u d_u / 2m$ and $\pi(v) = \alpha_u d_v / 2m$.

[1] it may happen that u is isolated in some round i, leading to u having degree 0 in that round. However, in that case, u can be safely ignored when computing $\mathcal{E}_{P^{(i)}}$. Hence, because the stationary distribution is always the same and so is the number of edges, we may assume that the degree of u is always d_u

▶ **Lemma 5.1.** *Let $p^{(0)}$ be an arbitrary initial probability distribution, and $\rho^{(0)} = p^{(0)}/\pi$. Suppose that for some $t \geqslant 1$ and $u \in V$, $|\rho^{(t)}(u) - \rho^{(0)}(u)| \geqslant \varepsilon > 0$. Then,*

$$\operatorname{Var}_\pi \rho^{(0)} - \operatorname{Var}_\pi \rho^{(t)} \geqslant \frac{\alpha_u}{4m} \sum_{i=1}^{t} \sum_{v \sim_i u} \left(\rho^{(i-1)}(u) - \rho^{(i-1)}(v)\right)^2 \geqslant \frac{2\varepsilon^2 \pi(u)}{t}.$$

We are now able to relate $\operatorname{Var}_\pi \rho^{(0)} - \operatorname{Var}_\pi \rho^{(t)}$ to $\mathcal{E}_{\overline{P}}(\rho^{(0)}, \rho^{(0)})$. The proof of the next theorem works roughly as follows. We divide the vertices in two classes: U contains all the vertices for which there exists an $1 \leqslant i \leqslant t-1$ such that $\rho^{(i)}(u)$ differs *significantly* from $\rho^{(0)}(u)$, while $V \setminus U$ contains the rest. We then use Lemma 5.1 to lower bound the contribution given by vertices in U to $\operatorname{Var}_\pi \rho^{(0)} - \operatorname{Var}_\pi \rho^{(t)}$. Since for $u \notin U$, $\rho^{(i)}(u)$ has not changed much from $\rho^{(0)}(u)$, we can instead directly lower bound its contribution to $\operatorname{Var}_\pi \rho^{(0)} - \operatorname{Var}_\pi \rho^{(t)}$ just looking at its contribution to $\mathcal{E}_{\overline{P}}(\rho^{(0)}, \rho^{(0)})$.

▶ **Theorem 5.2.** *Given a time interval of length t labelled $[1, t]$, let $\overline{P} = \frac{1}{t}(P^{(1)} + P^{(2)} + \cdots + P^{(t)})$ with spectral gap $\lambda(\overline{P})$. Then, for any initial probability distribution $p^{(0)}$ with likelihood $\rho^{(0)} = p^{(0)}/\pi$, it holds that*

$$\operatorname{Var}_\pi \rho^{(0)} - \operatorname{Var}_\pi \rho^{(t)} \geqslant \frac{1}{15t} \mathcal{E}_{\overline{P}}(\rho^{(0)}, \rho^{(0)}) \geqslant \frac{\lambda(\overline{P})}{15t}.$$

We remark we do not know if the dependency of t in the bound of Theorem 5.2 (which appears as a result of an application of the Cauchy-Schwarz inequality) is tight, or even if any dependency on t is needed at all.

From Theorem 5.2 it is easy to derive the following corollary:

▶ **Corollary 5.3.** *Given a lazy random walk on a sequence $\mathcal{G}$ of graphs with transition matrices $\{P^{(i)}\}_{i=1}^{\infty}$ such that (1) there exists π which is a stationary distribution for any $P^{(i)}$; (2) a time-window $t \geqslant 0$ such that, for any $i \geqslant 0$, $\overline{P}^{[i,i+t]} = \frac{1}{t}(P^{(i)} + P^{(i+1)} + \cdots + P^{(i+t)})$ is ergodic and has spectral gap $\lambda\big(\overline{P}^{[i,i+t]}\big) \geqslant \lambda > 0$. Then, $t_{mix}(\mathcal{G}) = O\big(\frac{t^2 \log(1/\pi_*)}{\lambda}\big)$.*

To highlight the applicability of Corollary 5.3, consider a sequence of connected graphs $\mathcal{G}$ with time-independent stationary distribution π in which, for any interval of t consecutive steps and subset of vertices A, there exists a transition matrix $P^{(i)}$ of a graph in the interval such that $\Phi_{P^{(i)}}(A) \geqslant \phi$. Then, $\Phi\left(\overline{P}\right) \geqslant \phi/t$ and $\lambda\left(\overline{P}\right) \geqslant \phi^2/t^2$. Hence, Corollary 5.3 gives us that $t_{\mathsf{mix}}(\mathcal{G}) = O(t^3 \log n/\phi^2)$.

Another natural question is whether our condition on the stationary distribution being fixed could be relaxed. This question is answered negatively by the following result:

▶ **Proposition 5.4.** *For any $t = \omega(\log n)$, there is a sequence of connected n-vertex bounded-degree expander graphs $\mathcal{G} = \{G^{(i)}\}_{i=1}^{\infty}$ and a constant $c > 0$ so that $p_{u,v}^{(t)} \geqslant n^{-1+c}$ for some vertices u and v.*

In Section 3 and Section 4 we have shown that the behaviour of a lazy random walk on a sequence of *connected* graphs with the same stationary distribution is comparable to the behaviour of random walks on static graphs, at least regarding mixing and hitting times. When the graphs are disconnected, however, the behaviour of random walks on dynamic graphs becomes more complicated. Theorem 5.2 shows that, if every t steps the average of the transition matrices applied in those steps is irreducible and strongly aperiodic with stationary distribution π, then the random walk will converge to π. However, π can be highly imbalanced and, as a result, mixing and hitting can be exponential in t and the number of vertices n. The next proposition shows an example of this behaviour.

▶ **Proposition 5.5.** *There is a sequence of n-vertex bounded-degree graphs $\mathcal{G} = \{G^{(i)}\}_{i=1}^{\infty}$ with transition matrices $\{P^{(i)}\}_{i=1}^{\infty}$ and a probability distribution π such that (1) for any i, π is stationary for $P^{(i)}$; (2) the average transition matrix $\overline{P}$ of any $4n$ consecutive steps is ergodic; (3) for any $t \geqslant 0$ there are two vertices u, v such that $p_{u,v}^{[0,t]} \leqslant 2^{-(n/4)-2}$. Moreover, $t_{mix}(\mathcal{G}) = O(\text{poly}(n))$, while $t_{hit}(\mathcal{G}) = 2^{\Omega(n)}$. There is also a sequence $\mathcal{G}'$ satisfying (1), (2), and (3) such that $t_{mix}(\mathcal{G}) = 2^{\Omega(n)}$.*

6 Bounds in terms of average edge connectivity

Recall that in Section 4 we proved several bounds which hold for graphs with sufficient expansion for small sets of vertices. Following a different direction, we now derive bounds on commute times for random walks on d-regular *static* graphs based on *average* connectivity measures (see the end of Section 2 for some basic relations between the commute time and hitting time). We assume $G = (V, E)$ is a connected, undirected and static graph with vertex set $V = \{1, \ldots, n\}$ and m edges. We denote with P the transition matrix of a lazy random walk on G and π its stationary distribution. Given A, B, the probability flow between A and B is defined as $\sum_{u \in A} \sum_{v \in B} \pi(u)P(u, v)$. The edge boundary of A, denoted with ∂A, is the set of edges with one endpoint in A and one in $V \setminus A$. For ease of notation we define $[i] = \{1, \ldots, i\}$. Also recall that we denote with C_{st} the expected commute time between s and t. We will use the following variational characterisation of the average commute time (see Aldous and Fill, [1, Theorem 3.36]):

$$C_{st} = \max_{g\colon V \to \mathbb{R}} \{1/\mathcal{E}_P(g, g)\colon 0 \leqslant g \leqslant 1, g(s) = 0, g(t) = 1\}. \tag{3}$$

▶ **Lemma 6.1.** *For any graph $G = (V, E)$ and $s, t \in V$, there exists a labelling of the vertices from 1 to n such that*

$$C_{st} \leqslant 2m \sum_{j=1}^{n-1} \frac{1}{|\partial[j]|}.$$

Furthermore, by considering the reversal of the labelling, we can also conclude that $C_{st} \leqslant 4m \sum_{j=1}^{n/2} \frac{1}{|\partial[j]|}$.

Note that the well-known Nash-Williams's inequality [26, Proposition 9.15] gives a very similar lower bound: it states that for every set of edge-disjoint cutsets separating s from t, $\{E_1, E_2, \ldots, E_k\}$, $C_{st} \geqslant 2m \sum_{j=1}^{k} \frac{1}{|E_j|}$. Note that in our upper bound however, the cutsets $\partial[j]$ are in general not edge-disjoint.

Finally, it can be shown that the lemma above holds even for a labelling such that the subgraph induced by $[i]$ is connected for every $1 \leqslant i \leqslant n$.

6.1 Commute times and edge-connectivity

We now apply Lemma 6.1 to obtain a bound on commute times that depends on the edge-connectivity of the graph, improving a result by Aldous and Fill [1, Proposition 6.22].

▶ **Lemma 6.2.** *Let $G = (V, E)$ be any graph with minimum degree δ so that any subset $S \subseteq V$ with $1 \leqslant |S| \leqslant n - 1$ satisfies $|\partial S| \geqslant \rho$ (in other words, G has edge-connectivity at least ρ). Then we have*

$$\sum_{i=1}^{n-1} \frac{1}{|\partial[i]|} = O(n/\delta^2 \cdot \log \delta + n/(\rho\delta)).$$

▶ **Theorem 6.3.** *Let $G = (V, E)$ be any graph with minimum degree δ, average degree $\overline{d}$ and edge-connectivity ρ. Then any commute time is bounded by $O(n^2\overline{d} \cdot (\frac{\log \delta}{\delta^2} + \frac{1}{\delta\rho}))$.*

We remark that Aldous and Fill [16, Proposition 6.22] proved that for any graph G with average degree $\overline{d}$ which is ρ-edge-connected, the maximum commute time is $O(n^2\overline{d} \cdot \rho^{-3/2})$. They also mention that if the graph is $\Omega(d)$-edge-connected, they obtain a bound of $O(n^2 \cdot d^{-1/2})$. For this case of maximal edge-connectivity, $\rho = \Theta(d)$, our bound is considerably better than the one by Aldous and Fill, and, modulo the $\log d$-factor, gives also the correct dependency on d. Furthermore, since the edge-connectivity ρ satisfies $\rho \leqslant \delta \leqslant d$, it is easy to verify that our bound is never worse than the bound in Aldous and Fill. In fact, as soon as $\delta \to \infty$, our upper bound will be asymptotically smaller than the bound by Aldous and Fill.

▶ Remark 6.4. For any pair of ρ and d there is a graph matching the upper bound in Theorem 6.3 up to a factor of $O(\log d)$.

References

1. David Aldous and Jim Fill. Reversible Markov chains and random walks on graphs. unpublished monograph, 2002.
2. Nima Anari and Shayan Oveis Gharan. Effective Resistance and Simple Random Walks. unpublished, 2014. URL: `https://homes.cs.washington.edu/~shayan/courses/cse599/adv-approx-4.pdf`.
3. John Augustine, Gopal Pandurangan, and Peter Robinson. Distributed Algorithmic Foundations of Dynamic Networks. *SIGACT News*, 47(1):69–98, 2016.
4. Chen Avin, Michal Koucký, and Zvi Lotker. Cover time and mixing time of random walks on dynamic graphs. *Random Structures Algorithms*, 52(4):576–596, 2018.
5. Itai Benjamini and Gady Kozma. A resistance bound via an isoperimetric inequality. *Combinatorica*, 25(6):645–650, 2005.
6. Petra Berenbrink, George Giakkoupis, Anne-Marie Kermarrec, and Frederik Mallmann-Trenn. Bounds on the voter model in dynamic networks. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 146:1–146:15, July 12–15 2016.
7. Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE Trans. Inform. Theory*, 52(6):2508–2530, 2006.
8. Ashok K. Chandra, Prabhakar Raghavan, Walter L. Ruzzo, Roman Smolensky, and Prasoon Tiwari. The electrical resistance of a graph captures its commute and cover times. *Comput. Complexity*, 6(4):312–340, 1996/97.
9. Andrea E. F. Clementi, Pierluigi Crescenzi, Carola Doerr, Pierre Fraigniaud, Francesco Pasquale, and Riccardo Silvestri. Rumor spreading in random evolving graphs. *Random Struct. Algorithms*, 48(2):290–312, 2016.
10. Andrea E. F. Clementi, Claudio Macci, Angelo Monti, Francesco Pasquale, and Riccardo Silvestri. Flooding Time of Edge-Markovian Evolving Graphs. *SIAM J. Discrete Math.*, 24(4):1694–1712, 2010.
11. Andrea E. F. Clementi, Riccardo Silvestri, and Luca Trevisan. Information spreading in dynamic graphs. *Distributed Computing*, 28(1):55–73, 2015.
12. Oksana Denysyuk and Luís E. T. Rodrigues. Random Walks on Evolving Graphs with Recurring Topologies. In *Distributed Computing - 28th International Symposium, DISC 2014, Austin, TX, USA, October 12-15, 2014. Proceedings*, pages 333–345, 2014.
13. Robert Elsässer, Burkhard Monien, and Stefan Schamberger. Load Balancing in Dynamic Networks. In *7th International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN 2004), 10-12 May 2004, Hong Kong, SAR, China*, pages 193–200, 2004.
14. Uriel Feige. A Tight Lower Bound on the Cover Time for Random Walks on Graphs. *Random Struct. Algorithms*, 6(4):433–438, 1995. `doi:10.1002/rsa.3240060406`.

15 Uriel Feige. A Tight Upper Bound on the Cover Time for Random Walks on Graphs. *Random Struct. Algorithms*, 6(1):51–54, 1995. doi:10.1002/rsa.3240060106.

16 James Allen Fill. Eigenvalue bounds on convergence to stationarity for nonreversible Markov chains, with an application to the exclusion process. *Ann. Appl. Probab.*, 1(1):62–87, 1991.

17 Bhaskar Ghosh, Frank Thomson Leighton, Bruce M. Maggs, S. Muthukrishnan, C. Greg Plaxton, Rajmohan Rajaraman, Andréa W. Richa, Robert Endre Tarjan, and David Zuckerman. Tight Analyses of Two Local Load Balancing Algorithms. *SIAM J. Comput.*, 29(1):29–64, 1999. doi:10.1137/S0097539795292208.

18 George Giakkoupis, Thomas Sauerwald, and Alexandre Stauffer. Randomized Rumor Spreading in Dynamic Graphs. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, pages 495–507, 2014.

19 Sharad Goel, Ravi Montenegro, and Prasad Tetali. Mixing time bounds via the spectral profile. *Electron. J. Probab.*, 11:1–26, 2006.

20 Jonathan Hermon and Perla Sousi. Random walk on dynamical percolation. *arXiv preprint*, 2019. arXiv:1902.02770.

21 Varun Kanade, Frederik Mallmann-Trenn, and Thomas Sauerwald. On coalescence time in graphs: When is coalescing as fast as meeting? In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 956–965, 2019.

22 Fabian Kuhn, Nancy A. Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 513–522, 2010.

23 Fabian Kuhn and Rotem Oshman. Dynamic networks: models and algorithms. *SIGACT News*, 42(1):82–96, 2011.

24 Henry Lam, Zhenming Liu, Michael Mitzenmacher, Xiaorui Sun, and Yajun Wang. Information dissemination via random walks in d-dimensional space. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1612–1622, 2012.

25 Ioannis Lamprou, Russell Martin, and Paul Spirakis. Cover time in edge-uniform stochastically-evolving graphs. *Algorithms (Basel)*, 11(10):Paper No. 149, 15, 2018.

26 David A. Levin and Yuval Peres. *Markov chains and mixing times*. American Mathematical Society, Providence, RI, 2017.

27 László Lovász and Ravi Kannan. Faster mixing via average conductance. In *31st Annual ACM Symposium on Theory of Computing*, pages 282–287. ACM, 1999.

28 Othon Michail and Paul G. Spirakis. Elements of the theory of dynamic networks. *Commun. ACM*, 61(2):72, 2018. doi:10.1145/3156693.

29 Yuval Peres, Alistair Sinclair, Perla Sousi, and Alexandre Stauffer. Mobile Geometric Graphs: Detection, Coverage and Percolation. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 412–428, 2011.

30 Yuval Peres, Alexandre Stauffer, and Jeffrey E. Steif. Random walks on dynamical percolation: mixing times, mean squared displacement and hitting times. *Probab. Theory Related Fields*, 162(3-4):487–530, 2015.

31 Alberto Pettarin, Andrea Pietracaprina, Geppino Pucci, and Eli Upfal. Tight bounds on information dissemination in sparse mobile networks. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011, San Jose, CA, USA, June 6-8, 2011*, pages 355–362, 2011.

32 Laurent Saloff-Coste and Jessica Zúñiga. Merging for time inhomogeneous finite Markov chains. I. Singular values and stability. *Electron. J. Probab.*, 14:1456–1494, 2009.

33 Laurent Saloff-Coste and Jessica Zúñiga. Merging for inhomogeneous finite Markov chains, Part II: Nash and log-Sobolev inequalities. *Ann. Probab.*, 39(3):1161–1203, 2011.

34 Atish Das Sarma, Anisur Rahaman Molla, and Gopal Pandurangan. Distributed computation in dynamic networks via random walks. *Theor. Comput. Sci.*, 581:45–66, 2015.

35 Alistair Sinclair and Mark Jerrum. Approximate counting, uniform generation and rapidly mixing Markov chains. *Information and Computation*, 82(1):93–133, 1989.

36 Perla Sousi and Sam Thomas. Cutoff for Random Walk on Dynamical Erdos-Renyi graph. *arXiv preprint*, 2018. `arXiv:1807.04719`.

Querying a Matrix Through Matrix-Vector Products

Xiaoming Sun
CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
University of Chinese Academy of Sciences, Beijing, China
sunxiaoming@ict.ac.cn

David P. Woodruff
Carnegie Mellon University, Pittsburgh, PA, US
dwoodruf@andrew.cmu.edu

Guang Yang
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
Conflux, Beijing, China
guang.research@gmail.com

Jialin Zhang
CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
University of Chinese Academy of Sciences, Beijing, China
zhangjialin@ict.ac.cn

Abstract

We consider algorithms with access to an unknown matrix $\mathbf{M} \in \mathbb{F}^{n \times d}$ via *matrix-vector products*, namely, the algorithm chooses vectors $\mathbf{v}^1, \ldots, \mathbf{v}^q$, and observes $\mathbf{M}\mathbf{v}^1, \ldots, \mathbf{M}\mathbf{v}^q$. Here the $\mathbf{v}^i$ can be randomized as well as chosen adaptively as a function of $\mathbf{M}\mathbf{v}^1, \ldots, \mathbf{M}\mathbf{v}^{i-1}$. Motivated by applications of sketching in distributed computation, linear algebra, and streaming models, as well as connections to areas such as communication complexity and property testing, we initiate the study of the number q of queries needed to solve various fundamental problems. We study problems in three broad categories, including linear algebra, statistics problems, and graph problems. For example, we consider the number of queries required to approximate the rank, trace, maximum eigenvalue, and norms of a matrix $\mathbf{M}$; to compute the AND/OR/Parity of each column or row of $\mathbf{M}$, to decide whether there are identical columns or rows in $\mathbf{M}$ or whether $\mathbf{M}$ is symmetric, diagonal, or unitary; or to compute whether a graph defined by $\mathbf{M}$ is connected or triangle-free. We also show separations for algorithms that are allowed to obtain matrix-vector products only by querying vectors on the right, versus algorithms that can query vectors on both the left and the right. We also show separations depending on the underlying field the matrix-vector product occurs in. For graph problems, we show separations depending on the form of the matrix (bipartite adjacency versus signed edge-vertex incidence matrix) to represent the graph.

Surprisingly, this fundamental model does not appear to have been studied on its own, and we believe a thorough investigation of problems in this model would be beneficial to a number of different application areas.

2012 ACM Subject Classification Theory of computation → Complexity classes; Theory of computation → Lower bounds and information complexity

Keywords and phrases Communication complexity, linear algebra, sketching

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.94

Category Track A: Algorithms, Complexity and Games

Related Version `https://arxiv.org/abs/1906.05736`

Acknowledgements We want to thank Roman Vershynin and Yan Shuo Tan for the helpful comments.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 94; pp. 94:1–94:16

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

Funding This work was supported in part by the National Natural Science Foundation of China Grants No. 61433014, 61761136014, 61872334, 61502449, 61602440, the 973 Program of China Grant No. 2016YFB1000201, and K.C.Wong Education Foundation. David Woodruff would like to thank the Chinese Academy of Sciences, as well as the Simons Institute for the Theory of Computing where part of this work was done. He also acknowledges partial support by the National Science Foundation under Grant No. CCF-1815840.

1 Introduction

Suppose there is an unknown matrix $\mathbf{M} \in \mathbb{F}^{n\times d}$ that you can only access via a sequence of *matrix-vector products* $\mathbf{M}\cdot\mathbf{v}^1,\ldots,\mathbf{M}\cdot\mathbf{v}^q$, where we call the vectors $\mathbf{v}^1,\ldots,\mathbf{v}^q$ the *query vectors*, which can be chosen in a randomized, possibly adaptive way. By adaptive, we mean that $\mathbf{v}^i$ can depend on $\mathbf{v}^1,\ldots,\mathbf{v}^{i-1}$ as well as $\mathbf{M}\mathbf{v}^1,\ldots,\mathbf{M}\mathbf{v}^{i-1}$. Here $\mathbb{F}$ is a field, and we study different fields for different applications. Suppose our goal is to determine if $\mathbf{M}$ satisfies a specific property $\mathcal{P}$, such as having approximately full rank, or for example whether $\mathbf{M}$ has two identical columns. A natural question is the following:

Question 1: How many queries q are necessary to determine if $\mathbf{M}$ has property $\mathcal{P}$?

A number of well-studied problems are special cases of this question, i.e., compressed sensing or sparse recovery, for which $\mathbf{M} \in \mathbb{R}^{1\times d}$ is an approximately k-sparse vector, and one would like a number q of queries close to k. It is known that if the query sequence is non-adaptive, meaning $\mathbf{v}^1,\ldots,\mathbf{v}^q$ are chosen before making any queries, then $q = \Theta(k\log(n/k))$ is necessary and sufficient [10, 5] to recover an approximately k-sparse vector[1]. However, if the queries can be adaptive, then $q = O(k\log\log n)$ queries suffice [12], while there is a lower bound of $\Omega(k+\log\log n)$ [24] (see also recent work [23, 13]).

The above problem is representative of an emerging field called *linear sketching* which is the underlying technique behind a number of algorithmic advances the past two decades. In this model one queries $\mathbf{M}\cdot\mathbf{v}^1,\ldots,\mathbf{M}\cdot\mathbf{v}^r$ for non-adaptive queries $\mathbf{v}^1,\ldots,\mathbf{v}^r$. For brevity we write this as $\mathbf{M}\cdot\mathbf{V}$, where $\mathbf{V}\in\mathbb{F}^{d\times r}$ has i-th column equal to $\mathbf{v}^i$. Linear sketching has played a central role in the development of streaming algorithms [2]. Perhaps more surprisingly, linear sketches are also known to achieve the minimal space necessary *of any, possibly non-linear, algorithm* for processing dynamic data streams under certain general conditions [20, 1, 15], which is an essential result for proving a number of lower bounds for approximating matchings in a stream [18, 4]. Linear sketching has also led to the fastest known algorithms for problems in numerical linear algebra, such as least squares regression and low rank approximation; for a survey see [29]. Note that given $\mathbf{M}\cdot\mathbf{V}$ and $\mathbf{M}'\cdot\mathbf{V}$, by linearity one can compute $(\mathbf{M}+\mathbf{M}')\cdot\mathbf{V} = \mathbf{M}\cdot\mathbf{V}+\mathbf{M}'\cdot\mathbf{V}$. This basic versatility property allows for fast updates in a data stream and mergeability in environments such as MapReduce and other distributed models of computation.

Given the applications above, we consider Question 1 an important question to understand for many different properties $\mathcal{P}$ of interest, which we describe in more detail below. A central goal of this work is to answer Question 1 for such properties and to propose this be a natural model of study in its own right.

One notable difference with our model and a number of appications of linear sketching is that we will allow for adaptive query sequences. In fact, our upper bounds will be non-adaptive, and our nearly matching lower bounds for each problem we consider will hold even

[1] Here the goal is to output a vector $\mathbf{M}'$ for which $\|\mathbf{M}-\mathbf{M}'\|_2 \le (1+\epsilon)\|\mathbf{M}-\mathbf{M}_k\|_2$, where $\mathbf{M}_k$ is the best k-sparse approximation to $\mathbf{M}$, and ϵ is a constant.

for adaptive query sequences. Our model is also related to property testing, where one tries to infer properties of a large unknown object by (possibly adaptively) sampling a sublinear number of locations of that object. We argue that linear queries are a natural extension of sampling locations of an object, and that this is a natural "sampling model" not only because of the desired properties of the distributed, linear algebra, and streaming applications above, but sometimes also for physical constraints, e.g., in compressed sensing, where optical devices naturally capture linear measurements.

From a theoretical standpoint, any property testing algorithm, i.e., one that samples q entries of $\mathbf{M}$, can be implemented in our model with q linear queries. However, our model gives the algorithm much more flexibility. From a lower bound perspective, as in the case of property testing [8], some of our lower bounds will be derived from communication complexity. However, not all of our bounds can be proved this way. For example, one notable result we show is an optimal lower bound on the number of queries needed to approximate the rank of $\mathbf{M} \in \mathbb{R}^{n\times n}$ up to a factor t by randomized, possibly adaptive algorithms; we show that $\frac{n}{t} + 1$ queries are necessary and sufficient. A natural alternative way to prove this would be to give part of the matrix to Alice, part of to Bob, and have the players exchange the $\mathbf{M}^L\mathbf{v}^i$ and $\mathbf{M}^R\mathbf{v}^i$, where $\mathbf{M} = \mathbf{M}^L + \mathbf{M}^R$ and $\mathbf{M}^L$ is Alice's part and $\mathbf{M}^R$ is Bob's part. Then, if the 2-player randomized communication complexity of approximating the rank of $\mathbf{M}$ up to a factor of t were known to be $\Omega(n^2/t)$, we would obtain a nearly-matching query lower bound of $\Omega(n/(t(b + \log n)))$, where b is the number of bits needed to specify the entries of $\mathbf{M}$ and the queries. However, it is unknown what the 2-player communication complexity of approximating the rank of $\mathbf{M}$ up to a factor t is over $\mathbb{R}$! We are not aware of any lower bound better than $\Omega(1)$ for constant t for this problem for adaptive queries. We note that for non-adaptive queries, there is an $\Omega(n^2)$ sketching lower bound over the reals given in [19], and an $\Omega(n^2/\log p)$ lower bound for finite fields (of size p) in [3]. There is also a property testing lower bound in [6], though such a lower bound makes additional assumptions on the input. Thus, our model gives a new lens to study this problem from, from which we are able to derive strong lower bounds for adaptive queries. Our techniques could be helpful for proving lower bounds in existing models, such as two-party communication complexity.

Our model is also related to linear decision tree complexity, see, e.g., [7, 14], though such lower bounds typically involve just seeing a threshold applied to $\mathbf{M}\mathbf{v}^i$, and typically $\mathbf{M}$ is a vector. In our case, we observe the entire output vector $\mathbf{M}\mathbf{v}^i$.

An interesting twist in our model is that in our formulation above, we only allowed to query $\mathbf{M}$ via matrix-vector products *on the right*, i.e., of the form $\mathbf{M} \cdot \mathbf{v}^i$. One could ask if there are natural properties $\mathcal{P}$ of $\mathbf{M}$ for which the number q_L of queries one would need to make if querying $\mathbf{M}$ via queries of the form $(\mathbf{u}^1)^T\mathbf{M}, (\mathbf{u}^2)^T\mathbf{M}, \ldots, (\mathbf{u}^{q_L})^T\mathbf{M}$ can be significantly smaller than the number q_R of queries one would need to make if querying $\mathbf{M}$ via queries of the form $\mathbf{M}\mathbf{u}^1, \mathbf{M}\mathbf{u}^2, \ldots, \mathbf{M}\mathbf{u}^{q_R}$:

Question 2: Are there natural problems for which $q_L \ll q_R$?

We show that this is in fact the case, namely, if we can only multiply on the right, then it takes $\Omega(n/\log n)$ queries to determine if there is a *column* of a matrix $\mathbf{M} \in \{0,1\}^{n\times n}$ which is all 1s. However, if we can multiply on the left, then the single query $(1, 1, \ldots, 1)$ can determine this.

We study a few problems around Question 2, which is motivated from several perspectives. First, matrices might be stored on computers in a specific encoding, e.g., a sparse row format, from which it may be much easier to multiply on the right than on the left. Also, in compressed sensing, it may be natural for physical reasons to obtain linear combinations of columns rather than rows.

Another important question is how the query complexity depends on the *underlying field* for which matrix-vector products are performed. Might it be that for a natural problem the query complexity if the matrix-vector products are performed modulo 2 is much higher than if the matrix-vector products are performed over the reals?

Question 3: Is there a natural problem for which the query complexity in our model over $\mathbb{F}[2]$ is much larger than that over the reals?

Yet another important application of this model is to querying graphs. A natural question is which representation to use for the graph. For example, a natural representation of a graph on n vertices is through its adjacency matrix $\mathbf{A} \in \{0,1\}^{n \times n}$, where $\mathbf{A}_{i,j} = 1$ if and only if $\{i,j\}$ occurs as an edge. A natural representation for a bipartite graph with n vertices in each part could be an $n \times n$ matrix $\mathbf{A}$ where $\mathbf{A}_{i,j} = 1$ iff there is an edge from the i-th left vertex to the j-th right vertex. Yet another representation could be the $\binom{n}{2} \times n$ edge-vertex incidence matrix, where the $\{i,j\}$-th row is either 0, or has exactly two ones, one in location i and one in location j. One often considers a signed edge-vertex incidence matrix, where one first arbitrarily fixes an ordering on the vertices and then the $\{i,j\}$-th entry has a 1 in the i-th position and a -1 in the j-th position if $i > j$, otherwise positions i and j are swapped. Yet another possible representation of a graph is through its Laplacian.

Question 4: Do some natural representations of graphs admit much more efficient query algorithms for certain problems than other natural representations?

We note that in the data stream model, where one sees a long sequence of insertions and deletions to the edges of a graph, each of the matrix representations above can be simulated and so they lead to the same complexity. We will show, perhaps surprisingly, that in this model there can be an exponential difference in the query complexity for two different natural representations of a graph for the same problem.

We next get into the details of our results. We would like to stress that even basic problems in this model are not immediately obvious how to tackle. As a puzzle for the reader, what is the query complexity of determining if a matrix $\mathbf{M} \in \mathbb{F}^{n \times n}$ is symmetric if one can only query vectors on the right? We will answer this later in the paper.

1.1 Formal Model and Our Results

We now describe our model and results formally in terms of an oracle. The oracle has a matrix $\mathbf{M} \in \mathbb{F}^{m \times n}$, for some underlying field $\mathbb{F}$ that we specify in each application. We can only query this matrix via matrix-vector products, i.e., we pick an arbitrary vector $\mathbf{x}$ and send it to the oracle, and the oracle will respond with a vector $\mathbf{y} = \mathbf{M} \cdot \mathbf{x}$. We focus our attention when the queries only occur on the right. Our goal is to approximate or test a number of properties of $\mathbf{M}$ with a minimal number of queries, i.e., to answer Question 1 for a large number of different application areas.

We study a number of problems as summarized in the table. Due to the space limitation, we leave some proofs in the full version. We assume $\mathbf{M}$ is an $m \times n$ matrix and $\varepsilon > 0$ is a parameter of the problem. The bounds hold for constant probability algorithms. In some problems, such as testing whether the matrix is a diagonal matrix, we always assume $m = n$, and in the graph testing problems we explicitly describe how the graph is represented using $\mathbf{M}$. Interestingly, we are able to prove very strong lower bounds for approximating the rank, which as described above, are unknown to hold for randomized communication complexity.

Motivated be streaming and statistics questions, we next study the query complexity of approximating the norm of each row of $\mathbf{M}$. We also study the computation of the majority or parity of each column or row of $\mathbf{M}$, the AND/OR of each column or row of $\mathbf{M}$, or equivalently, whether $\mathbf{M}$ has an all ones column or row, whether $\mathbf{M}$ has two identical columns or rows, and whether $\mathbf{M}$ contains an unusually large-normed row, i.e., a "heavy hitter". Here we show there are natural problems, such as computing the parity of all columns, which can be solved with 1 query if sketching on the left, but require $\Omega(n)$ queries if sketching on the right, thus answering Question 2. We also answer Question 3, observing for the natural problem of testing if a row is all ones, a single deterministic query suffices over the reals but over $\mathbb{F}[2]$ this deterministically requires $\Omega(n)$ queries.

For graph problems, we first argue if the graph is presented as an $n \times n$ bipartite adjacency matrix $\mathbf{M}$, then it requires $\Omega(n/\log n)$ possibly adaptive queries to determine if the graph is connected. In contrast, if the graph is presented as an $n \times \binom{n}{2}$ signed vertex-edge incidence matrix, then $\mathsf{polylog}(n)$ non-adaptive queries suffices. This answers Question 4, showing that the type of representation of the graph is critical in this model. Motivated by a large body of recent work on triangle counting (see, e.g., [11] and the references therein), we also give strong negative results for this problem in our model, which as with all of our lower bounds unless explicitly stated otherwise, hold even for algorithms which perform adaptive queries.

Table 1 Our Results.

Problem	Query Complexity
Linear Algebra Problems	
Approximate Rank (for any $p' > p$ distinguishing Rank$\leq p$ from Rank p')	$p+1$ (Section 3.1)
Trace Estimation	$\Omega(n/\log n)$ (Section 3.2)
Symmetric Matrix / Diagonal Matrix	$O(1)$ (Section 3.3 and full version)
Unitary Matrix	1 (full version)
Approximate Maximum Eigenvalue	$\Theta(\varepsilon^{-0.5}\log n)$ for adaptive queries, $\Theta(n)$ for non-adaptive queries (full version)
Streaming and Statistics Problems	
All Ones Column	$\Theta(n)$ over $\mathbb{F}[2]$, $\Omega(n/\log n)$ over $\mathbb{R}$ (Section 4.1)
Two Identical Columns Two Identical Rows	$\Theta(n)$ $O(\log m)$ (Section 4.2)
Approximate Row Norms / Heavy Hitters	$O\left(\varepsilon^{-2}\log m\right)$ (full version)
Majority of Columns Majority of Rows	$\Omega(n/\log n)$ over $\mathbb{R}$ $O(1)$ over $\mathbb{R}$ (full version)
Parity of Columns Parity of Rows	$\Theta(n)$ $O(1)$ (full version)
Graph Problems	
Connectivity given Bipartite Adjacency Matrix Connectivity given Signed Edge-Vertex Matrix	$\Omega(n/\log n)$ (Section 5.1) $O(\mathsf{polylog}(n))$ ([16], noted in Section 5.1)
Triangle Detection	$\Omega(n/\log n)$ (Section 5.2)

2 Preliminaries

We use capital bold letters, e.g., $\mathbf{A}, \mathbf{B}, \mathbf{M}$, to denote matrices, and use lowercase bold letters, e.g., $\mathbf{x}, \mathbf{y}$, to denote column vectors. Sometimes we write a matrix as a list of column vectors in square brackets, e.g., $\mathbf{M} = [\mathbf{m}_1, \ldots, \mathbf{m}_n]$. We use calligraphic letters, e.g., $\mathcal{D}$, to denote probability distributions, and use $\mathbf{M} \leftarrow \mathcal{D}$ to denote that $\mathbf{M}$ is sampled from distribution $\mathcal{D}$. In particular, we use $\mathcal{G}$ to denote a Gaussian distribution and $\mathbf{G}$ for a matrix whose entries are sampled from an independently and identically distributed (denoted as i.i.d. in the following) Gaussian distribution.

We call a matrix $\mathbf{M}$ i.i.d. Gaussian if each element is i.i.d. Gaussian. It is easy to check that if matrix $\mathbf{G}$ is a $p \times n$ i.i.d. Gaussian matrix, and $\mathbf{R}$ is an $n \times n$ rotation matrix, then $\mathbf{G} \times \mathbf{R}$ is still i.i.d. Gaussian, and has the same probability distribution of $\mathbf{G}$.

The *total variation distance*, sometimes called the statistical distance, between two probability measures P and Q is defined as $\mathsf{D}_{\mathsf{TV}}(P, Q) \stackrel{\text{def}}{=} \sup_A |P(A) - Q(A)|$.

Let $\mathbf{X}$ be an $n \times m$ matrix with each row i.i.d. drawn from an m-variate normal distribution $N(0, \Sigma)$. Then the distribution of the $m \times m$ random matrix $\mathbf{A} = \mathbf{X}^T\mathbf{X}$ is called the *Wishart distribution* with n degrees of freedom and covariance matrix Σ, denoted by $W_m(n, \Sigma)$. The distribution of eigenvalues of $\mathbf{A}$ is characterized in the following lemma.

▶ **Lemma 1** (Corollary 3.2.19 in [17]). *If $\mathbf{A}$ is $W_m(n, \lambda I_m)$, with $n > m - 1$, the joint density function of the eigenvalues $\mathbf{\Lambda} = (\lambda_1, \ldots, \lambda_m)$ of $\mathbf{A}$ (in descending order) is*

$$f(\mathbf{\Lambda}) = \frac{\pi^{m^2/2}}{(2\lambda)^{mn/2}\Gamma_m(m/2)\Gamma_n(n/2)} \exp\left(-\frac{1}{2\lambda}\sum_{i=1}^{m}\lambda_i\right)\prod_{i=1}^{m}\lambda_i^{(n-m-1)/2}\prod_{1\le i<j\le m}(\lambda_i - \lambda_j)$$

In particular, for $\lambda = 1$ and $n = m$, $\exists$ a constant Z_m independent from $\lambda_1, \ldots, \lambda_m$, such that

$$f(\mathbf{\Lambda}) = \frac{1}{Z_m}\exp\left(-\frac{1}{2}\sum_{i=1}^{m}\lambda_i\right)\prod_{i=1}^{m}\lambda_i^{-1/2}\prod_{1\le i<j\le m}(\lambda_i - \lambda_j)$$

3 Linear Algebra Problems

In this section we present our lower bound for rank approximation in Section 3.1, trace estimation in Section 3.2, and testing whether a matrix is symmetric. The results for testing diagonal or unitary matrices, and approximating the maximum eigenvalue is contained in the full version of our paper.

3.1 Lower Bound for Rank Approximation

In this section, we discuss how to approximate the rank of a given matrix $\mathbf{M}$ over the reals when the queries consist of right multiplication by vectors. A naïve algorithm to learn the rank is to pick random Gaussian query vectors non-adaptively. In order to approximate the rank, that is, to distinguish whether $\mathsf{rank}(\mathbf{M}) \le p$ or $\mathsf{rank}(\mathbf{M}) \ge p + 1$, this algorithm needs at least $p + 1$ queries, and it is not hard to see that the algorithm succeeds with probability 1. Indeed, if $\mathbf{H} \in \mathbb{R}^{n\times(p+1)}$ is the random Gaussian query matrix, and $\mathbf{M}$ the unknown $n \times n$ matrix, then writing $\mathbf{M}$ in its thin singular value decomposition as $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ have orthonormal columns, and $\mathbf{\Sigma}$ has positive diagonal entries, we have that $\mathrm{rank}(\mathbf{M} \cdot \mathbf{H}) = \mathrm{rank}(\mathbf{V}^T\mathbf{H})$, which by rotational invariance of the Gaussian distribution is the the same as the rank of a random Gaussian matrix, which will be the minimum of $p + 1$ and the rank of $\mathbf{M}$ with probability 1.

In the following, we will show that we cannot expect anything better. We will first show for non-adaptive queries, at least $p+1$ queries are necessary to learn the approximate rank. Then we generalize our results to adaptive queries. Our results hold for randomized algorithms by applying Yao's minimax principle.

3.1.1 Non-Adaptive Query Protocols

▶ **Theorem 2.** *Let constant $\varepsilon > 0$ be the error tolerance and let $\mathbf{M}$ be an $n \times n$ oracle matrix and suppose to start that we make non-adaptive queries. For integer $p < p' \leq n$, at least $p+1$ queries are necessary to distinguish $\mathsf{rank}(\mathbf{M}) \leq p$ from $\mathsf{rank}(\mathbf{M}) \geq p'$ with advantage $\geq \varepsilon$.*

Proof. Given any algorithm distinguishing $\mathsf{rank}(\mathbf{M}) \leq p$ from $\mathsf{rank}(\mathbf{M}) \geq p'$ for some $p' < n$, we can determine whether a $p' \times p'$ matrix $\mathbf{M}'$ has full rank p' or $\mathsf{rank}(\mathbf{M}') \leq p$, by padding $\mathbf{M}'$ to an $n \times n$ matrix $\mathbf{M}$. Therefore in what follows it suffices to prove the lower bound for two $n \times n$ matrices $\mathbf{M}_1$ and $\mathbf{M}_2$ where $\mathsf{rank}(\mathbf{M}_1) \leq p$ and $\mathsf{rank}(\mathbf{M}_2) = n$:

1. $\mathbf{M}_1 = \mathbf{U} \times \mathbf{G}^T$;
2. $\mathbf{M}_2 = \mathbf{U} \times \mathbf{G}^T + \frac{1}{Z(n)} \cdot \mathbf{U}^\perp \times \mathbf{H}^T$.

Here $\mathbf{U}$ has p columns and $\mathbf{U}^\perp$ has $(n-p)$ columns such that $\left[\mathbf{U}, \mathbf{U}^\perp\right]$ forms an $n \times n$ random orthonormal basis, $\mathbf{G}^T$ and $\mathbf{H}^T$ are $p \times n$ and $(n-p) \times n$ matrices whose entries sampled i.i.d. from the standard Gaussian distribution, and $Z(n)$ is a function in n which will be specified later. It immediately follows that $\mathsf{rank}(\mathbf{M}_1) \leq p$ and $\mathsf{rank}(\mathbf{M}_2) = n$ with overwhelmingly high probability. Then we assume $\mathsf{rank}(\mathbf{M}_2) = n$ and discuss the query lower bound for distinguishing $\mathbf{M}_1$ from $\mathbf{M}_2$.

Given $\mathbf{M} \in \{\mathbf{M}_1, \mathbf{M}_2\}$, without loss of generality we denote the q non-adaptive queries with an $n \times q$ orthonormal[2] matrix $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_q]$, where $q \leq p$ and each $n \times 1$ column vector $\mathbf{v}_i$ is a query to the oracle of matrix $\mathbf{M}$ which gets response $\mathbf{M} \cdot \mathbf{v}_i$, for $i \in [q]$. Then, it suffices to show that the following two distributions are hard to distinguish:

1. $\mathbf{M}_1 \times \mathbf{V} \equiv \mathbf{U}\mathbf{W}$, where $\mathbf{W} = \mathbf{G}^T\mathbf{V}$;
2. $\mathbf{M}_2 \times \mathbf{V} \equiv \mathbf{U}\mathbf{W} + \frac{1}{Z(n)} \cdot \mathbf{U}^\perp\mathbf{W}'$, where $\mathbf{W}' = \mathbf{H}^T\mathbf{V}$.

Note that $\left[\mathbf{U}, \mathbf{U}^\perp\right]$ is orthonormal, and hence $\mathbf{U}^T\mathbf{U} = \mathbf{I}_p$, $\left(\mathbf{U}^\perp\right)^T\mathbf{U}^\perp = \mathbf{I}_{n-p}$, $\mathbf{U}^T\mathbf{U}^\perp = \mathbf{0}_{p\times(n-p)}$. We introduce Lemma 3 to eliminate $\mathbf{U}, \mathbf{U}^\perp$ in the representation of $\mathbf{M} \times \mathbf{V}$.

▶ **Lemma 3.** *For $\mathbf{M}_1, \mathbf{M}_2$ and $\mathbf{V}$ defined as above, there is*

$$\mathsf{D}_{\mathsf{TV}}(\mathbf{M}_1\mathbf{V}, \mathbf{M}_2\mathbf{V}) = \mathsf{D}_{\mathsf{TV}}\left((\mathbf{M}_1\mathbf{V})^T\mathbf{M}_1\mathbf{V}, (\mathbf{M}_2\mathbf{V})^T\mathbf{M}_2\mathbf{V}\right)$$

Proof. The direction $\mathsf{D}_{\mathsf{TV}}(\mathbf{M}_1\mathbf{V}, \mathbf{M}_2\mathbf{V}) \geq \mathsf{D}_{\mathsf{TV}}\left((\mathbf{M}_1\mathbf{V})^T\mathbf{M}_1\mathbf{V}, (\mathbf{M}_2\mathbf{V})^T\mathbf{M}_2\mathbf{V}\right)$ is trivial following data processing inequality (i.e. for every $\mathbf{X}, \mathbf{Y}$ and function f, $\mathsf{D}_{\mathsf{TV}}(\mathbf{X}, \mathbf{Y}) \geq \mathsf{D}_{\mathsf{TV}}(f(\mathbf{X}), f(\mathbf{Y}))$). In what follows we only prove the other direction.

First we notice that for every fixed $n \times n$ orthonormal matrix $\mathbf{R}$ and for a random matrix $\mathbf{M}$ sampled as $\mathbf{M}_1$ or $\mathbf{M}_2$, the product $\mathbf{N} \stackrel{\text{def}}{=} \mathbf{R}\mathbf{M}$ follows exactly the same distribution of $\mathbf{M}$. Thus $\mathbf{N}\mathbf{V}$ and $\mathbf{M}\mathbf{V}$ are identically distributed.

Then, from a random sample $\mathbf{V}^T\mathbf{M}^T\mathbf{M}\mathbf{V}$ we can find $\mathbf{M}'$ such that $\mathbf{V}^T\mathbf{M}^T\mathbf{M}\mathbf{V} = (\mathbf{M}')^T\mathbf{M}'$ and $\mathbf{M}' = \mathbf{S}\mathbf{M}\mathbf{V}$ for some orthonormal matrix $\mathbf{S}$ and orthonormal query matrix $\mathbf{V}$. Although $\mathbf{M}'$ is not necessarily the same as $\mathbf{M}\mathbf{V}$ because of $\mathbf{S}$, we have $\mathbf{R}\mathbf{M}' \sim$

[2] Any non-orthonormal queries can be made orthonormal using a change of basis in post-processing.

$\mathbf{NV} \sim \mathbf{MV}$ for a uniformly random orthonormal matrix $\mathbf{R}$. Thus we transform a random sample from $\mathbf{V}^T\mathbf{M}^T\mathbf{MV}$ into a sample from $\mathbf{MV}$ via $\mathbf{RM'}$, and hence $\mathsf{D}_{\mathsf{TV}}(\mathbf{M}_1\mathbf{V}, \mathbf{M}_2\mathbf{V}) \leq \mathsf{D}_{\mathsf{TV}}\left((\mathbf{M}_1\mathbf{V})^T \mathbf{M}_1\mathbf{V}, (\mathbf{M}_2\mathbf{V})^T \mathbf{M}_2\mathbf{V}\right)$. ◀

Using Lemma 3, it suffices to prove an upper bound for $\mathsf{D}_{\mathsf{TV}}(\mathbf{\Lambda}, \mathbf{\Lambda'})$ as follows:

$$\mathsf{D}_{\mathsf{TV}}\left(\mathbf{UW}, \mathbf{UW} + \frac{\mathbf{U}^\perp\mathbf{W'}}{Z(n)}\right) = \mathsf{D}_{\mathsf{TV}}\left((\mathbf{UW})^T(\mathbf{UW}), \left(\mathbf{UW} + \frac{\mathbf{U}^\perp\mathbf{W'}}{Z(n)}\right)^T(\mathbf{UW} + \frac{\mathbf{U}^\perp\mathbf{W'}}{Z(n)})\right)$$
$$= \mathsf{D}_{\mathsf{TV}}\left(\mathbf{W}^T\mathbf{W}, \mathbf{W}^T\mathbf{W} + \frac{(\mathbf{W'})^T\mathbf{W'}}{Z^2(n)}\right) \leq \mathsf{D}_{\mathsf{TV}}\left(\mathbf{\Lambda}, \mathbf{\Lambda'}\right)$$

where $\mathbf{\Lambda} = \mathsf{diag}(\lambda_1, \ldots, \lambda_q), \mathbf{\Lambda'} = \mathsf{diag}(\lambda'_1, \ldots, \lambda'_q)$ are diagonal matrices such that $\mathbf{W}^T\mathbf{W} = \mathbf{A}^T\mathbf{\Lambda}\mathbf{A}$ and $\mathbf{W}^T\mathbf{W} + \frac{(\mathbf{W'})^T\mathbf{W'}}{Z^2(n)} = \mathbf{B}^T\mathbf{\Lambda'}\mathbf{B}$ for orthonormal matrices $\mathbf{A}$ and $\mathbf{B}$. The inequality follows because any algorithm separating $\mathbf{W}^T\mathbf{W}$ from $\mathbf{W}^T\mathbf{W} + \frac{(\mathbf{W'})^T\mathbf{W'}}{Z^2(n)}$ implies a separation of $\mathbf{\Lambda}$ from $\mathbf{\Lambda'}$ with the same advantage, by multiplying by random orthonormal matrices.

By Weyl's inequality [28, 31], for every $i \in [q]$, $\lambda'_i \in [\lambda_i - \|\mathbf{\Lambda'} - \mathbf{\Lambda}\|_2, \lambda_i + \|\mathbf{\Lambda'} - \mathbf{\Lambda}\|_2]$, and hence $\lambda'_i \in \left[\lambda_i - O\left(\frac{\|\mathbf{W'}\|_2}{Z^2(n)}\right), \lambda_i + O\left(\frac{\|\mathbf{W'}\|_2}{Z^2(n)}\right)\right]$. Notice that $\mathbf{W'}$ is an $(n-p) \times q$ i.i.d. Gaussian matrix, and hence $\|\mathbf{W'}\|_2^2$ is a chi-squared variable with $(n-p)q$ degrees of freedom, which is bounded by $O((n-p)q)$ with high probability (c.f. Example 2.12 in [27]). Recalling that $q \leq p$, in what follows we condition on the event $\lambda'_i \in \left[\lambda_i - O\left(\frac{np}{Z^2(n)}\right), \lambda_i + O\left(\frac{np}{Z^2(n)}\right)\right]$.

We then show the gaps between eigenvalues λ_i are sufficiently large. Note that since $\mathbf{G}^T$ is i.i.d Gaussian and $\mathbf{V}$ is an orthonormal matrix, each row in $\mathbf{W} = \mathbf{G}^T\mathbf{V}$ is independently drawn from an q-variate normal distribution, thus the probability distribution of $\mathbf{W}^T\mathbf{W}$ is a Wishart distribution $W_q(p, I_q)$. Let $q = p$ and $\lambda_1, \ldots, \lambda_p$ be sorted in descending order. Then by Lemma 1 the density function of $\mathbf{\Lambda}$ is:

$$f(\mathbf{\Lambda}) = \frac{1}{Z_p}\exp\left(-\frac{1}{2}\sum_{i=1}^{p}\lambda_i\right)\prod_{i=1}^{p}\lambda_i^{-1/2}\prod_{1\leq i<j\leq p}(\lambda_i - \lambda_j) \tag{1}$$

Let $\mathcal{E}$ denote the event that $\lambda_p \geq \frac{0.01}{\sqrt{n}}$ and $\forall 1 \leq i < j \leq p, \lambda_i - \lambda_j \geq \gamma = 2^{-\Theta(p^2 \log p)}$.

▶ **Lemma 4.** *For $\mathbf{W}^T\mathbf{W}$ defined as above and sufficiently small $\gamma = 2^{-\Theta(p^2 \log n)}$, $\Pr[\mathcal{E}] > 0.9$.*

Proof. By equation (2) in [25] we know that $\Pr[\sqrt{n}\lambda_p \geq y] = \exp\left(-(y^2/2 + y)\right)$. Thus for $y = 0.01$ and $\mathcal{E}_0 \stackrel{\text{def}}{=} \{\lambda_p \geq 0.01/\sqrt{n}\}$ we get:

$$\Pr[\mathcal{E}_0] = \Pr\left[\lambda_p \geq \frac{0.01}{\sqrt{n}}\right] = \exp(-0.01005) > 0.99000033$$

Also, we note that for every i, $\Pr[|\lambda_i| \leq 100n] \geq 1 - 2\exp(-32n)$, by setting $t = 8\sqrt{n}$ in Corollary 5.35 of [26]. In what follows we conditioned on $\mathcal{E}'_0$ that $|\lambda_i| \leq 100n$ for every $i \in [p]$.

Then we consider the joint distribution μ of $\lambda_1, \ldots, \lambda_p$ in $\mathbf{\Lambda}$. Let the $\mathcal{E}_i \stackrel{\text{def}}{=} \{\lambda_i - \lambda_{i+1} < \gamma\}$ be the event that λ_i and λ_{i+1} has a gap smaller than γ. Thus $\mathcal{E} = \mathcal{E}_0 \wedge \left(\wedge_{i=1}^{p-1}\overline{\mathcal{E}_i}\right)$. To lower bound $\Pr[\mathcal{E}]$, we need to upper bound the probability of $\mathcal{E}_i$ for $1 \leq i \leq p-1$.

Let f be the density function of μ as in (1), and let $\mathsf{Leb}(\cdot)$ be the Lebesgue measure in n dimensions. Then for every i,

$$\Pr[\mathcal{E}_i \mid \mathcal{E}'_0] = \mu(\lambda_i - \lambda_{i+1} < \gamma) \leq \mathsf{Leb}(\lambda_i - \lambda_{i+1} < \gamma) \cdot |f|_\infty = O(\gamma/n) \cdot |f|_\infty$$

Note that conditioning on $\mathcal{E}_0$ such that $\lambda_p \geq 0.01/\sqrt{n}$, the density function f is bounded as:

$$|f|_\infty \leq O\left(\exp\left(-\frac{1}{2}\lambda_1\right)\left(100\sqrt{n}\right)^{p/2}\lambda_1^{p^2/2}\right) = 2^{O\left(p^2\log n\right)}$$

As a result, we get $\Pr\left[\mathcal{E}_i \wedge \mathcal{E}_0 \mid \mathcal{E}_0'\right] \leq \gamma \cdot 2^{O\left(p^2\log n\right)}$.

Therefore, the probability of $\mathcal{E}$ is lower bounded for sufficiently small $\gamma = 2^{-\Theta(p^2\log n)}$,

$$\begin{aligned}
\Pr\left[\mathcal{E}\right] &\geq \Pr\left[\mathcal{E}_0'\right] \cdot \Pr\left[\mathcal{E}_0 \wedge \left(\wedge_{i=1}^{p-1}\overline{\mathcal{E}_i}\right) \mid \mathcal{E}_0'\right] \\
&\geq \Pr\left[\mathcal{E}_0'\right] \cdot \left(\Pr\left[\mathcal{E}_0 \mid \mathcal{E}_0'\right] - \sum_{i=1}^{p-1}\Pr\left[\mathcal{E}_i \wedge \mathcal{E}_0 \mid \mathcal{E}_0'\right]\right) \\
&> (1-2p\exp(-32n)) \cdot \left(0.99000033 - (p-1)\gamma \cdot 2^{O\left(p^2\log n\right)}\right) > 0.9
\end{aligned}$$

◀

Conditioned on event $\mathcal{E}$ and recalling that $\lambda_i' \in \left[\lambda_i - O\left(\frac{np}{Z^2(n)}\right), \lambda_i + O\left(\frac{np}{Z^2(n)}\right)\right]$, the probability density of $\mathbf{\Lambda}'$ has only a negligible difference from that of $\mathbf{\Lambda}$, since the small disturbance of eigenvalues is dominated by the corresponding terms in $f(\mathbf{\Lambda})$.

$$\begin{aligned}
\frac{f(\mathbf{\Lambda}')}{f(\mathbf{\Lambda})} &= \frac{\exp\left(-\frac{1}{2}\sum_{i=1}^p \lambda_i'\right)\prod_{i=1}^p \lambda_i'^{-1/2}\prod_{1\leq i<j\leq p}(\lambda_i'-\lambda_j')}{\exp\left(-\frac{1}{2}\sum_{i=1}^p \lambda_i\right)\prod_{i=1}^p \lambda_i^{-1/2}\prod_{1\leq i<j\leq p}(\lambda_i-\lambda_j)} \\
&\leq \exp\left(\frac{p\cdot np}{Z^2(n)}\right)\left(\frac{\lambda_p - \frac{np}{Z^2(n)}}{\lambda_p}\right)^{-p/2}\prod_{1\leq i<j\leq p}\frac{\lambda_i-\lambda_j+\frac{2np}{Z^2(n)}}{\lambda_i-\lambda_j} \\
&\leq \exp\left(\frac{np^2}{Z^2(n)}\right)\cdot\left(1+\frac{np}{\lambda_p\cdot Z^2(n)}\right)^p\left(1+\frac{2np}{Z^2(n)\cdot\min_{i\neq j}|\lambda_i-\lambda_j|}\right)^{p(p-1)/2} \\
&\leq \exp\left(\frac{np^2}{Z^2(n)}\right)\cdot\left(1+\frac{100\sqrt{n}\cdot np}{Z^2(n)}\right)^p\left(1+\frac{2np}{Z^2(n)\cdot\gamma}\right)^{p(p-1)/2} = 1+O\left(\frac{np^3\gamma^{-1}}{Z^2(n)}\right)
\end{aligned}$$

Similarly we can prove $f(\mathbf{\Lambda}')/f(\mathbf{\Lambda}) \geq 1 - O\left(np^3\gamma^{-1}/Z^2(n)\right)$. Thus the total variation distance between $\mathbf{\Lambda}$ and $\mathbf{\Lambda}'$ conditioned on $\mathcal{E}$ is $\mathsf{D}_{\mathsf{TV}}\left(\mathbf{\Lambda}, \mathbf{\Lambda}' \mid \mathcal{E}\right) \leq O\left(np^3\gamma^{-1}/Z^2(n)\right) = O\left(1/n^2\right)$ for sufficiently large $Z(n) \geq (np)^{1.5}\gamma^{-0.5} = 2^{\Theta(p^2\log n)}$. Thus, for sufficiently large n, we have:

$$\mathsf{D}_{\mathsf{TV}}\left(\mathbf{\Lambda}, \mathbf{\Lambda}'\right) \leq \Pr[\overline{\mathcal{E}}] + \Pr[\mathcal{E}]\cdot\mathsf{D}_{\mathsf{TV}}\left(\mathbf{\Lambda}, \mathbf{\Lambda}' \mid \mathcal{E}\right) \leq 0.1 + O\left(1/n^2\right) < 0.11$$

Therefore, with as many as $q = p$ non-adaptive queries to the oracle matrix $\mathbf{M}$, the two distributions $\mathbf{M}_1$ and $\mathbf{M}_2$ cannot be distinguished with advantage greater than 0.11. At least $p+1$ queries are necessary to distinguish those two matrices $\mathbf{M}_1$ and $\mathbf{M}_2$ of rank $\leq p$ and rank n, respectively.

Indeed, the above argument holds for every constant advantage ε if $y = \varepsilon/3$, $t > \sqrt{12n/\varepsilon}$, and γ sufficiently small in the proof of Lemma 4, and let $Z(n)$ be sufficiently large. ◀

3.1.2 Equivalence Between Adaptive and Non-Adaptive Protocols

Now, we consider the adaptive query matrix $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_q]$ where $\mathbf{v}_i$ is the i-th query vector. Without loss of generality, we can assume that $\forall i, \mathbf{v}_i$ is a unit vector and it is orthogonal to query vectors $\mathbf{v}_1, \cdots, \mathbf{v}_{i-1}$. This gives us the following formal definition of an adaptive query protocol.

▶ **Definition 5.** *For a target matrix* $\mathbf{M}$*, an adaptive query protocol* P *will output a sequence of query vectors* $\mathbf{v}_1, \mathbf{v}_2, \cdots$*. It is called a normalized adaptive protocol if for any* i*, the query vector* $\mathbf{v}_i$ *output by* P *satisfies*

1. $\mathbf{v}_i$ *is a unit vector;*
2. $\mathbf{v}_i$ *is orthogonal to the vectors* $\mathbf{v}_1, \cdots, \mathbf{v}_{i-1}$*;*
3. $\mathbf{v}_i$ *is deterministically determined by* $\mathbf{M} \times [\mathbf{v}_1, \ldots, \mathbf{v}_{i-1}]$*.*

Let P^{std} be a standard protocol which outputs $\mathbf{e}_1, \mathbf{e}_2, \cdots$ where $\mathbf{e}_i$ is the i-th standard basis. We then show that adaptivity is unnecessary by proving that P^{std} has the same power as any normalized adaptive protocol.

More formally, we show the following lemma:

▶ **Lemma 6.** *Fix any* $n \times p$ *matrix* $\mathbf{U}$ *and any normalized adaptive protocol* P*. Let* $\mathbf{G}^T$ *be a* $p \times n$ *i.i.d Gaussian matrix. Fix* $q \leq n$ *to be the number of queries. Let matrix* $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_q]$ *and* $\mathbf{V}^{std} = [\mathbf{e}_1, \cdots, \mathbf{e}_q]$ *be the query matrix outputed by protocol* P *and* P^{std}*, correspondingly. Then, the probability distribution of* $\mathbf{UG}^T\mathbf{V}$ *is the same as the distribution of* $\mathbf{UG}^T\mathbf{V^{std}}$*.*

Proof. Since $\mathbf{G}^T\mathbf{V^{std}}$ is i.i.d Gaussian, it is enough to show $\mathbf{G}^T\mathbf{V}$ is also i.i.d Gaussian. We will show it column-by-column.

Denote $\mathbf{V}_i = [\mathbf{v}_1, \ldots, \mathbf{v}_i]$ and $\mathbf{V}_i^{std} = [\mathbf{e}_1, \cdots, \mathbf{e}_i]$. Note that $\mathbf{v}_1, \ldots, \mathbf{v}_q$ are unit vectors and orthogonal to each other. We first define unitary rotation matrices $R_1, R_2, \cdots$ recursively as follows. The matrix R_1 will take $\mathbf{v}_1$ to $\mathbf{e}_1$. The matrix R_i will take $\mathbf{e}_j$ to $\mathbf{e}_j$ for any $j < i$ and takes $R_{i-1} \cdots R_1 \mathbf{v}_i$ to $\mathbf{e}_i$. Note, R_i only depends on the first i query vectors. We have $R_i \cdots R_1 \mathbf{V}_i = \mathbf{V}_i^{std}$ for any $i \leq q$, and $\mathbf{G}^T\mathbf{V} = \mathbf{G}^T \cdot R_1^{-1} \cdots R_q^{-1} \cdot \mathbf{V}^{std}$. In the following, we use induction to show $\mathbf{G}^T \cdot R_1^{-1} \cdots R_i^{-1} \cdot \mathbf{V}_i^{std}$ is i.i.d Gaussian for any $i \leq q$.

For $i = 1$, since R_1 is determined by $\mathbf{v}_1$ which is independent of $\mathbf{G}^T$ and R_1 is a unitary matrix, $\mathbf{G}^T R_1^{-1}$ is i.i.d Gaussian. Thus, $\mathbf{G}^T R_1^{-1} \times \mathbf{V}_1^{std}$ is the first column which is also i.i.d Gaussian.

Now, suppose $\mathbf{G}^T \cdot R_1^{-1} \cdots R_i^{-1} \cdot \mathbf{V}_i^{std}$ is i.i.d Gaussian. We will prove $\mathbf{G}^T \cdot R_1^{-1} \cdots R_{i+1}^{-1} \cdot \mathbf{V}_{i+1}^{std}$ is also i.i.d Gaussian. Let $\mathbf{G'} = \mathbf{G}^T \cdot R_1^{-1} \cdots R_i^{-1}$ which is i.i.d Gaussian. Since R_{i+1} is determined by $\mathbf{v}_1, \cdots, \mathbf{v}_{i+1}$, it is determined by the response of the first i queries, that is, determined by $\mathbf{UG}^T\mathbf{V}_i = \mathbf{UG'V}_i^{std}$. It means R_{i+1} is determined by the first i columns of $\mathbf{UG'}$. Therefore, it is dependent on the first i columns of $\mathbf{G'}$. On the other hand, $R_{i+1}\mathbf{e}_j = \mathbf{e}_j$ for any $j \leq i$, and thus $R_{i+1}^{-1} = \begin{bmatrix} I_i & 0 \\ 0 & R' \end{bmatrix}$ where I_i is the $i \times i$ identity matrix, and R' depends on the first i columns of $\mathbf{G'}$. Consequently, in the multiplication of $\mathbf{G'} \times R_{i+1}^{-1}$, the first i columns are the same as those in $\mathbf{G'}$. In the $(i+1)$-th column, the a-th element is $\sum_{b \geq i+1} g'_{ab} r'_{b,i+1}$ where $g'_{ab}, r'_{b,i+1}$ are the elements in $\mathbf{G'}, R'$ correspondingly. Since $r'_{b,i+1}$ only depends on the first i columns of $\mathbf{G'}$, it is independent of g'_{ab} when $b \geq i+1$. Thus, the $(i+1)$-th column is also i.i.d Gaussian and independent of the first i columns. Therefore, we show $\mathbf{G}^T \cdot R_1^{-1} \cdots R_{i+1}^{-1} \cdot \mathbf{V}_{i+1}^{std}$ is still i.i.d Gaussian.

By induction $\mathbf{G}^T\mathbf{V}$ is i.i.d Gaussian. This finishes our proof. ◀

We then show for $\mathbf{M}_2 = \mathbf{U} \times \mathbf{G}^T + \frac{1}{\mathsf{poly}(n)} \cdot \mathbf{U}^\perp \times \mathbf{H}^T$, adaptivity is also unnecessary by a similar argument.

▶ **Corollary 7.** *Consider* $\mathbf{M}_2 = \mathbf{U} \times \mathbf{G}^T + \frac{1}{\mathsf{poly}(n)} \cdot \mathbf{U}^\perp \times \mathbf{H}^T$*. For any fixed* $\mathbf{U}, \mathbf{U}^\perp$*, and any fixed normalized adaptive protocol* P*,* $\mathbf{M}_2\mathbf{V}$ *has the same distribution as* $\mathbf{M}_2\mathbf{V}^{std}$*.*

Proof. It is enough to show both $\mathbf{G}^T \cdot \mathbf{V}$ and $\mathbf{H}^T \cdot \mathbf{V}$ are i.i.d Gaussian. ◀

Combining these results and Theorem 2, together with Yao's minimax principle [30],

▶ **Theorem 8.** *Let constant $\varepsilon > 0$ be the error tolerance and let $\mathbf{M}$ be an $n \times n$ oracle matrix with adaptive queries. For every integer $p < n$, at least $p+1$ queries are necessary for any randomized algorithm to distinguish whether $\mathsf{rank}(\mathbf{M}) \leq p$ or $\mathsf{rank}(\mathbf{M}) \geq p+1$ with advantage $\geq \varepsilon$.*

3.2 Lower Bound for Trace Estimation

We lower bound the number of queries needed to approximate the trace $\mathsf{tr}(\mathbf{M})$ of a matrix $\mathbf{M}$. In particular we reduce this problem to triangle detection as will be proved in Theorem 14.

▶ **Theorem 9.** *For any integer $C > 0$ and symmetric $n \times n$ matrix $\mathbf{M}$ with entries in $\{0, 1, 2, \ldots, n^3\}$, the number of possibly adaptively chosen query vectors, with entries in $\{0, 1, 2, \ldots, n^C\}$, needed to approximate $\mathsf{tr}(\mathbf{M})$ up to any relative error, is $\Omega(n/\log n)$.*

Proof. Suppose we had a possibly adaptive query algorithm making $q(n)$ queries which for a symmetric matrix $\mathbf{M}$, could approximate $\mathsf{tr}(\mathbf{M})$ up to any relative error. If $\mathbf{M} = \mathbf{A}^3$ for a symmetric matrix $\mathbf{A}$, we can run the trace esimation algorithm on $\mathbf{M}$ as follows: if $\mathbf{x}_1$ is the first query, we compute $\mathbf{A}\mathbf{x}_1$, then $\mathbf{A}(\mathbf{A}\mathbf{x}_1)$, then $\mathbf{A}(\mathbf{A}(\mathbf{A}\mathbf{x}_1)) = \mathbf{A}^3\mathbf{x}_1$. This then determines the second query $\mathbf{x}_2$, and we similarly compute $\mathbf{A}\mathbf{x}_2$, then $\mathbf{A}(\mathbf{A}\mathbf{x}_2)$, then $\mathbf{A}(\mathbf{A}(\mathbf{A}\mathbf{x}_2)) = \mathbf{A}^3\mathbf{x}_2$, etc. Thus, given only query access to $\mathbf{A}$, we can simulate the algorithm on $\mathbf{M} = \mathbf{A}^3$ with $3q(n)$ adaptive queries.

Now, it is well known that for an undirected graph G with adjacency matrix $\mathbf{A}$, the trace $\mathsf{tr}(\mathbf{A}^3)/6$ is the number of triangles in G. By the argument above, it follows that with $3q(n)$ queries to $\mathbf{A}$, we can determine if G has a triangle or has no triangles. On the other hand, by Theorem 14 below, at least $\Omega(n/\log n)$ queries to $\mathbf{A}$ are necessary for any adaptive algorithm to decide if there is a triangle in G. Therefore $3q(n) = \Omega(n/\log n)$ and hence we complete the proof with $q(n) = \Omega(n/\log n)$. ◀

3.3 Deciding if M is a Symmetric Matrix

▶ **Theorem 10.** *Given an $n \times n$ matrix $\mathbf{M}$ over any finite field or over fields $\mathbb{R}$ or $\mathbb{C}$, $O(\log(\frac{1}{\varepsilon}))$ queries are enough to test whether $\mathbf{M}$ is symmetric or not with probability $1-\varepsilon$.*

Proof. We choose two random vectors $\mathbf{u}$ and $\mathbf{v}$, where over a finite field we choose from a uniform distribution and over fields $\mathbb{R}$ or $\mathbb{C}$ we choose the Gaussian distribution. We then compute $\mathbf{M}\mathbf{u}$ and $\mathbf{M}\mathbf{v}$. We declare $\mathbf{M}$ to be symmetric if and only if $\mathbf{u}^T \cdot \mathbf{M}\mathbf{v} = \mathbf{v}^T \cdot \mathbf{M}\mathbf{u}$. It is easy to check that if $\mathbf{M}$ is symmetric, the test will succeed. We then show if $\mathbf{M}$ is not symmetric, $\mathbf{u}^T\mathbf{M}\mathbf{v} \neq \mathbf{v}^T\mathbf{M}\mathbf{u}$ with constant probability, so we obtain success probability $1-\varepsilon$ by repeating the test $O(\log(\frac{1}{\varepsilon}))$ times.

Let $\mathbf{A} = \mathbf{M} - \mathbf{M}^T$. When $\mathbf{M}$ is not symmetric, $\mathbf{A}$ is not 0. Thus, $\mathbf{u}^T\mathbf{M}\mathbf{v} = \mathbf{v}^T\mathbf{M}\mathbf{u}$ means $\mathbf{u}^T\mathbf{A}\mathbf{v} = 0$. We can treat this as a degree-2 polynomial in the entries of $\mathbf{v}^T$ and $\mathbf{u}$, i.e., this is $\sum_{i,j} u_i v_j \mathbf{A}_{i,j} = \sum_i u_i \sum_j v_j \mathbf{A}_{i,j}$. Thus, this is a non-zero polynomial and has at most constant probability of evaluating to 0 for any underlying field. To see this, for each i, let $t_i = \sum_j v_j \mathbf{A}_{i,j}$. Then there will be at least one t_i which is non-zero with probability at least $1/2$, for any underlying field. So now we get $\sum_i u_i t_i$. Fix all the u_i except u_i for a given t_i that is non-zero. Then we obtain $S + u_i t_i$. Then if u_i has at least two possible values, this is 0 in one case and non-zero in the other case. So we obtain a probability of at least $1/4$ of detection overall. ◀

4 Streaming and Statistics Problems

In this section we discuss testing an all ones column/row and identical columns/rows. Our results for row norms and majority/parity of columns/rows are in the full version.

4.1 Testing Existence of an All Ones Column/Row

Given a matrix $\mathbf{M} \in \{0,1\}^{m\times n}$, we want to test if $\mathbf{M}$ has a column (or row) with all 1 entries. It is trivial to test whether $\mathbf{M}$ has an all 1 column (or row) using n queries, *e.g.* $\mathbf{e}_1, \ldots, \mathbf{e}_n$. We consider this problem both over $\mathbb{F}[2]$ and $\mathbb{R}$. Note in the case over $\mathbb{R}$, if we allow an arbitrary query vector, we can set one query $\mathbf{v} = \{1, 2, 4, 8, ...2^n\}$, and then reconstruct $\mathbf{M}$ exactly. Thus, in order to avoid such trivial cases, we also restrict the entries in the query to be in $\{0, 1, 2, \ldots, n^C\}$.

For testing the existence of an all ones column, we reduce the problem to the communication complexity of Disjointness. Disjointness requires $\Omega(n)$ bits of communication to decide whether two sets with characteristic vectors $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$ are disjoint with constant probability. Suppose the fist $m-1$ rows in $\mathbf{M}$ equal $\mathbf{x}^T$ while the last row equals $\mathbf{y}^T$. If we can decide whether $\mathbf{M}$ has an all ones column with q non-adaptive queries $\mathbf{v}_1, \ldots, \mathbf{v}_q$, then we obtain a protocol for Disjointness with communication q by letting Alice send a message $\left(\mathbf{x}^T\mathbf{v}_1, \ldots, \mathbf{x}^T\mathbf{v}_q\right)$. Thus from the communication complexity lower bound of Disjointness, $q = \Omega(n)$ queries over $\mathbb{F}[2]$ are necessary to test if there is an all ones column in $\mathbf{M}$, which shows that the naïve algorithm is already optimal. For queries over $\mathbb{R}$, note that each entry $\mathbf{x}^T\mathbf{v}_j$ in the message is represented with $\log n$ bits, and as a result $q \geq \Omega\left(n/\log n\right)$.

Testing the existence of an all ones row with queries over $\mathbb{R}$ is trivial deterministically by querying $\mathbf{v} = (1, 1, \ldots, 1)$. Next we study the query complexity of testing an all 1s row deterministically with queries over $\mathbb{F}[2]$. With any $q \leq n-1$ queries $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_q]$, there is a non-zero vector $\mathbf{z} \neq \mathbf{0}$ such that $\mathbf{z}^T\mathbf{V} = \mathbf{0}$. Therefore the query matrix $\mathbf{V}$ cannot distinguish whether a row is from $\mathbf{x}^T$ or $\mathbf{x}^T + \mathbf{z}^T$. However, $\mathbf{x}^T$ and $\mathbf{x}^T + \mathbf{z}^T$ cannot be both all 1 rows, and hence n queries are necessary. This result also shows that the query complexity of the same problem over different fields might be quite different. We note for randomized algorithms, $O(\log(1/\epsilon))$ queries suffice over $\mathbb{F}[2]$ since the inner product of a row which is not all 1s disagrees with the parity of the query with probability $1/2$.

Evaluating the OR/AND function of columns/rows of a Boolean matrix can be reduced to testing existence of an all 1 or all 0 column/row, and hence the same bounds follow.

4.2 Identical Columns/Rows

Given an $m \times n$ matrix $\mathbf{M}$, we want to test whether $\mathbf{M}$ has two identical columns or rows. The trivial solution naively retrieves all information with n queries (column vectors).

Testing identical columns can be reduced to Disjointness. Suppose Alice and Bob have $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$. Let Alice expand her vector $\mathbf{x}$ to an $\frac{m}{2} \times n$ matrix $\mathbf{M}_1$ as follows: the first row is $(1, \mathbf{x}^T) = (1, x_1, \ldots, x_n)$; for $2 \leq i \leq \frac{m}{2}$ the i-th row is $(1, z_1^{(i)}, \ldots, z_n^{(i)})$ where $z_j^{(i)} = 1$ if $x_j = 1$, and $z_j^{(i)}$ is uniformly random over $\{0,1\}$ if $x_j = 0$, for $1 \leq j \leq n$. Bob expands his vector $\mathbf{y}$ to $\mathbf{M}_2$ similarly. Putting $\mathbf{M}_1, \mathbf{M}_2$ together, we let $\mathbf{M} \stackrel{\text{def}}{=} \begin{bmatrix}\mathbf{M}_1 \\ \mathbf{M}_2\end{bmatrix}$. Then $\mathbf{M}$ is an $m \times (n+1)$ matrix with the first column being all 1s. For $j \geq 2$, the j-th column is all 1s if and only if $x_j = y_j = 1$, in which case $\mathbf{M}$ has two identical rows of all 1 entries. For columns where x_j, y_j are not both equal to 1, without loss of generality we may assume the j-th and j'-th columns satisfy $x_j = x_{j'} = 0$ and $y_j = y_{j'}$. Then two columns are identical only

if $(z_j^{(2)}, \ldots, z_j^{(\frac{m}{2})}) = (z_{j'}^{(2)}, \ldots, z_{j'}^{(\frac{m}{2})})$, which happens with probability $\leq 1/2^{\frac{m}{2}-1}$. Therefore the overall probability of two not-all-ones columns in $\mathbf{M}$ being identical is bounded by $m^2/2^{m/2} = 2^{-\Omega(m)}$.

That is, except for an exponentially small error $2^{-\Omega(m)}$, two identical columns in $\mathbf{M}$ are both all ones columns, which turns out to be equivalent to the case that two vectors $\mathbf{x}, \mathbf{y}$ held by Alice and Bob are not disjoint. Then, because DISJOINTNESS requires $\Omega(n)$ bits of communication, at least $\Omega(n)$ oracle queries to $\mathbf{M}$ are necessary. To test identical rows with error ε, if suffices to make $q = O(\log(m/\varepsilon))$ random queries with each entry uniform random over $\{0, 1\}$. Since for every pair of distinct rows, a random query distinguishes them with probability $\frac{1}{2}$, with $\lceil \log(m^2/\varepsilon) \rceil$ queries each pair of distinct rows is miscounted as identical with probability $\leq \varepsilon/m^2$. By a union bound, the overall false-positive error is bounded by $\frac{\varepsilon}{m^2} \cdot \binom{m}{2} < \varepsilon$, while there is no false-negative error since for all queries, identical rows always lead to identical outputs.

5 Graph Problems

5.1 Connectivity

▶ **Theorem 11.** *Given the bipartite adjacency matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$ of a graph, we need $\Omega(n/\log n)$ queries to decide whether the graph is connected with constant probability*

Proof. Consider two row vectors $\mathbf{u}, \mathbf{v} \in \{0, 1\}^{n-1}$ and construct matrix $\mathbf{A}$ as follows. Firstly, the first $n/2$ rows of $\mathbf{A}$ equals to $\mathbf{u}$ and the rest are equal to $\mathbf{v}$, then add an all 1s column to $\mathbf{A}$. Now, matrix $\mathbf{A}$ can be treated as a bipartite adjacency matrix of a graph G with n vertices in each part, where $\mathbf{A}_{i,j} = 1$ iff there is an edge from the i-th left vertex to the j-th right vertex. Then G is disconnected if and only if the two vectors $\mathbf{u}$ and $\mathbf{v}$ are 0 on the same position. Thus any algorithm that uses $q(n)$ non-adaptive queries on the right of $\mathbf{A}$ to decide the connectivity of G immediately implies a protocol for set disjointness, provided we replace 1s with 0s in the input characteristic vectors to the set disjointness problem. So the communication is at most $q(n) \log n$, thus $q(n) = \Omega(n/\log n)$. ◀

▶ **Theorem 12.** *Given the signed edge-vertex incidence matrix $\mathbf{M} \in \{0, \pm 1\}^{n \times \binom{n}{2}}$ of a graph G with n vertices, the connectivity of G can be decided with $\mathsf{polylog}(n)$ non-adaptive queries.*

This follows from the main theorem of [16]. By the following theorem, every cut of G is multiplicatively approximated and hence G is connected iff H is connected, since a graph is disconnected iff it has a zero cut.

▶ **Theorem 13** ([16])**.** *There is a distribution on $\binom{n}{2} \times \mathsf{polylog}(n)$ matrices $\mathbf{S}$ such that from $\mathbf{MS}$, one can construct a (1 ± 0.1)-sparsifier H of the graph G with constant probability. Here, $\mathbf{x}^T \mathbf{L}_G x = (1 \pm .1)\mathbf{x}^T \mathbf{L}_H x$ for all x, with constant probability, where $\mathbf{L}_G$ and $\mathbf{L}_H$ are the corresponding graph Laplacians.*

5.2 Triangle Detection

▶ **Theorem 14.** *If an $n \times n$ matrix $\mathbf{A}$ is the adjacency matrix of a graph G, then determining whether G contains a triangle or not requires $\Omega(n/\log n)$ queries, even for randomized algorithms succeeding with constant probability.*

See the full version for a proof using communication complexity.

6 Conclusions

We initiated the study of querying a matrix through matrix-vector products. We illustrated that for some quantities, if one can only query matrix-vector products on one side, the problem becomes harder. We also illustrated the importance of the underlying field defining the matrix-vector products, as well as the representation of the graph for graph problems. Given connections to sketching algorithms, streaming, and compressed sensing, we believe this area deserves its own study. Some interesting open questions are for computing matrix norms, such as Schatten-p norms, for which tight bounds in streaming and communication complexity models remain elusive; for recent work on this see [21, 22, 9]. Given the success of our model in proving lower bounds for approximate rank, which we also do not have streaming or communication lower bounds for, perhaps tight bounds in our query model are possible for matrix norms. Such bounds may give insight for other models.

References

1 Yuqing Ai, Wei Hu, Yi Li, and David P. Woodruff. New Characterizations in Turnstile Streams with Applications. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 20:1–20:22, 2016.

2 Noga Alon, Yossi Matias, and Mario Szegedy. The Space Complexity of Approximating the Frequency Moments. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 20–29, 1996.

3 Sepehr Assadi, Sanjeev Khanna, and Yang Li. On Estimating Maximum Matching Size in Graph Streams. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1723–1742, 2017.

4 Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum Matchings in Dynamic Graph Streams and the Simultaneous Communication Model. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1345–1364, 2016.

5 Khanh Do Ba, Piotr Indyk, Eric Price, and David P. Woodruff. Lower Bounds for Sparse Recovery. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1190–1197, 2010.

6 Maria-Florina Balcan, Yi Li, David P. Woodruff, and Hongyang Zhang. Testing Matrix Rank, Optimally. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 727–746, 2019.

7 Anders Björner, László Lovász, and Andrew CC Yao. Linear decision trees: volume estimates and topological bounds. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 170–177. ACM, 1992.

8 Eric Blais, Joshua Brody, and Kevin Matulef. Property Testing Lower Bounds via Communication Complexity. *Computational Complexity*, 21(2):311–358, 2012.

9 Vladimir Braverman, Stephen R. Chestnut, Robert Krauthgamer, Yi Li, David P. Woodruff, and Lin Yang. Matrix Norms in Data Streams: Faster, Multi-Pass and Row-Order. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 648–657, 2018.

10 Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(8):1207–1223, 2006.

11 Talya Eden, Amit Levi, Dana Ron, and C Seshadhri. Approximately counting triangles in sublinear time. *SIAM Journal on Computing*, 46(5):1603–1646, 2017.

12 Piotr Indyk, Eric Price, and David P. Woodruff. On the Power of Adaptivity in Sparse Recovery. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 285–294, 2011.

13 Akshay Kamath and Eric Price. Adaptive Sparse Recovery with Limited Adaptivity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2729–2744, 2019.

14 Daniel M. Kane, Shachar Lovett, and Shay Moran. Near-optimal linear decision trees for k-SUM and related problems. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 554–563, 2018.

15 Sampath Kannan, Elchanan Mossel, Swagato Sanyal, and Grigory Yaroslavtsev. Linear Sketching over F_2. In *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, pages 8:1–8:37, 2018.

16 Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single Pass Spectral Sparsification in Dynamic Streams. *SIAM J. Comput.*, 46(1):456–477, 2017.

17 John T Kent and R J Muirhead. Aspects of Multivariate Statistical Theory. *The Statistician*, 33(2):251, 1984.

18 Christian Konrad. Maximum Matching in Turnstile Streams. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 840–852, 2015.

19 Yi Li, Huy L. Nguyen, and David P. Woodruff. On Sketching Matrix Norms and the Top Singular Vector. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1562–1581, 2014.

20 Yi Li, Huy L. Nguyen, and David P. Woodruff. Turnstile streaming algorithms might as well be linear sketches. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 174–183, 2014.

21 Yi Li and David P. Woodruff. On approximating functions of the singular values in a stream. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 726–739, 2016.

22 Yi Li and David P. Woodruff. Embeddings of Schatten Norms with Applications to Data Streams. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 60:1–60:14, 2017.

23 Vasileios Nakos, Xiaofei Shi, David P. Woodruff, and Hongyang Zhang. Improved Algorithms for Adaptive Compressed Sensing. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 90:1–90:14, 2018.

24 Eric Price and David P. Woodruff. Lower Bounds for Adaptive Sparse Recovery. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 652–663, 2013.

25 Jianhong Shen. On the singular values of Gaussian random matrices. *Linear Algebra and its Applications*, 326:1–14, 2001.

26 Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *Compressed Sensing: Theory and Applications*, November 2010. doi:10.1017/CBO9780511794308.006.

27 Martin Wainwright. High-dimensional statistics: A non-asymptotic viewpoint. URL: https://www.stat.berkeley.edu/~mjwain/stat210b/Chap2_TailBounds_Jan22_2015.pdf.

28 Hermann Weyl. Das asymptotische Verteilungsgesetz der Eigenwerte linearer partieller Differentialgleichungen (mit einer Anwendung auf die Theorie der Hohlraumstrahlung). *Mathematische Annalen*, 71(4):441–479, 1912.

29 David P. Woodruff. Sketching as a Tool for Numerical Linear Algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1-2):1–157, 2014.

30 Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science, FOCS 1977*, pages 222–227. IEEE, 1977.

31 Qiaochu Yuan. Singular value decomposition, 2017. URL: `https://qchu.wordpress.com/2017/03/13/singular-value-decomposition/`.

Dynamic Ordered Sets with Approximate Queries, Approximate Heaps and Soft Heaps

Mikkel Thorup
Department of Computer Science, University of Copenhagen, Denmark
mikkel2thorup@gmail.com

Or Zamir
Blavatnik School of Computer Science, Tel Aviv University, Israel
orzamir@mail.tau.ac.il

Uri Zwick
Blavatnik School of Computer Science, Tel Aviv University, Israel
zwick@tau.ac.il

Abstract

We consider word RAM data structures for maintaining ordered sets of integers whose SELECT and RANK operations are allowed to return *approximate* results, i.e., ranks, or items whose rank, differ by less than Δ from the exact answer, where $\Delta = \Delta(n)$ is an error parameter. Related to approximate SELECT and RANK is approximate (one-dimensional) NEAREST-NEIGHBOR. A special case of approximate SELECT queries are approximate MIN queries. Data structures that support approximate MIN operations are known as *approximate heaps* (priority queues). Related to approximate heaps are *soft heaps*, which are approximate heaps with a different notion of approximation.

We prove the optimality of all the data structures presented, either through matching cell-probe lower bounds, or through equivalences to well studied static problems. For approximate SELECT, RANK, and NEAREST-NEIGHBOR operations we get matching cell-probe lower bounds. We prove an equivalence between approximate MIN operations, i.e., approximate heaps, and the static *partitioning* problem. Finally, we prove an equivalence between soft heaps and the classical *sorting* problem, on a smaller number of items.

Our results have many interesting and unexpected consequences. It turns out that approximation greatly speeds up some of these operations, while others are almost unaffected. In particular, while SELECT and RANK have identical operation times, both in comparison-based and word RAM implementations, an interesting separation emerges between the approximate versions of these operations in the word RAM model. Approximate SELECT is much faster than approximate RANK. It also turns out that approximate MIN is *exponentially* faster than the more general approximate SELECT. Next, we show that implementing soft heaps is harder than implementing approximate heaps. The relation between them corresponds to the relation between sorting and partitioning.

Finally, as an interesting byproduct, we observe that a combination of known techniques yields a *deterministic* word RAM algorithm for (exactly) sorting n items in $O(n \log\log_w n)$ time, where w is the word length. Even for the easier problem of finding duplicates, the best previous deterministic bound was $O(\min\{n \log\log n, n \log_w n\})$. Our new unifying bound is an improvement when w is sufficiently large compared with n.

2012 ACM Subject Classification Theory of computation → Data structures design and analysis

Keywords and phrases Order queries, word RAM, lower bounds

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.95

Category Track A: Algorithms, Complexity and Games

Funding Mikkel Thorup's research is supported by his Advanced Grant DFF-0602-02499B from the Danish Council for Independent Research and by his Investigator Grant 16582, Basic Algorithms Research Copenhagen (BARC), from the VILLUM Foundation. Part of this research was carried out while Or Zamir and Uri Zwick were visiting BARC, Copenhagen, funded by the VILLUM Foundation, grant 16582. Part of the research was carried out while Uri Zwick was visiting IRIF, Paris, with support from the FSMP.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 95; pp. 95:1–95:13

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

1 Introduction

A data structure for maintaining *dynamic ordered sets* supports the operations INSERT, DELETE, and either, or both, of the operations SELECT and RANK. INSERT inserts an item, with an associated key, into the set. We assume that the keys of all items are distinct. DELETE receives a reference to an item in the set and deletes it from the set. SELECT receives an index i and returns the i-th item in the sorted order of the items currently in the set. RANK receives an item, not necessarily in the set, and returns its *rank*, i.e., the number of items in the set whose keys are smaller than the key of the item. If both SELECT and RANK operations are implemented, then it is easy to use them to support other operations such as PREDECESSOR, SUCCESSOR and NEAREST-NEIGHBOR. More specifically, PREDECESSOR(x) = SELECT(RANK(x)), SUCCESSOR(x) = SELECT(RANK$(x)+1$), and NEAREST-NEIGHBOR(x) is the closer of these two to x. (We assume here that x is not in the set.)

In the comparison-based model, each operation can be implemented in $O(\log n)$ time, and this is optimal. In this paper, however, our main focus is the word RAM model. Assuming integer keys, it models what can be implemented in a programming language such as C [18] which has been used for efficient portable code since 1978. Word operations take constant time. The word size w, measured in bits, is a unifying parameter of the model. All integers considered are assumed to fit in a word. If $|S| = n$, where S is the set of items in the data structure, then we assume that $w \geq \log n$, so that we can at least index the items in S. Integers can not only be compared. We can use all the standard arithmetic and bit-wise operations. Moreover, the random access memory of the word RAM implies that we can allocate tables or arrays of words, accessing entries in constant time using indices that may be computed from keys. These word RAM features are used in many classic algorithms, e.g., radix sort from 1929 [7] and hash tables from 1956 [9]. We note that by handling integer keys, we immediately handle floating point keys since their standard bit-representation (sign,exponent,mantissa) is such that casting them as integers gives the right ordering.

In the word RAM model, a more diverse picture emerges than in the comparison-based model. Pătraşcu and Thorup [20] obtained a data structure that supports all the above operations in $O(\log n/\log w)$ time. For SELECT and RANK this matches a lower bound of Fredman and Saks [12] which holds even if we only want to support SELECT or only want to support RANK. However, if we only want to support PREDECESSOR, SUCCESSOR and/or NEAREST-NEIGHBOR (plus INSERT and DELETE), then Andersson and Thorup [2] have shown that this can done in $O\sqrt{\log n/\log\log n}$) time per operation. This is the best possible bound in terms of n, matching a static lower bound of Beame and Fich [3].

In this paper, we consider word RAM data structures for maintaining ordered sets of integers where SELECT and RANK are allowed to return *approximate* results, i.e., ranks, or items whose rank, differ by *less* than Δ from the exact answer, where $\Delta = \Delta(n)$ is an error parameter. (When $\Delta = 1$, the data structure has to return exact results.) We use Δ-SELECT as a shorthand for Δ-approximate SELECT, and similarly for the other operations.

A Δ-MIN operation is an operation that returns one of the Δ smallest items in the ordered set. This is clearly a special case of Δ-SELECT operations. A data structure that supports INSERT, DELETE and Δ-MIN operations is known as an *approximate heap*, or Δ-heap.

An EXTRACT-Δ-MIN operation finds one of the smallest Δ items, using a Δ-MIN operation, returns it, and deletes it from the Δ-heap, using a DELETE operation. Our Δ-heap data structure satisfies the following *fairness* condition: an item cannot be one of the Δ smallest items in the data structure for 2Δ consecutive EXTRACT-Δ-MIN operations without being returned, and deleted, by one of the these EXTRACT-Δ-MIN operations.

Related to approximate heaps are *soft heaps*, introduced by Chazelle [6], which are, in a sense, approximate heaps with a different notion of approximation. Soft heaps were used by Chazelle [5] to obtain the fastest deterministic algorithm for computing minimum spanning trees in the comparison model. The implementation of soft heaps was simplified by Kaplan et al. [17]. Soft heaps were also used recently by Kaplan et al. [16] to obtain simplified optimal algorithms for some selection problems.

A *soft heap* is a heap data structure which is allowed to *increase* the keys of some of the items stored in the heap. Each item has both an *original* key, and a *current* key, which might be larger than its original key. An item whose current key is larger than its original key is said to be *corrupt*. A MIN operation returns an item with the minimum current key. (Once an item becomes corrupt, it remains corrupt, as its current key can only be increased. The user can examine the original and current keys of all items, including those of the item returned by a MIN operation.) A q-soft heap is a soft heap such that after any sequence that includes n INSERT operations, at most n/q of the items *in* the heap are corrupt. Clearly, an (n/Δ)-soft heap is also a Δ-heap. We show, however, that implementing an (n/Δ)-soft heap in the word RAM model is, in general, harder than implementing a Δ-heap.

Motivation. In addition to being natural computational problems that lead to many interesting, and perhaps unexpected, theoretical results, the problems we consider are also well motivated in practice. In many real life applications, keys are obtained using inexact measurements, or may change over time. We may be interested in 'sampling' items of given ranks, e.g., for statistical purposes, but typically when we ask for an item of rank i, an item whose rank is between $i - \Delta$ and $i + \Delta$, for a small enough Δ, will serve just as well. It is thus interesting to know whether allowing approximation can speed up such operations.

We note that the direct motivation for approximate versions of SELECT, RANK and MIN, is very different from the motivation of soft heaps which at first may look a bit peculiar, but which have proven to be useful data structure inside some important algorithms.

Our results

We show, among other things, that if $\Delta = n^\varepsilon$, for any $\varepsilon > 0$, then INSERT, DELETE and Δ-SELECT can be implemented in *constant time*.

While SELECT and RANK operations have the same running times in the word RAM model when exact results are required, an interesting separation emerges between the approximate versions of these operations. Surprisingly, Δ-SELECT is *easier* than Δ-RANK. We also show that Δ-MIN is "exponentially" faster than Δ-SELECT. As mentioned, we also show that soft heaps are harder to implement than approximate heaps.

Our results follow from a full characterization of the time needed for each subset of dynamic operations via either a matching cell-probe lower bound, or via an equivalence to the well-studied exact static problems of *ordered partition* and *sorting*.

More specifically, we obtain the following four main results:

(1) A data structure that supports INSERT, DELETE and Δ-SELECT in $O(\log n/\log(w\Delta))$ time, where n is the number of items in the set. We also obtain a matching lower bound that shows that at least one of these operations must take $\Omega(\log n/\log(w\Delta))$ time. For $\Delta = n^\varepsilon$, for any $\varepsilon > 0$, we get $O(1)$ time for each operations.

(2) An augmentation of the previous data structure that also supports Δ-RANK operations. INSERT, DELETE and Δ-SELECT operations still take $t_u = O(\log n/\log(w\Delta))$ time, while Δ-RANK takes $O\left(\log n/\log(w\Delta) + pred(\frac{n}{\Delta}, t_u\Delta, n, w)\right)$ time, where $pred(n, t, s, w)$ is the

time for dynamically answering exact *predecessor queries* on a set of n items when the update time is $O(t)$, the space of the data structure is $O(s)$, and the word length is w. (It is known, for example that $pred(n,t,n,w) = O\left(\sqrt{\frac{\log n}{\log\log n}}\right)$, for every $w \geq \log n$, where $t = O\left(\sqrt{\frac{\log n}{\log\log n}}\right)$, and that this is the optimal bound in terms of n.) We also provide a matching lower bound for the complexity of Δ-RANK operations that shows that the $pred(\frac{n}{\Delta}, t_u\Delta, n, w)$ term in the upper bound cannot be removed. For $\Delta = n^\varepsilon$, for any $\varepsilon > 0$, we get $O(1)$ time for Δ-SELECT, while Δ-RANK operations still cost $\Omega\left(\sqrt{\frac{\log n}{\log\log n}}\right)$. This exhibits a somewhat surprising separation between Δ-SELECT and Δ-RANK, given that the exact versions of these operations have the same running times.

(3) A three-way equivalence between (i) *approximate heaps*, (ii) *approximate sorting* and (iii) (exact) *ordered partition*: There is a Δ-heap with $O(P_\Delta(n,w))$ time per operation, where $P_\Delta(n,w)$ is *non-decreasing* in n, if and only if it is possible, for every n, to Δ-sort a set of n items in $O(nP_\Delta(n,w))$ time, if and only if it is possible, for every n, to partition n items into $\frac{n}{\Delta}$ sets of size roughly Δ in $O(nP_\Delta(n,w))$ time, such that the items in each set are smaller than the items in the next set. (A sequence is Δ-sorted if no item is at distance greater than Δ from its position in the sorted sequence.)

(4) A three-way equivalence between (i) *soft heaps*, (ii) *exact heaps* and (iii) exact sorting: There is q-soft heap with $O(t)$ time per operation, if and only if there is an *exact* heap holding up to q items with $O(t)$ time per operation, if and only if it is possible to sort up to q items in $O(t)$ time per item.

Han and Thorup [15] showed that n items can be ordered partitioned into $\sqrt{n}$ sets of size roughly $\sqrt{n}$ in $O(n)$ time. By iterating, if follows that n items can be partitioned into sets of size roughly $n^{2^{-k}}$ in $O(kn)$ time. Thus, n items can be partitioned into subsets of size Δ in $O(n\log(\frac{\log n}{\log\Delta}))$ time. If $\Delta < w$ we can do even better. We only do $\log\frac{\log n}{\log w}$ partitioning iterations after which we are left with sets of size roughly w. These sets can be completely sorted in linear time using the *dynamic fusion node* of Pătraşcu and Thorup [20]. Combining these two results, we get that partitioning into sets of size Δ can be done in $O(n\log(\frac{\log n}{\log(w\Delta)}))$ time. In particular, for $\Delta = 1$, we get exact sorting in $O(n\log\log_w n)$ time, a bound that has not been observed before. By (3), there is a Δ-heap with $O(\log(\frac{\log n}{\log(w\Delta)}))$ time per operation. By the lower bound in (1), Δ-SELECT requires $\Omega((\log n)/\log(w\Delta))$ time. It follows that $time(\Delta\text{-MIN}) = O(\log time(\Delta\text{-SELECT}))$, i.e., Δ-MIN is "exponentially" faster than Δ-SELECT. To get a constant time for Δ-MIN, we currently need $\Delta = n^\varepsilon$, for some $\varepsilon > 0$, as for Δ-SELECT. But, while the result for Δ-SELECT is optimal, and hence cannot be improved, improved partitioning algorithms could potentially yield $O(1)$ time of Δ-MIN for smaller values of Δ.

The bounds we give in this paper are amortized. However, all claimed time bounds can be made worst-case using the techniques of Andersson and Thorup [2]. All our data structures use linear space.

Dumitrescu [10] and Fredman [11] considered *comparison-based* data structures that support Δ-SELECT and Δ-MIN operations. The optimal time bounds for these operations are $\Theta(\log\frac{n}{\Delta})$. Δ-RANK operations can be easily supported within the same time bounds. Thus, there is no separation between Δ-SELECT and Δ-RANK in this model, and to get a constant time per operation, Δ has to be *linear* in n.

A summary of our results for Δ-SELECT, Δ-NEAREST-NEIGHBOR and Δ-RANK are given in Table 1. Update time refers to the time of INSERT and DELETE operations. The first row gives the result of a general value of Δ. The query times are optimal given the update times and assuming $O(n)$ space. (We have no proof that the same query times cannot be obtained with

Error	Update time	Δ-SELECT	Δ-NEAREST	Δ-RANK
Δ	$t_u = \Theta\left(\frac{\log n}{\log(w\Delta)}\right)$	$\Theta\left(\frac{\log n}{\log(w\Delta)}\right)$	$\Theta\left(pred(\frac{n}{\Delta}, t_u\Delta, n, w)\right)$	$\Theta\left(\begin{array}{c}\Delta\text{-SELECT} + \\ \text{NEAREST}\end{array}\right)$
$\Delta = \log n$	$\Theta\left(\frac{\log n}{\log\log n}\right)$	$\Theta\left(\frac{\log n}{\log\log n}\right)$	$\Theta\left(\sqrt{\frac{\log n}{\log\log n}}\right)$	$\Theta\left(\frac{\log n}{\log\log n}\right)$
$\Delta = \sqrt{n}$	$\Theta(1)$	$\Theta(1)$	$\Theta\left(\sqrt{\frac{\log n}{\log\log n}}\right)$	$\Theta\left(\sqrt{\frac{\log n}{\log\log n}}\right)$
$\Delta = \frac{n}{w}$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$

Figure 1 Running times for Δ-RANK, and Δ-SELECT and Δ-NEAREST-NEIGHBOR queries.

smaller update times.) The second and third rows specialize the results for the representative cases $\Delta = \log n$ and $\Delta = \sqrt{n}$, giving bounds that hold for all values of $w \geq \log n$. With regard to the second row, we note that the time bounds obtained for $\Delta = \log n$ are identical to the exact case, i.e., $\Delta = 1$. We also note that if only Δ-NEAREST-NEIGHBOR queries are to be answered, then the same query time of $\Theta\left(\sqrt{\frac{\log n}{\log\log n}}\right)$ can be obtained with a reduced update time of $\Theta\left(\sqrt{\frac{\log n}{\log\log n}}\right)$. (See [2].) In the third row $\Delta = \sqrt{n}$ can be replaced by $\Delta = n^\varepsilon$, for every $\varepsilon > 0$. Finally, in the forth row, we consider the case $\Delta = \frac{n}{w}$ (or equivalently $\Delta = \frac{n}{w^k}$, for any $k \geq 1$), where all update and query time drop down to a constant.

Chazelle [6], and Kaplan et al. [17], obtained $(1/\varepsilon)$-soft heaps with $O(\log\frac{1}{\varepsilon})$ amortized time per operation, which is optimal in the comparison-based model. In the word-RAM, we obtain by (4) $(1/\varepsilon)$-soft heaps with $O(\log\log\frac{1}{\varepsilon})$ amortized time per operation, or even $O(\sqrt{\log\log\frac{1}{\varepsilon}})$ amortized expected time per operation. We also obtain a w-soft heap with $O(1)$ time per operation.

The rest of the paper is organized as follows. In Section 2 we give a high-level description of the data structures and the equivalences obtained in this paper. The full details are given in the full version of the paper. In Section 3 we present our matching cell-probe lower bounds. We end in Section 4 with some concluding remarks and open problems.

2 High-level description of data structures and equivalences

In this section we sketch the techniques we use and give high level descriptions of the data structures and equivalences obtained in the paper.

2.1 Dynamic sets with approximate SELECT

Our goal is to obtain a data structure that supports INSERT, DELETE and Δ-SELECT operations in $O(\log n/\log(w\Delta))$ time per operation, which we later show is optimal. As mentioned, Pătraşcu and Thorup [20] obtained a data structure that supports INSERT, DELETE and exact SELECT operations in $O(\log n/\log w)$ time. Thus, we may assume that, say, $\Delta > w^3$ and devise a data structure that supports each operation in $O(\log n/\log\Delta)$ time.

Our data structure is built around a B-tree (see, e.g., [8]). The degree DEGREE$[v]$ of each node v in the tree is in the range $[\frac{1}{4}D, 4D]$, where $D = \Delta^{1/3}$. As with standard B-trees, a node v of degree d contains an array CHILD$[v]$ of size d with pointers to its d children,

a pointer PARENT[v] to its parent, and an array SPLIT[v] of $d-1$ *splitters*, $s_1, s_2, \ldots, s_{d-1}$. We also let $s_0 = -\infty$ and $s_d = \infty$. The keys of all items in the subtree of the i-th child CHILD[v][i] of v are all in the range $[s_i, s_{i+1})$, for $i = 0, 1, \ldots, d-1$. A non-root node v also contains its *index* INDEX[v] such that $v =$ CHILD[PARENT[v]][INDEX[v]], i.e., v is the INDEX[v]-th child of its parent. The leaves of a B-tree are all at the same depth.

The B-tree used differs from a standard B-tree in several important ways. The first is that the leaves of the tree do not contain single items, but rather *buckets* that contain between Δ and 2Δ items. These buckets are referred to as *leaf buckets*. The items in each leaf bucket are not sorted, but they all lie between the appropriate splitters in the non-leaf nodes of the tree. Second, each internal node v of the tree has a *buffer* BUFFER[v] associated with it. The size of each such buffer is at most $B = \Delta^{2/3}$. The operations on the B-tree are done *lazily*. An inserted item is simply placed at the buffer of the root. When a buffer is full, its items are partitioned according to the splitters stored in the node, and sent to the appropriate children. We refer to this operation as *flushing* the buffer.

All the items in the data structure reside in leaf buckets and buffers. The splitters in the internal nodes of the tree are copies of keys of items that belonged to the data structure at some stage.

The partitioning of the items in a buffer is done using the fast partitioning algorithm of Han and Thorup [15]. Their algorithm partitions q items according to $O(\sqrt{q})$ splitters in $O(q)$ time. The choice $B = \Delta^{2/3}$ and $D = \Delta^{1/3}$ ensures that a buffer can be flushed in $O(B)$ time. The use of this fast partitioning algorithm is the *only* place in which the data structure relies on the power of the word RAM model.

Another difference between the B-tree used and a standard one is that we impose explicit conditions on the *size* of each subtree. The size of a subtree is the total number of items in the leaf buckets and the buffers that belong to the subtree. The size of a subtree of height i (where the leaves are at height 0), is required to be in the range $[\frac{1}{2}\Delta, 2\Delta]D^i$. To maintain this condition, we store at each node v a $size[v]$ field that holds its current size. (To achieve that, $size[v]$ is updated by relevant INSERT and DELETE operations.) A simple calculation shows that the size condition implies that the degree of each node is in the range $[\frac{D}{4}, 4D]$. It also implies that the height h of the tree is at most $h \leq \log_D \frac{2n}{\Delta} \leq \frac{3 \log n}{\log \Delta}$, as $D = \Delta^{1/3}$. We thus need to show that the (amortized) cost of each INSERT, DELETE and Δ-SELECT is of the order of the depth of the tree.

To support approximate SELECT operations, we augment the B-tree by several additional components. To each non-leaf node v we add a *(rough) locator* array LOCATE[v] of size $\lfloor size_0[v]/(\frac{1}{2}\Delta D^{i-1}) \rfloor \leq 4D$, where $size_0[v]$ is the size of v when it was last rebuilt (see below). If v is at height i, where $i > 0$, then the j-entry of the array, for $j = 0, 1, \ldots, 4D-1$, contains the index of the child of v that contained the item of rank $\frac{1}{2}\Delta D^{i-1} \cdot j$ in the subtree rooted at v, i.e., the $(\frac{1}{2}\Delta D^{i-1} \cdot j)$-th item in the sorted order of all items in the subtree of v, when this subtree was last rebuilt. INSERT and DELETE operations performed after the last rebuilding of the subtree of v make the information in LOCATE[v] slightly inaccurate, but to an extent that can be tolerated, as we are only aiming for approximate results.

Finally, we also store at each non-leaf node v of degree d an array SUM[v] of size $d+1$ such that SUM[v][i], for $i = 0, 1, \ldots, d$, is the sum of sizes of the subtrees rooted at the first i children of v at the time the subtree of v was last rebuilt.

Using this augmented B-tree we can implement INSERT, DELETE and Δ-SELECT operations in $O(\log n / \log \Delta)$ amortized time. The challenge is to time the flushing of buffers and the rebuilding of subtrees so that, on the one hand, we do not spend too much time, and, on the other hand, the information in the B-tree is always sufficiently accurate so that the rank error in each SELECT operation is at most Δ.

Essentially, to locate an item whose rank is close to k, we navigate the tree using the LOCATE and SUM arrays. We start with v being the root and i being the height of the root. To find the child we need to descend to, we let $j \leftarrow \text{LOCATE}[v][\lfloor k/(\frac{1}{2}\Delta D^{i-1})\rfloor]$. It is easy to see that the k-th item in the subtree of v, at the time of the last rebuilding, is either contained in the j-th child of v, if $\text{SUM}[v][j] \leq i < \text{SUM}[v][j+1]$, or otherwise in the $(j+1)$-st child of v. (This follows as $k \leq (\frac{1}{2}\Delta D^{i-1})\lceil k/(\frac{1}{2}\Delta D^{i-1})\rceil$.) In the latter case, we increment j. We now descend to the j-th child of v, letting $v \leftarrow \text{CHILD}[v][j]$, $i \leftarrow i-1$, $k \leftarrow k - \text{SUM}[r][j]$, and repeat the process from there until we get to a leaf. We then return an arbitrary item contained in the corresponding leaf bucket. In the full version of the paper we analyze this process and show that the rank of the returned item is close enough to k.

To satisfy the *fairness* condition mentioned above, two minor changes are needed: (1) the insertion buffer should be emptied once for every Δ updates (not just every Δ insertions); (2) the base sets are maintained as queues (FIFO).

It is possible to convert the amortized time bounds into worst-case time bounds using the techniques of Andersson and Thorup [2].

2.2 Dynamic sets with approximate SELECT and RANK

Pătraşcu and Thorup [20] devised a data structure that supports INSERT, DELETE and *exact* SELECT and RANK operations in $O(\log n/\log w)$ time. An interesting separation between SELECT and RANK operations emerges when approximate results are allowed. In this section we explain how the data structure of Section 2.1 can be extended to support Δ-RANK operations. While the time of INSERT, DELETE and Δ-SELECT operations remains unchanged, i.e., $O(\log n/\log(w\Delta))$, the time of Δ-RANK operations is $O\left(\log n/\log(w\Delta) + pred(\frac{n}{\Delta}, t_u\Delta, n, w)\right)$ time, where $pred(n,t,s,w)$ is the time for dynamically answering *predecessor queries* on a set of n items when the update is $O(t)$, the space used by the data structure is $O(s)$, and the word length is w. We later give a lower bound that shows that appearance of the $pred(\frac{n}{\Delta}, t_u\Delta, n, w)$ term here cannot be avoided.

Quite a lot is known about $pred(n,t,s,w)$, the time for answering exact PREDECESSOR queries. Beame and Fich [3] gave $\Omega\left(\sqrt{\frac{\log n}{\log\log n}}\right)$ and $\Omega\left(\frac{\log w}{\log\log w}\right)$ lower bounds for the static version of the problem, where polynomial time preprocessing and polynomial space are allowed. (Static means no updates.) Pătraşcu and Thorup [19] obtained a complete query-space tradeoff, when again no updates are allowed, i.e., they determined $pred(n,\infty,s,w)$ asymptotically for all s and w. Pătraşcu and Thorup [19] showed that $pred\left(n, \frac{\log n}{\log w}, n, w\right) = \Theta\left(\frac{\log n}{\log w}\right)$ and that $pred(w,1,w,w) = O(1)$. Further complications arise when deterministic vs. randomized variants of the problem are considered. We refer the reader to [19, 20] for the exact details. The picture simplifies considerably when $pred(n,t,s) = \max_w pred(n,t,s,w)$ is considered. Here, it is known that $pred(n, \sqrt{\frac{\log n}{\log\log n}}, n, w) = pred(n, n^{O(1)}, n^{O(1)}) = \Theta\left(\sqrt{\frac{\log n}{\log\log n}}\right)$.

As in Section 2.1, we may assume that $\Delta > w^3$, as the case $\Delta \leq w^3$ is covered by the exact algorithm of Pătraşcu and Thorup [20], and aim for $t_u = O(\log n/\log\Delta)$ and $O\left(\log n/\log\Delta + pred(\frac{n}{\Delta}, t_u\Delta, n, w)\right)$ bounds, respectively.

It is not difficult to see that the rank of an item appearing in a leaf bucket or as a splitter in the data structure of Section 2.1 can be easily approximated in $O(\log n/\log\Delta)$ time. A Δ-RANK operation, however, must also be able to return the (approximate) rank of an item that does not appear in the data structure. To achieve that, we maintain an exact dynamic predecessor data structure on the splitters. Given an item not in the data structure we

do a PREDECESSOR query on the key of the item, and return the approximate rank of the returned splitter. As there are only $O(\frac{n}{\Delta})$ splitters, the total time of a Δ-RANK operation is $O\left(\log n/\log\Delta + pred(\frac{n}{\Delta}, t_u\Delta, n, w)\right)$, as required. As we need to update the predecessor data structure, on average, only once in every Δ operations, we can afford to spend $t_u\Delta$ time on updates to the predecessor data structure. In Section 3, we show that the use of a predecessor data structure is essential.

2.3 Equivalence between approximate heaps and partitioning

The MIN operation is a very special SELECT operation in which the smallest, i.e., the item of rank 0, is sought after. Thus, any data structure that supports approximate SELECT operations also support approximate MIN operations. However, MIN operations could potentially be faster than general SELECT operations. This is true, for example, for the exact versions of these problems on the word RAM.

A data structure that supports INSERT, DELETE and MIN operations is a *priority queue*. A Δ-heap is a data structure that supports Δ-MIN operations which return one of the smallest Δ items in the data structure. (For $\Delta = 1$ we get an exact priority queue.)

Thorup [21] obtained an equivalence between priority queues and sorting. Namely, there is an (exact) priority queue that supports each operation in $P(n)$ time, where $P(n)$ is non-decreasing in n, if and only if it is possible, for every n, to sort n items in $O(n\,P(n))$ time. Here we extend this result to obtain an equivalence between Δ-heaps and the *ordered* Δ*-partition* problem: Given n items, partition them into $k \approx \frac{n}{\Delta}$ sets $A_1, A_2, \ldots, A_k$, each of size about Δ, such that $A_i < A_{i+1}$, i.e., all items in A_i are smaller than all items in A_{i+1}, for $i = 0, 1, \ldots, k-1$. We show that there is a Δ-heap with $O(P_\Delta(n))$ time per operation, where $P_\Delta(n)$ is non-decreasing in n, if and only if it is possible to Δ-partition a set of n items in $O(nP_\Delta(n))$ time.

Han and Thorup [15] showed that the ordered Δ-partition problem is equivalent to the problem of partitioning a set A of size n according to $k \approx \frac{n}{\Delta}$ splitters $s_1 < s_2 < \ldots < s_k$, producing sets $A_0, A_1, \ldots, A_k$ such that $s_i \le A_i < s_{i+1}$, for $i = 0, 1, \ldots, k$, where $s_0 = -\infty$ and $s_{k+1} = +\infty$. Note that in this variant the sets $A_0, A_1, \ldots, A_k$ are *not* necessarily of (roughly) equal size. This is the version of the ordered Δ-partition problem that we use in this section.

We sketch here the construction of a $O(\Delta \log n)$-approximate heap using an algorithm for the ordered Δ-partition problem, which is the interesting direction of the equivalence. In the full version of the paper we complete the details of this construction, remove the $O(\log n)$ from the approximation factor, and prove the other direction of the equivalence.

Most of the items in the priority queue constructed are stored in *buckets* $A_0, A_1, \ldots, A_k$ separated by $k+2$ *base splitters* $-\infty = s_0 < s_1 < s_2 < \ldots < s_k < s_{k+1} = +\infty$. Thus, $s_0 \le A_0 < s_1 \le A_1 < \ldots < s_k \le A_k < s_{k+1}$. The splitters are copies of keys of items that belong or belonged to the priority queue at some stage. The items within each bucket are unsorted. We let $\Phi = \Delta \log n$ and require that $\frac{\Phi}{4} \le |A_i| \le \Phi$, for $i = 0, 1, \ldots, k-1$, and $|A_k| \le \Phi$.

A logarithmic number of splitters $t_0 < t_1 < \ldots < t_{\ell+1}$, where $\ell = \Theta(\log n)$, also serve as *level splitters*. We have $t_0 = s_0 = -\infty$ and $t_{\ell+1} = s_{k+1} = +\infty$. Each level splitter t_j has a *level buffer* B_j associated with it that can hold up to $4^j\Delta$ items. We also maintain a counter q_j that provides an upper bound of the size of B_j. Finally, there is also an *insertion buffer* $B = B_0$ that can hold up to Φ items. We maintain the following invariants, for $j = 1, 2, \ldots \ell$:

(i) There are more than $\frac{1}{2}4^j \cdot \Phi$ bucket items smaller than t_j, if $t_j < \infty$.
(ii) There are less than $\frac{7}{4}4^j \cdot \Phi$ items smaller than t_j.
(iii) The keys of all items in B_j are in $[t_j, t_{j+2})$. Here $t_{\ell+1} = t_{\ell+2} = \infty$.

To insert an item, we simply add it to the insertion buffer. If the insertion buffer is not full, this completes the operation. To delete an item, given a pointer to it, we simply delete it from its bucket or buffer, and update some counters. A MIN operation returns an arbitrary item from the bucket A_0, referred to as the *head*. The rank of the returned item is at most $2\Phi = 2\Delta \log n$, as only items in the head and the insertion buffer may be smaller than the returned item.

When the insertion buffer B is full, i.e., it contains $\Phi = \Delta \log n$ items, we use the ordered Δ-partition algorithm to split the items in B according to the $O(\log n)$ level splitters. The amortized cost per item is $O(P_\Delta(\Delta \log n)) = O(P_\Delta(n))$. Recall that $O(P_\Delta(n))$ is the time, per item, needed to order partition n items into sets of size roughly Δ, or equivalently, to Δ-sort n items. (We may assume that $\Delta \leq n^{1/2}$, as $\Delta = n^{1/2}$ already yields a constant time per operation, and hence $\Delta \log n \leq n$.) Let $B^1 < B^2 < \ldots < B^\ell$ be the resulting partition. The items in B^j are added, one by one, into the level buffer B_j.

When a level buffer B_j is full, or when we need to change the level splitter t_j, we split its items according to all base splitters that are smaller than t_{j+2}. As the number of items smaller than t_{j+2} is $O(4^j\Phi)$, and as each bucket is of size $\Theta(\Phi)$, the number of splitters smaller than t_{j+2} is $O(4^j)$. The number of items in the buffer is at most $4^j\Delta$. Thus, the splitting can be done in $O(4^j\Delta \cdot P_\Delta(n))$ time. The items in each set of the partition generated are added, one by one, to the appropriate buckets.

The above "idyllic" description ignored the fact that base buckets may get too large or too small, and that the three invariants imposed on the level splitters may be violated. However, as we allowed enough slack in the size of the base buckets and in the invariants, it is not too difficult to fix these problems by merging and splitting adjacent base buckets, and by periodically redefining the level splitters. The fairly technical details are given in the full version of the paper. The amortized time per operation remains $O(P_\Delta(n))$.

2.4 Equivalence between soft heaps and sorting

A q-soft heap is a heap which is allowed to corrupt, i.e., increase the keys, of a small fraction of its items. More specifically, after n INSERT operations, at most n/q of the items *in* the heap are allowed to be corrupt. The hope, of course, is that allowing corruptions leads to a more efficient implementation of the heap operations. (It is important to note that the number of corrupt items is related to the number of insertions, not to the number of items currently in the heap.)

It follows easily from the definition that as long as less than q items are inserted into a q-soft heap, the heap is not allowed to corrupt any item. Thus, a q-soft heap with $O(t)$ time per operation can be used to exactly sort $q-1$ items in $O(qt)$ time. By Thorup's [21] equivalence between sorting and priority queues, we get that if there is q-soft heap with $O(t)$ time per operation, then there is also an *exact* heap that can hold up to q items with $O(t)$ time per operation. It is also easy, to obtain this result directly. To implement an exact heap on at most q items, we use a $2q$-soft heap. After every $2q$ insertions we rebuild the $2q$-soft heap using at most q insertions. The amortized time per operation stays $O(t)$.

More surprising is that we also have the opposite implication. If there is an exact heap on at most q items with $O(t)$ time per operation, then there is also a q-soft heap with $O(t)$ time per operation. (Note that the number of items in the q-soft heap is not bounded.) To

prove this result we use the simplified construction of soft heaps by Kaplan et al. [17]. We observe that certain components in this comparison-based implementation of soft heaps can be replaced by exact heaps that hold up to q items. This also leads to a clearer understanding of how soft heaps work.

2.5 Faster exact sorting

Han and Thorup [15] obtained a deterministic linear time algorithm for partitioning n items into a sequence of $\sqrt{n}$ sets, each of size roughly $\sqrt{n}$, such that the items in each set are smaller than the items in the next set. This leads, by repeated partitioning, to a deterministic $O(n \log\log n)$-time algorithm. A different deterministic $O(n \log\log n)$-time sorting algorithm was earlier obtained by Han [14].

More recently, Pătraşcu and Thorup [20], improving results of Fredman and Willard [13], implemented *dynamic fusion trees*, supporting INSERT, DELETE, RANK and SELECT in $O(\log_w n)$ time per operation, where n is the size of the set and w is the word length. This leads immediately to an $O(n \log_w n)$-time sorting algorithm. This algorithm is faster than the $O(n \log\log n)$-time algorithm for sufficiently large w, e.g., $w = n^{\omega(1/\log\log n)}$.

A simple combination of these two algorithms gives rise to a deterministic $O(n \log\log_w n)$-time algorithm. This bound subsumes the two previous bounds. Perform $\log\log_w n$ partitioning steps of [15] that partition the n input items into sets of size w. These sets are then sorted in linear time using the dynamic fusion trees of [20]. The $O(n \log\log_w n)$-time algorithm is asymptotically faster than the $O(n \log\log n)$-time algorithm for much smaller values of w, namely $w = n^{1/\log^{o(1)} n}$.

Han and Thorup [15] obtain a faster randomized $O(n\sqrt{\log\log n})$-time sorting algorithm. Andersson et al. [1] and Belazzougui et al. [4] have shown that sorting can be done in linear expected time when $w = \Omega(\log^2 n \log\log n)$.

It is an open problem whether randomization can speed up partitioning algorithms and lead, in particular, to faster data structures for the various operations considered in this paper, such as Δ-SELECT.

3 Lower bounds

In this section we give cell-probe lower bounds that match the upper bounds obtained by the data structures for approximate SELECT and RANK given in Sections 2.1 and 2.2.

3.1 Lower bound for approximate SELECT

Pătraşcu and Thorup [20], relying on previous results of Fredman and Sacks [12], proved the following cell-probe lower bound.

▶ **Theorem 3.1.** *For any cell-probe data structure that supports* INSERT, DELETE *and (exact)* SELECT *(or* RANK*) operations, if both* INSERT *and* DELETE *require $t = t(n)$ time per operation, then* SELECT *(or* RANK*) operations must take $\Omega\left(\frac{\log n}{\log(w \cdot t(n))}\right)$ time.*

Relying on this result, using a simple reduction, we obtain our lower bound for Δ-SELECT.

▶ **Theorem 3.2.** *For any cell-probe data structure that supports* INSERT, DELETE *and* Δ-SELECT *(or* Δ-RANK*) operations, if both* INSERT *and* DELETE *require at most $O(\log n)$ time per operation, then* SELECT *(or* RANK*) operations must take $\Omega\left(\frac{\log n}{\log(w\Delta)}\right)$ time.*

Proof. Given a Δ-approximate data structure, we can construct an exact data structure by *duplicating* each item 2Δ times. More specifically, when an item is to be inserted into the exact data structure, we insert 2Δ copies of the item into the Δ-approximate data structure. When an item is to be deleted from the exact data structure, we delete its 2Δ copies from the Δ-approximate data structure. To select the item of rank i, we Δ-approximately select an item of rank $2\Delta i + \Delta$. As the rank of the item returned differs from $2\Delta i + \Delta$ by *less* than Δ, the item returned must be a copy of the item of rank i. To compute the rank of an item, not necessarily in the data structure, we do a Δ-RANK query on the approximate data structure, divide the returned rank by 2Δ and round down. It is again easy to see that this is the exact rank of the queried item.

If the times of INSERT and DELETE of the Δ-approximate data structures are $t(n)$, then the times of INSERT and DELETE in the exact data structure are $\Delta t(\Delta n)$. If $s(n)$ is the time of Δ-SELECT (or RANK), then the time for exact SELECT (or RANK) is $s(\Delta n)$. If follows from Theorem 3.1 that $s(\Delta n) = \Omega\left(\frac{\log n}{\log(w \cdot \Delta t(\Delta n))}\right)$, or equivalently $s(n) = \Omega\left(\frac{\log \frac{n}{\Delta}}{\log(w \cdot \Delta \cdot t(n))}\right)$. If $t(n) = O(\log n)$, then as $w \geq \log n$, we also get that $t(n) = O(w)$. We may also assume that $\Delta < n^{1/2}$, as otherwise, the lower bound is a constant. Thus, $s(n) = \Omega\left(\frac{\log n}{\log(w \cdot \Delta)}\right)$, as claimed. ◀

The lower bound for Δ-SELECT is tight, as it matches the upper bound of the data structure given in Section 2.2.

3.2 Lower bound for approximate RANK

The lower bound of Theorem 3.2 holds also for Δ-RANK, but it is not tight for all values of Δ. We provide here a tight lower bound that matches the performance of the data structure given in Section 2.2. The lower bound relies on the fact an exact PREDECESSOR operation can be performed using one SELECT and one RANK operations, namely, PREDECESSOR(k) = SELECT(RANK(k)).

▶ **Theorem 3.3.** *For any linear space cell-probe data structure that supports* INSERT, DELETE *and* Δ-RANK *operations, if* INSERT *and* DELETE *take at most* $t_u = O\left(\log n / \log(w\Delta)\right)$ *time per operation, then* Δ-RANK *operations must take* $\Omega\left(\log n / \log(w\Delta) + pred(\frac{n}{\Delta}, t_u\Delta, n, w)\right)$ *time.*

Proof. Suppose that we are given a linear space data structure that supports INSERT and DELETE operations in $t_u = O((\log n / \log(w\Delta))$ time, and Δ-RANK operations in $r(n)$ time. We already known, by Theorem 3.2, that $r(n) = \Omega\left(\log n / \log(w\Delta)\right)$ time. Thus, we only need to show that $r(n) = \Omega\left(pred(\frac{n}{\Delta}, t_u\Delta, n, w)\right)$. We show that we can use the approximate data structure given to obtain a data structure that supports exact PREDECESSOR operations in $O(r(\Delta n))$. It would then follow that $r(n) = \Omega(pred(\frac{n}{\Delta}, t_u\Delta, n, w))$.

We first add to the data structure given to us the ability to support Δ-SELECT operations. This is easily done by using *both* the given data structure *and* our Δ-SELECT data structure of Section 2.1 to hold the same set of items. INSERT and DELETE operations still take $t_u = O((\log n / \log(w\Delta))$ time, and Δ-SELECT and RANK operations still take $r(n)$ time, as we already know that $r(n) = \Omega\left(\log n / \log(w\Delta)\right)$.

We now use the duplication technique used in the proof of Theorem 3.2. We get an exact data structure in which INSERT and DELETE take $\Delta t(\Delta n)$ time, and exact SELECT and RANK, and hence exact PREDECESSOR, take $r(\Delta n)$ time. Thus, $r(n) = \Omega\left(pred(\frac{n}{\Delta}, t_u\Delta, n, w)\right)$, as claimed. ◀

4 Concluding remarks and open problems

Data structures for supporting dynamic ordered sets, i.e., data structures that support INSERT, DELETE, and either one or both of SELECT and RANK, and possibly some other operations, are among the most basic and natural data structures. Pătraşcu and Thorup [20] obtained essentially optimal implementation of such data structures in the word RAM model, the model that most closely represents what can be done on a real computer.

Surprisingly, not much attention was paid before to data structures that support *approximate* versions of these operations. There are many conceivable applications in which, for example, we do not insist on knowing the exact rank of an item in the set, a good enough approximation might be enough.

We show that allowing approximation greatly speeds up some of these operations, while the complexity of the other operations remains essentially unchanged. We obtain a full characterization of the time needed to implement approximate versions of these operations. A fairly interesting picture emerges. While the exact versions of SELECT and RANK have the same complexity, a separation emerges between the approximate versions of these problems.

We also considered approximate MIN operations that correspond to the implementation of approximate heaps (priority queues). It is known that exact MIN operations are easier than the more general SELECT operations. We show that the "exponential" gap between these operations persists when approximation is allowed. We obtained a equivalence between approximate heaps and approximate sorting, extending the equivalence of Thorup [21] for the exact versions of these problems.

Closely related to approximate heaps are Chazelle's [6] soft heaps that feature prominently in his deterministic minimum spanning tree algorithm [5]. The exact relation between approximate and soft heaps was not understood before. Looking at these two data structures using the "word RAM lens" reveals an essential difference between these two data structures. Approximate heaps correspond to approximate sorting, or partitioning, while q-soft heaps actually correspond to the *exact* sorting of sets of size q. This might explain the additional usefulness of soft heaps.

The closer look at ordered set data structures, partitioning algorithms and sorting algorithms also revealed, as a byproduct, a new deterministic sorting algorithm that runs in $O(n \log \log_w n)$ time. The new bound subsumes and improves on the two previously known bounds of $O(n \log \log n)$ and $O(n \log_w n)$.

We focused in this paper on deterministic algorithm. Studying the effect of randomization on the various problems considered is an interesting research topic. In particular, it would be interesting to know whether randomization can speed up (ordered) partitioning algorithms.

References

1 Arne Andersson, Torben Hagerup, Stefan Nilsson, and Rajeev Raman. Sorting in Linear Time? *J. Comput. Syst. Sci.*, 57(1):74–93, 1998. Announced at STOC'95. doi:10.1006/jcss.1998.1580.

2 Arne Andersson and Mikkel Thorup. Dynamic Ordered Sets with Exponential Search Trees. *J. ACM*, 54(3):Article 13, 2007. Combines results announed at FOCS'96, STOC'00, and SODA'01.

3 Paul Beame and Faith Fich. Optimal Bounds for the Predecessor Problem and Related Problems. *J. Comput. System Sci.*, 65(1):38–72, 2002. Announced at STOC'99.

4 Djamal Belazzougui, Gerth Stølting Brodal, and Jesper Sindahl Nielsen. Expected Linear Time Sorting for Word Size $\Omega(\log^2 n \log \log n)$. In *Proc. 14th SWAT*, pages 26–37, 2014. doi:10.1007/978-3-319-08404-6_3.

5 Bernard Chazelle. A minimum spanning tree algorithm with Inverse-Ackermann type complexity. *J. ACM*, 47(6):1028–1047, 2000. doi:10.1145/355541.355562.

6 Bernard Chazelle. The soft heap: an approximate priority queue with optimal error rate. *J. ACM*, 47(6):1012–1027, 2000. doi:10.1145/355541.355554.

7 L. J. Comrie. The Hollerith and Powers Tabulating Machines. *Trans. Office Machinary Users' Assoc., Ltd*, pages 25–37, 1929-30.

8 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Cliff Stein. *Introduction to algorithms*. The MIT Press, 3rd edition, 2009.

9 A. I. Dumey. Indexing for rapid random access memory systems. *Computers and Automation*, 5(12):6–9, 1956.

10 Adrian Dumitrescu. A Selectable Sloppy Heap. *CoRR*, abs/1607.07673, 2016. arXiv:1607.07673.

11 Michael L. Fredman. Comments on Dumitrescu's "A Selectable Sloppy Heap". *CoRR*, abs/1610.02953, 2016. arXiv:1610.02953.

12 Michael L. Fredman and Michael E. Saks. The Cell Probe Complexity of Dynamic Data Structures. In *Proc.* 21^{st} *STOC*, pages 345–354, 1989.

13 Michael L. Fredman and Dan E. Willard. Surpassing the information theoretic bound with fusion trees. *J. Comput. Syst. Sci.*, 47:424–436, 1993. Announced at STOC'90.

14 Yijie Han. Deterministic Sorting in $O(n \log\log n)$ Time and Linear Space. *J. Algorithms*, 50(1):95–105, 2004. Announced at STOC'02.

15 Yijie Han and Mikkel Thorup. Integer Sorting in $O(n\sqrt{\log\log n})$ Expected Time and Linear Space. In *Proc.* 43^{rd} *FOCS*, pages 135–144, 2002.

16 Haim Kaplan, László Kozma, Or Zamir, and Uri Zwick. Selection from Heaps, Row-Sorted Matrices, and X+Y Using Soft Heaps. In *2nd Symposium on Simplicity in Algorithms, SOSA@SODA 2019, January 8-9, 2019 - San Diego, CA, USA*, pages 5:1–5:21, 2019. doi:10.4230/OASIcs.SOSA.2019.5.

17 Haim Kaplan, Robert Endre Tarjan, and Uri Zwick. Soft Heaps Simplified. *SIAM J. Comput.*, 42(4):1660–1673, 2013. doi:10.1137/120880185.

18 B.W. Kernighan and D.M. Ritchie. *The C Programming Language*. Prentice Hall, 1978.

19 Mihai Pătraşcu and Mikkel Thorup. Time-space trade-offs for predecessor search. In *Proc.* 38^{th} *STOC*, pages 232–240, 2006. doi:10.1145/1132516.1132551.

20 Mihai Pătraşcu and Mikkel Thorup. Dynamic Integer Sets with Optimal Rank, Select, and Predecessor Search. In *Proc.* 55^{th} *FOCS*, pages 166–175, 2014. doi:10.1109/FOCS.2014.26.

21 Mikkel Thorup. Equivalence between priority queues and sorting. *J. ACM*, 54(6):28, 2007. Announced at FOCS'02. doi:10.1145/1314690.1314692.

Amplification with One NP Oracle Query

Thomas Watson
University of Memphis, Memphis, TN, USA
Thomas.Watson@memphis.edu

Abstract

We provide a complete picture of the extent to which amplification of success probability is possible for randomized algorithms having access to one NP oracle query, in the settings of two-sided, one-sided, and zero-sided error. We generalize this picture to amplifying one-query algorithms with q-query algorithms, and we show our inclusions are tight for relativizing techniques.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases Amplification, NP, oracle, query

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.96

Category Track A: Algorithms, Complexity and Games

Related Version The full version of the paper is available at `https://eccc.weizmann.ac.il/report/2018/058/`.

Funding Supported by NSF grant CCF-1657377.

1 Introduction

Amplification of the success probability of randomized *algorithms* is a ubiquitous tool in complexity theory. We investigate amplification for randomized *reductions* to NP-complete problems, which can be modeled as randomized algorithms with the ability to make queries to an NP oracle. The usual amplification strategy involves running multiple independent trials, which would also increase the number of NP oracle queries, so this does not generally work if we restrict the number of queries. We study, and essentially completely answer, the following question:

> *If a language is solvable with success probability p by a randomized polynomial-time algorithm with access to one* NP *oracle query, what is the highest success probability achievable with one query (or $q > 1$ many queries) to an* NP *oracle?*

The question makes sense for two-sided error ($\mathsf{BPP}^{\mathsf{NP}[1]}$), one-sided error ($\mathsf{RP}^{\mathsf{NP}[1]}$), and zero-sided error ($\mathsf{ZPP}^{\mathsf{NP}[1]}$), and it was mentioned in [2] as "an interesting problem worthy of further investigation." Partial results for zero-sided error were shown in [3]. The question is also relevant to the extensive literature on bounded NP queries (the boolean hierarchy); e.g., $\mathsf{ZPP}^{\mathsf{NP}[1]}$ shows up frequently in the context of the "two queries problem" [4], which was the main application area of the results from [3].

Our first contribution characterizes the best amplification achievable by relativizing techniques in the two-sided error setting. In general, the best strategy for amplifying plain randomized algorithms is to take the majority vote of q independent trials, which in our setting would naively involve q NP oracle queries. One may suspect this majority vote strategy is optimal for us. We show this intuition is a red herring; it is possible to do better by "combining" NP oracle queries across different trials. As an extreme example, consider the special case of randomized *mapping* reductions to NP problems. These are equivalent to Arthur–Merlin games (AM), for which amplification is possible by running independent

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 96; pp. 96:1–96:13


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

trials and simply having Merlin's message consist of certificates for a majority of the trials. However, if we allow one NP oracle query, but do not necessarily output the same bit the oracle returns, then combining queries is less straightforward, and it turns out amplification is only possible to a limited extent.

Our main take-home message is that starting with success probability greater than $\frac{1}{2} + \frac{1}{2} \cdot \frac{1}{k+1}$, where k is an integer, we can get arbitrarily close to $\frac{1}{2} + \frac{1}{2} \cdot \frac{1}{k}$ success probability while still using one NP query; using q nonadaptive queries, roughly a factor q improvement over this is possible.

We give precise definitions in Section 2, but we now clarify our notation before stating the theorem. For $\epsilon \in (0, 1]$ (the *advantage*), $\mathsf{BPP}_{\epsilon}^{\mathsf{NP}[1]}$ is the set of all languages solvable by a randomized polynomial-time algorithm that may make one query to an NP oracle and produces the correct output with probability $\geq \frac{1}{2} + \frac{1}{2}\epsilon$ on each input. For convenience, we define $\mathsf{BPP}_{>\epsilon}^{\mathsf{NP}[1]}$ by requiring that for some constant c there exists such an algorithm with advantage $\geq \epsilon + n^{-c}$, and we define $\mathsf{BPP}_{\epsilon>}^{\mathsf{NP}[1]}$ by requiring that for every constant d there exists such an algorithm with advantage $\geq \epsilon - 2^{-n^d}$; the reason for these conventions is just that they naturally arise in the proofs (e.g., standard majority amplification implies $\mathsf{BPP}_{>0} = \mathsf{BPP}_{1>}$). We make similar definitions for $\mathsf{BPP}^{\mathsf{NP}\|[q]}$ but allowing q nonadaptive NP oracle queries. Allowing q adaptive NP queries is equivalent to allowing $2^q - 1$ nonadaptive NP queries [1].

▶ **Theorem 1** (Two-sided error). *For integers $1 \leq q \leq k$:*

- *If q is odd:*

$$\mathsf{BPP}_{>1/(k+1)}^{\mathsf{NP}[1]} \subseteq \mathsf{BPP}_{q/k>}^{\mathsf{NP}\|[q]} \quad \textit{and} \quad \mathsf{BPP}_{1/k}^{\mathsf{NP}[1]} \not\subseteq \mathsf{BPP}_{>q/k}^{\mathsf{NP}\|[q]} \textit{ relative to an oracle.}$$

- *If q, k are even:*

$$\mathsf{BPP}_{>1/(k+1)}^{\mathsf{NP}[1]} \subseteq \mathsf{BPP}_{q/k>}^{\mathsf{NP}\|[q]} \quad \textit{and} \quad \mathsf{BPP}_{1/(k-1)}^{\mathsf{NP}[1]} \not\subseteq \mathsf{BPP}_{>q/k}^{\mathsf{NP}\|[q]} \textit{ relative to an oracle.}$$

The word "oracle" has two meanings here. Besides the bounded NP oracle queries of central interest, "relative to an oracle" means there exists a language such that the separation holds when all computations (the randomized algorithm and the NP verifier) can make polynomially many adaptive queries to an oracle for that language. In particular, in the context of our relativized separations, randomized algorithms have access to two oracles. The separations in Theorem 1 are tight since the inclusions relativize. This implies that using "black-box simulation" techniques, it is not possible to significantly improve any of our inclusions.

If we start with advantage $> \frac{1}{k+1}$ where k is an integer, Theorem 1 tells us the best advantage achievable with q nonadaptive NP queries using relativizing techniques: if k is even we can amplify to essentially $\frac{q}{k}$; if k is odd we can amplify to essentially $\frac{q}{k}$ if q is odd, and $\frac{q}{k+1}$ if q is even. (Theorem 1 does not explicitly mention the case where q is even and k is odd, but in this case the best inclusion and separation are obtained by applying the theorem to the even integer $k + 1$.)

A subtle issue is whether "$q/k>$" in the inclusion subscripts can be improved to "q/k"; e.g., it remains open to show that $\mathsf{BPP}_{>1/3}^{\mathsf{NP}[1]} \subseteq \mathsf{BPP}_{1/2}^{\mathsf{NP}[1]}$ or that $\mathsf{BPP}_{>1/3}^{\mathsf{NP}[1]} \not\subseteq \mathsf{BPP}_{1/2}^{\mathsf{NP}[1]}$ relative to an oracle.

The proof of Theorem 1 appears in Section 3. No such nontrivial inclusion was known before; for relativized separations, the case $q = 1$, $k = 2$ was shown in [7].

Zero-sided error algorithms must output the correct bit with probability at least some $\epsilon \in (0, 1]$ and output $\perp$ (plead ignorance) with the remaining probability. We define the advantage (the subscript of $\mathsf{ZPP}^{\mathsf{NP}\|[q]}$) to be this ϵ.

[3] proved that $\mathsf{ZPP}^{\mathsf{NP}[1]}_{>0} \subseteq \mathsf{ZPP}^{\mathsf{NP}[1]}_{1/4}$ and $\mathsf{ZPP}^{\mathsf{NP}[1]}_{>1/2} \subseteq \mathsf{ZPP}^{\mathsf{NP}[1]}_{1>}$,[1] and left it unresolved what happens between advantages $\frac{1}{4}$ and $\frac{1}{2}$. We settle this decade-old open problem: amplification is possible between $\frac{1}{4}$ and $\frac{1}{3}$ and between $\frac{1}{3}$ and $\frac{1}{2}$.

▶ **Theorem 2** (Zero-sided error). *For integers $1 \leq q \leq k \leq 4$:*

- *If $k = 4$:* $\mathsf{ZPP}^{\mathsf{NP}[1]}_{>0} \subseteq \mathsf{ZPP}^{\mathsf{NP}\|[q]}_{q/k>}$.
- *If $k \leq 3$:* $\mathsf{ZPP}^{\mathsf{NP}[1]}_{>1/(k+1)} \subseteq \mathsf{ZPP}^{\mathsf{NP}\|[q]}_{q/k>}$.
- *If $q = 1$:* $\mathsf{ZPP}^{\mathsf{NP}[1]}_{1/k} \not\subseteq \mathsf{ZPP}^{\mathsf{NP}\|[q]}_{>q/k}$ *relative to an oracle.*

Moreover, the "$q/k>$" in the inclusion subscripts can be improved to "q/k" if $q < k \geq 3$.

The proof of Theorem 2 appears in Section 4. The "moreover" part uses a trick described in [3] for getting a tiny boost in the advantage. Like the situation with $\mathsf{BPP}^{\mathsf{NP}[1]}$, it remains open to show that $\mathsf{ZPP}^{\mathsf{NP}[1]}_{>1/3} \subseteq \mathsf{ZPP}^{\mathsf{NP}[1]}_{1/2}$ or that $\mathsf{ZPP}^{\mathsf{NP}[1]}_{>1/3} \not\subseteq \mathsf{ZPP}^{\mathsf{NP}[1]}_{1/2}$ relative to an oracle. There is no reason to consider $k > 4$ in Theorem 2, since then $\mathsf{ZPP}^{\mathsf{NP}[1]}_{>1/(k+1)} \subseteq \mathsf{ZPP}^{\mathsf{NP}[1]}_{>0} \subseteq \mathsf{ZPP}^{\mathsf{NP}\|[q]}_{q/4>}$.

We conjecture that the third bullet in Theorem 2 also holds for $q > 1$ (i.e., the relativized separations $\mathsf{ZPP}^{\mathsf{NP}[1]}_{1/4} \not\subseteq \mathsf{ZPP}^{\mathsf{NP}\|[2]}_{>2/4}$ and $\mathsf{ZPP}^{\mathsf{NP}[1]}_{1/4} \not\subseteq \mathsf{ZPP}^{\mathsf{NP}\|[3]}_{>3/4}$ and $\mathsf{ZPP}^{\mathsf{NP}[1]}_{1/3} \not\subseteq \mathsf{ZPP}^{\mathsf{NP}\|[2]}_{>2/3}$). This remains open, though we are aware of how to prove that $\mathsf{ZPP}^{\mathsf{NP}[1]}_{1/4} \not\subseteq \mathsf{ZPP}^{\mathsf{NP}\|[2]}_{>3/4}$. Anyway, $q = 1$ is the most natural case, and we provide a complete proof for it.

One-sided error algorithms must always output 0 if the answer is 0, and must output 1 with probability at least some $\epsilon \in (0, 1]$ if the answer is 1. We define the advantage (the subscript of $\mathsf{RP}^{\mathsf{NP}\|[q]}$) to be this ϵ. The proof of Theorem 3 appears in the full version of this paper [6] and is relatively straightforward.

▶ **Theorem 3** (One-sided error).

- $\mathsf{RP}^{\mathsf{NP}[1]}_{>1/2} \subseteq \mathsf{RP}^{\mathsf{NP}[1]}_{1>}$.
- $\mathsf{RP}^{\mathsf{NP}[1]}_{>0} \subseteq \mathsf{RP}^{\mathsf{NP}[1]}_{1/2} \cap \mathsf{RP}^{\mathsf{NP}\|[2]}_{1>}$ *and* $\mathsf{RP}^{\mathsf{NP}[1]}_{1/2} \not\subseteq \mathsf{RP}^{\mathsf{NP}[1]}_{>1/2}$ *relative to an oracle.*

Finally, we point out that none of the inclusions in this paper can be strengthened to yield advantage exactly 1 via relativizing techniques, since $\mathsf{BPP} \subseteq \mathsf{ZPP}^{\mathsf{NP}[1]}_{>1/2}$ relativizes [2] but $\mathsf{BPP} \not\subseteq \mathsf{P}^{\mathsf{NP}}$ relative to an oracle [folklore].

2 Definitions

We formally define the relevant complexity classes in Section 2.1 and their decision tree analogues (which are used for relativized separations) in Section 2.2.

2.1 Time complexity

We think of a randomized algorithm M as taking a uniformly random string $s \in \{0,1\}^r$ (for some number of coins r that depends on the input length); we let $M_s(x)$ denote M running on input x with outcome s.

For $\epsilon \in (0, 1]$ (the *advantage*) and integer $q \geq 1$, language L is in $\mathsf{BPP}^{\mathsf{NP}\|[q]}_{\epsilon}$ iff there is a polynomial-time randomized algorithm M (taking input x and coin tosses $s \in \{0,1\}^r$) and a language $L' \in \mathsf{NP}$ such that the following hold.

[1] [7] gave an alternative proof of the latter but with only $1 - \frac{1}{\text{poly}}$, rather than $1 - \frac{1}{\exp}$, success probability.

Syntax: The computation of $M_s(x)$ produces a tuple of query strings $(z_1, \ldots, z_q)$ and a truth table $out\colon \{0,1\}^q \to \{0,1\}$; the output is then $out(L'(z_1), \ldots, L'(z_q))$.
Correctness: The output is $L(x)$ with probability $\geq \frac{1}{2} + \frac{1}{2}\epsilon$.

$\mathsf{RP}^{\mathsf{NP}\|[q]}_{\epsilon}$ is defined similarly except for correctness, we require the output is always 0 if $L(x) = 0$, and is 1 with probability $\geq \epsilon$ if $L(x) = 1$. $\mathsf{ZPP}^{\mathsf{NP}\|[q]}_{\epsilon}$ is defined similarly except $out\colon \{0,1\}^q \to \{0,1,\perp\}$ and for correctness, we require the output is always $L(x)$ or $\perp$, and is $L(x)$ with probability $\geq \epsilon$.

For $\mathcal{C} \in \left\{\mathsf{BPP}^{\mathsf{NP}\|[q]}, \mathsf{RP}^{\mathsf{NP}\|[q]}, \mathsf{ZPP}^{\mathsf{NP}\|[q]}\right\}$, we define

$$\mathcal{C}_{>\epsilon} = \bigcup_{\text{constants } c} \mathcal{C}_{\epsilon+n^{-c}} \quad \text{and} \quad \mathcal{C}_{\epsilon>} = \bigcap_{\text{constants } d} \mathcal{C}_{\epsilon-2^{-n^d}}.$$

When $q = 1$ we may drop the $\|$ from the superscripts.

2.2 Decision tree complexity

We think of a randomized decision tree T as the uniform distribution over a multiset of corresponding deterministic decision trees T_s indexed by $s \in \{0,1\}^r$; we denote this as $T \sim \left\{T_s : s \in \{0,1\}^r\right\}$. In this setting, "query" actually has two meanings for us: a decision tree makes queries to individual input bits, then it forms an NP-type (DNF) oracle query.

We define a $\mathsf{BPP}^{\mathsf{NP}\|[q]}_{\epsilon}$-type decision tree T for $f\colon \{0,1\}^n \to \{0,1\}$ on input x as follows.

Syntax: $T \sim \left\{T_s : s \in \{0,1\}^r\right\}$ where each T_s makes queries to the bits of x until it reaches a leaf, which is labeled with a tuple of DNFs $(\varphi_1, \ldots, \varphi_q)$ and a function $out\colon \{0,1\}^q \to \{0,1\}$; the output is then $out(\varphi_1(x), \ldots, \varphi_q(x))$.
Correctness: The output is $f(x)$ with probability $\geq \frac{1}{2} + \frac{1}{2}\epsilon$.
Cost: The maximum height of any T_s, plus the maximum width of any DNF appearing at a leaf.

An $\mathsf{RP}^{\mathsf{NP}\|[q]}_{\epsilon}$-type decision tree is defined similarly except for correctness we require the output is always 0 if $f(x) = 0$, and is 1 with probability $\geq \epsilon$ if $f(x) = 1$. A $\mathsf{ZPP}^{\mathsf{NP}\|[q]}_{\epsilon}$-type decision tree is defined similarly except $out\colon \{0,1\}^q \to \{0,1,\perp\}$ and for correctness, we require the output is always $f(x)$ or $\perp$, and is $f(x)$ with probability $\geq \epsilon$.

We follow the convention of overloading complexity class names as decision tree complexity measures: for $\mathcal{C} \in \left\{\mathsf{BPP}^{\mathsf{NP}\|[q]}, \mathsf{RP}^{\mathsf{NP}\|[q]}, \mathsf{ZPP}^{\mathsf{NP}\|[q]}\right\}$, $\mathcal{C}^{\mathsf{dt}}_{\epsilon}(f)$ denotes the minimum cost of any $\mathcal{C}_{\epsilon}$-type decision tree for a partial function f, and $\mathcal{C}^{\mathsf{dt}}_{\epsilon}$ also denotes the class of all families of f's with $\mathcal{C}^{\mathsf{dt}}_{\epsilon}(f) \leq \mathrm{polylog}(n)$, and we define

$$\mathcal{C}^{\mathsf{dt}}_{>\epsilon} = \bigcup_{\text{constants } c} \mathcal{C}^{\mathsf{dt}}_{\epsilon+\log^{-c} n} \quad \text{and} \quad \mathcal{C}^{\mathsf{dt}}_{\epsilon>} = \bigcap_{\text{constants } d} \mathcal{C}^{\mathsf{dt}}_{\epsilon-n^{-d}}.$$

3 Two-sided error

To prove Theorem 1, we first restate it in a more convenient form.

▶ **Theorem 1** (Two-sided error, restated). *For integers $1 \leq q \leq k$:*

(i) *If k, q are odd:* $\mathsf{BPP}^{\mathsf{NP}[1]}_{>1/(k+1)} \subseteq \mathsf{BPP}^{\mathsf{NP}\|[q]}_{q/k>}$.

(ii) *If k is even:* $\mathsf{BPP}^{\mathsf{NP}[1]}_{>1/(k+1)} \subseteq \mathsf{BPP}^{\mathsf{NP}\|[q]}_{q/k>}$.

(iii) *If q, k are even:* $\mathsf{BPP}^{\mathsf{NP}[1]}_{1/(k-1)} \not\subseteq \mathsf{BPP}^{\mathsf{NP}\|[q]}_{>q/k}$ *relative to an oracle.*

(iv) *If q is odd:* $\mathsf{BPP}^{\mathsf{NP}[1]}_{1/k} \not\subseteq \mathsf{BPP}^{\mathsf{NP}\|[q]}_{>q/k}$ *relative to an oracle.*

We prove the inclusions (i) and (ii) in Section 3.1 and the separations (iii) and (iv) in Section 3.2.

3.1 Inclusions

We prove the $q = 1$ case of (i) in Section 3.1.1 and the $q = 1$ case of (ii) in Section 3.1.2 (together these show that $\mathsf{BPP}^{\mathsf{NP}[1]}_{>1/(k+1)} \subseteq \mathsf{BPP}^{\mathsf{NP}[1]}_{1/k>}$ for all integers $k \geq 1$), then we generalize to the $q > 1$ case of (i) in Section 3.1.3 and the $q > 1$ case of (ii) in the full version [6]. The techniques from [3] for the zero-sided error setting are not particularly helpful for the two-sided error setting, so we develop the ideas from scratch.

We now describe the common setup. For some constant c we have $L \in \mathsf{BPP}^{\mathsf{NP}[1]}_{1/(k+1)+n^{-c}}$, witnessed by a polynomial-time randomized algorithm M (taking input x and coin tosses $s \in \{0,1\}^r$) and a language $L' \in \mathsf{NP}$. For an arbitrary constant d, we wish to show $L \in \mathsf{BPP}^{\mathsf{NP}\|[q]}_{q/k-2^{-n^d}}$.

Fix an input x. The first step is to sample a sequence of $m = O(n^{2c+d})$ many independent strings $s^1, \ldots, s^m \in \{0,1\}^r$, so with probability $\geq 1 - 2^{-n^d-1}$, the sequence is *good* in the sense that on input x, M still has advantage strictly greater than $\frac{1}{k+1}$ when its coin tosses are chosen uniformly from the multiset $\{s^1, \ldots, s^m\}$. Then we design a polynomial-time randomized algorithm which, given a good sequence, outputs $L(x)$ with advantage $\geq \frac{q}{k}$ after making q nonadaptive NP oracle queries. Hence, over the random $s^1, \ldots, s^m$ and the other randomness of our algorithm,

$$\begin{aligned}\mathbb{P}[\text{output is } L(x)] &\geq \mathbb{P}\big[\text{output is } L(x) \,\big|\, s^1, \ldots, s^m \text{ is good}\big] - \mathbb{P}[s^1, \ldots, s^m \text{ is bad}] \\ &\geq \left(\tfrac{1}{2} + \tfrac{1}{2} \cdot \tfrac{q}{k}\right) - 2^{-n^d-1} \;=\; \tfrac{1}{2} + \tfrac{1}{2}\left(\tfrac{q}{k} - 2^{-n^d}\right).\end{aligned}$$

Henceforth fix a good sequence $s^1, \ldots, s^m$, and let z^h and $out^h \colon \{0,1\} \to \{0,1\}$ be the query string and truth table produced by $M_{s^h}(x)$ (so the output is $out^h(L'(z^h))$). We assume w.l.o.g. that each out^h is nonconstant, and is hence either identity or negation. Henceforth assume that identity is at least as common as negation among $out^1, \ldots, out^m$; the proof is completely analogous if negation is more common.

Taking probabilities over a uniformly random $h \in [m]$, we make the following definitions.

$$\begin{aligned}\alpha &= \tfrac{1}{2}\mathbb{P}[out^h = \mathrm{id}] & \beta &= \tfrac{1}{2}\mathbb{P}[out^h = \mathrm{neg}] \\ a &= \mathbb{P}\big[out^h = \mathrm{id},\ L'(z^h) = 1\big] - \alpha & b &= \mathbb{P}\big[out^h = \mathrm{neg},\ L'(z^h) = 1\big] - \beta\end{aligned}$$

The key observation is now

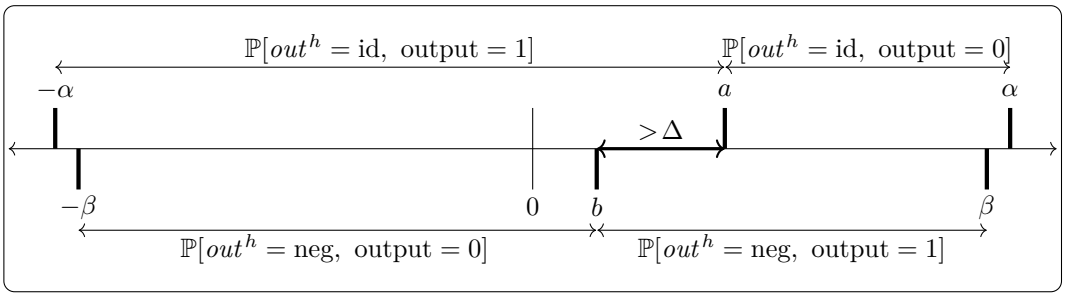
$$\begin{aligned}&(a + \alpha) + (\beta - b) \\ &= \mathbb{P}\big[out^h = \mathrm{id},\ \text{output} = 1\big] + \big(\mathbb{P}[out^h = \mathrm{neg}] - \mathbb{P}\big[out^h = \mathrm{neg},\ \text{output} = 0\big]\big) \\ &= \mathbb{P}\big[out^h = \mathrm{id},\ \text{output} = 1\big] + \mathbb{P}\big[out^h = \mathrm{neg},\ \text{output} = 1\big] \;=\; \mathbb{P}[\text{output} = 1]\end{aligned}$$

and thus, defining $\Delta = \frac{1}{2} \cdot \frac{1}{k+1}$, we have

$$a - b \;=\; (a + \alpha) + (\beta - b) - \tfrac{1}{2} \;=\; \mathbb{P}[\text{output} = 1] - \tfrac{1}{2} \begin{cases} > \Delta & \text{if } L(x) = 1 \\ < -\Delta & \text{if } L(x) = 0 \end{cases}$$

because of M's advantage w.r.t. a good sequence $s^1, \ldots, s^m$.

This figure shows an example of how these values may fall on the number line if $L(x) = 1$:



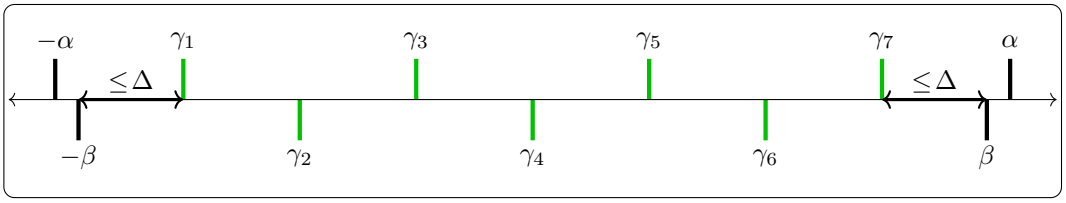
The following summarizes the key properties so far.

$$\alpha \geq \beta \qquad a \in [-\alpha, \alpha] \qquad a - b > \Delta \ \text{ if } L(x) = 1$$
$$\alpha + \beta = \tfrac{1}{2} \qquad b \in [-\beta, \beta] \qquad b - a > \Delta \ \text{ if } L(x) = 0$$

Also, for any real p, testing whether $a \geq p$ can be expressed as an NP oracle query: a witness consists of a list of witnesses for $L'(z^h) = 1$ for at least $(p + \alpha)m$ many h's with $out^h = \text{id}$. Similarly, testing whether $b \geq p$ can be expressed as an NP oracle query.

3.1.1 Proof of (i): $q = 1$

For $i \in [k]$ define $\gamma_i = (i - \frac{k+1}{2})\Delta$. We have $\beta - \gamma_k \leq \Delta$ and $\gamma_1 - (-\beta) \leq \Delta$ since $\beta \leq \frac{1}{4} = \big((k+1) - \frac{k+1}{2}\big)\Delta$. This figure shows an example with $k = 7$:



Our algorithm now picks one of these k possibilities uniformly at random:[2]

- for some odd $i \in [k]$: output 1 iff $a \geq \gamma_i$,
- for some even $i \in [k]$: output 0 iff $b \geq \gamma_i$.

First suppose $L(x) = 1$. We have $a > \gamma_1$ since $a - b > \Delta$ and $b \geq -\beta$ and $\gamma_1 - (-\beta) \leq \Delta$. Consider the greatest odd $j \in [k]$ such that $a \geq \gamma_j$; thus $a \geq \gamma_i$ for $\frac{j+1}{2}$ many odd i's $(1, 3, \ldots, j)$. If $j < k$ then $b < \gamma_{j+1}$ since $a - b > \Delta$ and $a < \gamma_{j+2}$; thus $b < \gamma_i$ for at least $\frac{k-j}{2}$ many even i's $(j+1, j+3, \ldots, k-1)$. Hence the probability of outputting 1 is at least $\frac{1}{k}\big(\frac{j+1}{2} + \frac{k-j}{2}\big) = \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{k}$.

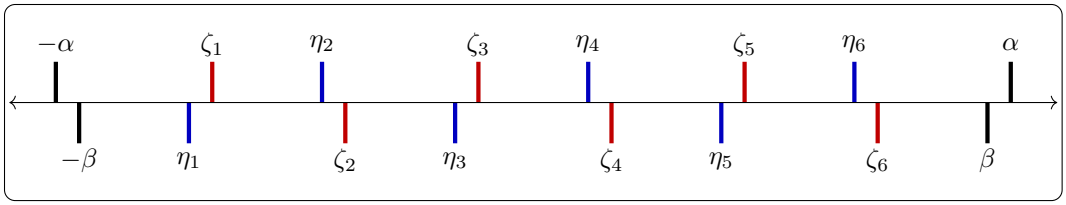
Now suppose $L(x) = 0$. We have $a < \gamma_k$ since $b - a > \Delta$ and $b \leq \beta$ and $\beta - \gamma_k \leq \Delta$. Consider the least odd $j \in [k]$ such that $a < \gamma_j$; thus $a < \gamma_i$ for $\frac{k-j+2}{2}$ many odd i's $(j, j+2, \ldots, k)$. If $j > 1$ then $b > \gamma_{j-1}$ since $b - a > \Delta$ and $a \geq \gamma_{j-2}$; thus $b \geq \gamma_i$ for at least $\frac{j-1}{2}$ many even i's $(2, 4, \ldots, j-1)$. Hence the probability of outputting 0 is at least $\frac{1}{k}\big(\frac{k-j+2}{2} + \frac{j-1}{2}\big) = \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{k}$.

[2] Of course, if k is not a power of 2 and we insist on using uniform coin flips as our only source of randomness, then we must incur a tiny error since it is not possible to exactly sample $i \in [k]$ uniformly. We sweep this pedantic issue under the rug throughout the paper.

That concludes the formal proof, but here is an intuitive way to visualize what is happening: Call γ_i for odd i "upper marks," and call γ_i for even i "lower marks," and assume for convenience all lower marks are in $(-\beta, \beta)$. Suppose $L(x) = 1$ and $b = -\beta$ so $a > \gamma_1$; then at least one upper mark is left of a and all $\frac{k-1}{2}$ lower marks are right of b, resulting in $\frac{k+1}{2}$ of the algorithm's possibilities outputting 1. Now as we continuously sweep a and b to the right, keeping $a - b$ fixed, a passes each upper mark before b passes the preceding lower mark, so at all times at least $\frac{k+1}{2}$ of the possibilities output 1. Suppose $L(x) = 0$ and $b = \beta$ so $a < \gamma_k$; then at least one upper mark is right of a and all $\frac{k-1}{2}$ lower marks are left of b, resulting in $\frac{k+1}{2}$ of the algorithm's possibilities outputting 0. Now as we continuously sweep a and b to the left, keeping $b - a$ fixed, a passes each upper mark before b passes the succeeding lower mark, so at all times at least $\frac{k+1}{2}$ of the possibilities output 0.

3.1.2 Proof of (ii): $q = 1$

For $i \in [k]$ define $\zeta_i = -\beta + i\Delta$ and $\eta_i = -\alpha + i\Delta$. Note that $\alpha - \zeta_k = \Delta$ (so $\zeta_1, \ldots, \zeta_k$ divide the interval $[-\beta, \alpha]$ into $k+1$ subintervals each of length Δ) and $\beta - \eta_k = \Delta$ (so $\eta_1, \ldots, \eta_k$ divide the interval $[-\alpha, \beta]$ into $k+1$ subintervals each of length Δ). This figure shows an example with $k = 6$:



Our algorithm now picks one of these $2k$ possibilities uniformly at random:

- for some odd $i \in [k]$: output 1 iff $a \geq \zeta_i$,
- for some even $i \in [k]$: output 0 iff $b \geq \zeta_i$,
- for some even $i \in [k]$: output 1 iff $a \geq \eta_i$,
- for some odd $i \in [k]$: output 0 iff $b \geq \eta_i$.

First suppose $L(x) = 1$. We have $a > \zeta_1$ since $a - b > \Delta$ and $b \geq -\beta$. Consider the greatest odd $j \in [k]$ such that $a \geq \zeta_j$; thus $a \geq \zeta_i$ for $\frac{j+1}{2}$ many odd i's $(1, 3, \ldots, j)$. We have $b < \zeta_{j+1}$ since $a - b > \Delta$ and either $a < \zeta_{j+2}$ (if $j < k-1$) or $a \leq \alpha$ and $\alpha - \zeta_k = \Delta$ (if $j = k-1$); thus $b < \zeta_i$ for at least $\frac{k-j+1}{2}$ many even i's $(j+1, j+3, \ldots, k)$. Consider the greatest even $j' \in [k]$ such that $a \geq \eta_{j'}$, or let $j' = 0$ if it does not exist; thus $a \geq \eta_i$ for $\frac{j'}{2}$ many even i's $(2, 4, \ldots, j')$. If $j' < k$ then $b < \eta_{j'+1}$ since $a - b > \Delta$ and $a < \eta_{j'+2}$; thus $b < \eta_i$ for at least $\frac{k-j'}{2}$ many odd i's $(j'+1, j'+3, \ldots, k-1)$. Hence the probability of outputting 1 is at least $\frac{1}{2k}\left(\frac{j+1}{2} + \frac{k-j+1}{2} + \frac{j'}{2} + \frac{k-j'}{2}\right) = \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{k}$.

Now suppose $L(x) = 0$. Consider the least odd $j \in [k]$ such that $a < \zeta_j$, or let $j = k+1$ if it does not exist; thus $a < \zeta_i$ for $\frac{k-j+1}{2}$ many odd i's $(j, j+2, \ldots, k-1)$. If $j > 1$ then $b > \zeta_{j-1}$ since $b - a > \Delta$ and $a \geq \zeta_{j-2}$; thus $b \geq \zeta_i$ for at least $\frac{j-1}{2}$ many even i's $(2, 4, \ldots, j-1)$. We have $a < \eta_k$ since $b - a > \Delta$ and $b \leq \beta$ and $\beta - \eta_k = \Delta$. Consider the least even $j' \in [k]$ such that $a < \eta_{j'}$; thus $a < \eta_i$ for $\frac{k-j'+2}{2}$ many even i's $(j', j'+2, \ldots, k)$. We have $b > \eta_{j'-1}$ since $b - a > \Delta$ and either $a \geq \eta_{j'-2}$ (if $j' > 2$) or $a \geq -\alpha$ (if $j' = 2$); thus $b \geq \eta_i$ for at least $\frac{j'}{2}$ many odd i's $(1, 3, \ldots, j'-1)$. Hence the probability of outputting 0 is at least $\frac{1}{2k}\left(\frac{k-j+1}{2} + \frac{j-1}{2} + \frac{k-j'+2}{2} + \frac{j'}{2}\right) = \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{k}$.

That concludes the formal proof, but here is an intuitive way to visualize what is happening: Call ζ_i for odd i and η_i for even i "upper marks," and call ζ_i for even i and η_i for odd i "lower marks," and assume for convenience all lower marks are in $(-\beta, \beta)$. Suppose $L(x) = 1$ and $b = -\beta$ so $a > \zeta_1$; then at least one upper mark is left of a and all k lower marks are right of b, resulting in $k+1$ of the algorithm's possibilities outputting 1. Now as we continuously sweep a and b to the right, keeping $a - b$ fixed, a passes each upper mark (ζ_i or η_i) before b passes the corresponding preceding lower mark (ζ_{i-1} or η_{i-1} respectively), so at all times at least $k+1$ of the possibilities output 1. Suppose $L(x) = 0$ and $b = \beta$ so $a < \eta_k$; then at least one upper mark is right of a and all k lower marks are left of b, resulting in $k+1$ of the algorithm's possibilities outputting 0. Now as we continuously sweep a and b to the left, keeping $b - a$ fixed, a passes each upper mark (ζ_i or η_i) before b passes the corresponding succeeding lower mark (ζ_{i+1} or η_{i+1} respectively), so at all times at least $k+1$ of the possibilities output 0.

3.1.3 Proof of (i): $q > 1$

For $i \in [k]$ define I_i as the set of q successive integers starting with i and wrapping around to 1 when k is exceeded: $I_i = \{i, i+1, \ldots, i+q-1\}$ if $i \leq k-q+1$, and $I_i = \{i, i+1, \ldots, k, 1, 2, \ldots, i+q-1-k\}$ if $i > k-q+1$. Define $i^{\nwarrow} = \min(\text{odd } i' \in I_i) - k - 1$ and $i^{\swarrow} = \min(\text{even } i' \in I_i) - k - 1$; the $-k-1$ is a simple way to ensure $i^{\nwarrow}, i^{\swarrow} < \min(i' \in I_i)$. Since k, q are odd, the sorted order of $I_i \cup \{i^{\nwarrow}, i^{\swarrow}\}$ alternates between odd and even numbers.

Our algorithm picks $i \in [k]$ uniformly at random and for each $i' \in I_i$ does an oracle query to see whether $a \geq \gamma_{i'}$ if i' is odd, or whether $b \geq \gamma_{i'}$ if i' is even. Consider the greatest odd $i^\sharp \in I_i$ such that $a \geq \gamma_{i^\sharp}$, or let $i^\sharp = i^{\nwarrow}$ if it does not exist. Consider the greatest even $i^\flat \in I_i$ such that $b \geq \gamma_{i^\flat}$, or let $i^\flat = i^{\swarrow}$ if it does not exist. Our algorithm outputs 1 if $i^\sharp > i^\flat$, or 0 if $i^\flat > i^\sharp$.

First suppose $L(x) = 1$. Consider the greatest odd $j \in [k]$ such that $a \geq \gamma_j$ (which exists since $a > \gamma_1$). We have $i^\sharp > i^\flat$ if one of the following mutually exclusive events holds:

(1) $j \in I_i$, since then $i^\sharp = j$ and $i^\flat \leq j-1$ (since $b < \gamma_{j+1}$ if $j < k$);
(2) i is odd and $i \leq j-q-1$, since then $i^\sharp = i+q-1$ and trivially $i^\flat \leq i+q-2$;
(3) i is even and $j+1 \leq i \leq j-q-1+k$, since then either:
- $i \leq k-q$, in which case $i^\sharp = i^{\nwarrow} > i^{\swarrow} = i^\flat$, or
- $i = k-q+2$, in which case $i^\sharp = 1$ and $i^\flat = i^{\swarrow} < 1$, or
- $i \geq k-q+4$, in which case $i^\sharp = i+q-1-k$ and $i^\flat \leq i+q-2-k$.

There are q many type-(1) i's. If $j > q$ then there are $\frac{j-q}{2}$ many type-(2) i's $(1, 3, \ldots, j-q-1)$ and $\frac{k-j}{2}$ many type-(3) i's $(j+1, j+3, \ldots, k-1)$. If $j \leq q$ then there are $\frac{k-q}{2}$ many type-(3) i's $(j+1, j+3, \ldots, j-q-1+k)$. Either way, $i^\sharp > i^\flat$ holds for at least $q + \frac{k-q}{2} = \frac{k+q}{2}$ many i's, and hence the probability of outputting 1 is at least $\frac{1}{k} \cdot \frac{k+q}{2} = \frac{1}{2} + \frac{1}{2} \cdot \frac{q}{k}$.

Now suppose $L(x) = 0$. Consider the least odd $j \in [k]$ such that $a < \gamma_j$ (which exists since $a < \gamma_k$). As a special case, if $j = 1$ then $i^\sharp = i^{\nwarrow}$ and so $i^\flat > i^\sharp$ if $i^{\swarrow} > i^{\nwarrow}$, which happens for $\frac{k+q}{2}$ many i's $(1, 3, \ldots, k-q+1$ and $k-q+2, k-q+3, \ldots, k)$. Now assume $j > 1$. We have $i^\flat > i^\sharp$ if one of the following mutually exclusive events holds:

(1) $j-1 \in I_i$, since then $i^\sharp \leq j-2$ and $i^\flat \geq j-1$ (since $b > \gamma_{j-1}$ if $j > 1$);
(2) i is even and $i \leq j-q-2$, since then $i^\sharp = i+q-2$ and $i^\flat = i+q-1$;
(3) i is odd and $j \leq i \leq j-q-2+k$, since then either:
- $i \leq k-q+1$, in which case $i^\sharp = i^{\nwarrow} < i^{\swarrow} \leq i^\flat$, or
- $i \geq k-q+3$, in which case $i^\sharp = i+q-2-k$ and $i^\flat \geq i+q-1-k$.

There are q many type-(1) i's. If $j > q$ then there are $\frac{j-q-2}{2}$ many type-(2) i's $(2, 4, \ldots, j-q-2)$ and $\frac{k-j+2}{2}$ many type-(3) i's $(j, j+2, \ldots, k)$. If $j \leq q$ then there are $\frac{k-q}{2}$ many type-(3) i's $(j, j+2, \ldots, j-q-2+k)$. Either way, $i^\flat > i^\sharp$ holds for at least $q + \frac{k-q}{2} = \frac{k+q}{2}$ many i's, and hence the probability of outputting 0 is at least $\frac{1}{k} \cdot \frac{k+q}{2} = \frac{1}{2} + \frac{1}{2} \cdot \frac{q}{k}$.

3.2 Separations

The relativized separations follow routinely [5] from the corresponding decision tree complexity separations:

(iii) If q, k are even: $\mathsf{BPP}^{\mathsf{NP}[1]\mathsf{dt}}_{1/(k-1)} \not\subseteq \mathsf{BPP}^{\mathsf{NP}\|[q]\mathsf{dt}}_{>q/k}$.

(iv) If q is odd: $\mathsf{BPP}^{\mathsf{NP}[1]\mathsf{dt}}_{1/k} \not\subseteq \mathsf{BPP}^{\mathsf{NP}\|[q]\mathsf{dt}}_{>q/k}$.

We prove (iii) in Section 3.2.1 and (iv) in the full version [6]; the arguments are similar in structure. Our proof of (iv) also works if q is even, but in that case the result is subsumed by (iii). The case $q = 1$, $k = 2$ of (iii) was proven in [7], but our proof is somewhat different even specialized to that case.

Let $\mathrm{wt}(\cdot)$ refer to Hamming weight. Henceforth fix the constants q and k, and assume $q < k$ since otherwise there is nothing to prove.

3.2.1 Proof of (iii)

Define the partial function $f\colon \{0,1\}^n \to \{0,1\}$ that interprets its input as $(x, y) \in \{0,1\}^{n/2} \times \{0,1\}^{n/2}$, such that

$$f(x,y) = \begin{cases} 1 & \text{if } \mathrm{wt}(x) = \mathrm{wt}(y) + 1 \leq \frac{k}{2} \\ 0 & \text{if } \mathrm{wt}(x) = \mathrm{wt}(y) \leq \frac{k}{2} - 1 \end{cases}.$$

▶ **Lemma 4.** $\mathsf{BPP}^{\mathsf{NP}[1]\mathsf{dt}}_{1/(k-1)}(f) \leq \frac{k}{2}$.

▶ **Lemma 5.** $\mathsf{BPP}^{\mathsf{NP}\|[q]\mathsf{dt}}_{q/k+\delta}(f) \geq \Omega(\delta n)$ *for every* $\delta(n)$.

The separation follows by taking $\delta = \log^{-c} n$ for any constant c.

Proof of Lemma 4. Given (x, y), pick one of these $k-1$ possibilities uniformly at random:

- for some $i \in [\frac{k}{2}]$: output 1 iff $\mathrm{wt}(x) \geq i$,
- for some $i \in [\frac{k}{2} - 1]$: output 0 iff $\mathrm{wt}(y) \geq i$.

The decision tree does not directly query any bits of (x, y), and the DNF has width $i \leq \frac{k}{2}$ (it is the OR over all i-subsets of either x's bits or y's bits, of the AND of those bits), so the cost is $\frac{k}{2}$. If $f(x, y) = 1$ with $\mathrm{wt}(x) = j$ and $\mathrm{wt}(y) = j - 1$, then the probability of outputting 1 is $\frac{j+((k/2-1)-(j-1))}{k-1} = \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{k-1}$ since conditioned on picking x, the output is 1 iff $i \leq j$, and conditioned on picking y, the output is 1 iff $i \geq j$. Similarly, if $f(x, y) = 0$ with $\mathrm{wt}(x) = \mathrm{wt}(y) = j$, then the probability of outputting 1 is $\frac{j+((k/2-1)-j)}{k-1} = \frac{1}{2} - \frac{1}{2} \cdot \frac{1}{k-1}$. ◀

Proof of Lemma 5. It suffices to show that for some distribution on valid inputs (x, y) to f, every cost-$o(\delta n)$ $\mathsf{P}^{\mathsf{NP}\|[q]}$-type decision tree T has advantage $< \frac{q}{k} + \delta$ over a random input. Let $T(x, y)$ denote the output produced after T receives the answers to its DNF queries. Let u be the leaf reached after seeing only 0's, and say u is labeled with DNFs $(\varphi_1, \ldots, \varphi_q)$ and function $out\colon \{0,1\}^q \to \{0,1\}$ (so if (x, y) leads to u then $T(x, y) = out(\varphi_1(x, y), \ldots, \varphi_q(x, y))$).

We generate the distribution on valid inputs (x, y) as follows. Let $v^0 = w^0 \in \{0,1\}^{n/2}$ be the all-0 string, and for $i = 1, \ldots, \frac{k}{2}$ obtain v^i by flipping a uniformly random 0 of v^{i-1}

to a 1, and for $i = 1, \ldots, \frac{k}{2} - 1$ obtain w^i by flipping a uniformly random 0 of w^{i-1} to a 1. Pick a uniformly random $j \in [\frac{k}{2}]$, and then let (x, y) be either the 1-input (v^j, w^{j-1}) or the 0-input (v^{j-1}, w^{j-1}) with probability $\frac{1}{2}$ each.

Let v denote $(v^0, \ldots, v^{k/2})$ and w denote $(w^0, \ldots, w^{k/2-1})$, and call (v, w) *good* iff:

- for each $j \in [\frac{k}{2}]$: both inputs (v^j, w^{j-1}) and (v^{j-1}, w^{j-1}) lead to u, and
- for each $j \in [\frac{k}{2}]$ and each $i \in [q]$: $\varphi_i(v^j, w^{j-1}) \geq \varphi_i(v^{j-1}, w^{j-1}) \geq \varphi_i(v^{j-1}, w^{j-2})$ (the latter inequality is only required if $j > 1$).

We claim that

(1) $\mathbb{P}[(v, w) \text{ is bad}] < \frac{\delta}{2}$, and

(2) $\mathbb{P}\big[T(x, y) = f(x, y) \,\big|\, (v, w) \text{ is good}\big] \leq \frac{1}{2} + \frac{1}{2} \cdot \frac{q}{k}$,

from which it follows that

$$\mathbb{P}[T(x, y) = f(x, y)] \leq \mathbb{P}\big[T(x, y) = f(x, y) \,\big|\, (v, w) \text{ is good}\big] + \mathbb{P}[(v, w) \text{ is bad}] < \tfrac{1}{2} + \tfrac{1}{2}(\tfrac{q}{k} + \delta).$$

We argue claim (1). Since the path to u queries $o(\delta n)$ locations, with probability $\geq 1 - o(k\delta) > 1 - \frac{\delta}{4}$ each of the 1's placed throughout v and w avoids these locations, in which case the first bullet holds in the definition of good. Fixing j and i in the second bullet, if we condition on $\varphi_i(v^{j-1}, w^{j-1}) = 1$ and choose an arbitrary term of φ_i that accepts (v^{j-1}, w^{j-1}), then since the term has width $o(\delta n)$, with probability $\geq 1 - o(\delta)$ the 1 that is placed to obtain v^j from v^{j-1} avoids this term, in which case the term continues to accept (v^j, w^{j-1}) and so $\varphi_i(v^j, w^{j-1}) = 1$. Thus $\mathbb{P}\big[\varphi_i(v^j, w^{j-1}) \geq \varphi_i(v^{j-1}, w^{j-1})\big] \geq \mathbb{P}\big[\varphi_i(v^j, w^{j-1}) = 1 \,\big|\, \varphi_i(v^{j-1}, w^{j-1}) = 1\big] \geq 1 - o(\delta)$. Similarly, $\mathbb{P}\big[\varphi_i(v^{j-1}, w^{j-1}) \geq \varphi_i(v^{j-1}, w^{j-2})\big] \geq 1 - o(\delta)$. A union bound over j and i shows that the second bullet holds with probability $\geq 1 - o(kq\delta) > 1 - \frac{\delta}{4}$, so finally the two bullets hold simultaneously with probability $> 1 - \frac{\delta}{2}$.

We argue claim (2). Condition on any particular good (v, w). We abbreviate the q-tuple $(\varphi_1(x, y), \ldots, \varphi_q(x, y))$ as $\varphi(x, y) \in \{0, 1\}^q$. Consider the sequence of k inputs (v^0, w^0), $(v^1, w^0), (v^1, w^1), (v^2, w^1), \ldots$ (like climbing a ladder but placing both feet on each rung). Each of these possibilities for (x, y) leads to u and thus $T(x, y) = \mathit{out}(\varphi(x, y))$. Also, the corresponding sequence of $\varphi(x, y)$'s is monotonically nondecreasing in each of the q coordinates. Thus the sequence of inputs can be partitioned into segments of lengths say $\ell_0, \ell_1, \ldots, \ell_q$ (which sum to k) such that for the first ℓ_0 (x, y)'s in the sequence, $\varphi(x, y)$ has weight 0 (hence $T(x, y)$ is the same), and for the next ℓ_1 (x, y)'s in the sequence, $\varphi(x, y)$ is the same weight-1 string (hence $T(x, y)$ is the same), and so on. Since each segment alternates between 0-inputs and 1-inputs of f, we have $T(x, y) = f(x, y)$ for at most $\lceil \frac{\ell_i}{2} \rceil \leq \frac{\ell_i + 1}{2}$ inputs in the i^{th} segment.

Thus, out of the k possibilities for (x, y) given (v, w), at most $\sum_{i=0}^{q} \frac{\ell_i + 1}{2} = \frac{k}{2} + \frac{q+1}{2}$ are such that $T(x, y) = f(x, y)$. This implies that $\mathbb{P}\big[T(x, y) = f(x, y) \,\big|\, (v, w) \text{ is good}\big] \leq \frac{1}{2} + \frac{1}{2} \cdot \frac{q+1}{k}$, which is almost what we want. This issue can be fixed by observing that since k is even and $q + 1$ (the number of segments) is odd, at least one segment must have even length, in which case $\lceil \frac{\ell_i}{2} \rceil = \frac{\ell_i}{2}$. Thus, out of the k possibilities for (x, y) given (v, w), $T(x, y) = f(x, y)$ holds for at most $\frac{k}{2} + \frac{q}{2}$ of them, which gives (2). ◀

4 Zero-sided error

We now prove Theorem 2, restated here for convenience.

▶ **Theorem 2** (Zero-sided error, restated). *For integers $1 \leq q \leq k \leq 4$:*

(i) *If $k = 4$:* $\mathsf{ZPP}^{\mathsf{NP}[1]}_{>0} \subseteq \mathsf{ZPP}^{\mathsf{NP}\|[q]}_{q/k>}$.

(ii) *If* $k \leq 3$*:* $\mathsf{ZPP}^{\mathsf{NP}[1]}_{>1/(k+1)} \subseteq \mathsf{ZPP}^{\mathsf{NP}\|[q]}_{q/k>}$.

(iii) $\mathsf{ZPP}^{\mathsf{NP}[1]}_{1/k} \not\subseteq \mathsf{ZPP}^{\mathsf{NP}[1]}_{>1/k}$ *relative to an oracle.*

Moreover, the "$q/k>$*" in the inclusion subscripts can be improved to "*q/k*" if* $q < k \geq 3$.

We prove the inclusions (i) and (ii) in Section 4.1 and the separations (iii) in the full version [6].

4.1 Inclusions

Straightforwardly generalizing the proof of $\mathsf{ZPP}^{\mathsf{NP}[1]}_{>0} \subseteq \mathsf{ZPP}^{\mathsf{NP}[1]}_{1/4}$ in [3] yields (i), but we take a different tack by showing in Section 4.1.1 that (i) follows directly from Theorem 3. We prove (ii) from first principles in Section 4.1.2; our proof for the case $k = 1$ is equivalent to the one in [3], but we include it for completeness.

4.1.1 Proof of (i)

Let $L \in \mathsf{ZPP}^{\mathsf{NP}[1]}_{>0} \subseteq \mathsf{RP}^{\mathsf{NP}[1]}_{>0}$. By Theorem 3 and closure of $\mathsf{ZPP}^{\mathsf{NP}[1]}_{>0}$ under complement,

$$L \in \mathsf{RP}^{\mathsf{NP}[1]}_{1/2} \text{ by some algorithm } M^1, \qquad L \in \mathsf{RP}^{\mathsf{NP}\|[2]}_{1>} \text{ by some algorithm } M^2,$$
$$\overline{L} \in \mathsf{RP}^{\mathsf{NP}[1]}_{1/2} \text{ by some algorithm } \overline{M}^1, \qquad \overline{L} \in \mathsf{RP}^{\mathsf{NP}\|[2]}_{1>} \text{ by some algorithm } \overline{M}^2.$$

We let each of these four M-algorithms refer to the entire computation, including the NP oracle queries, which we elide for convenience. (Note that $\overline{M}^i$ does not mean "complement of M^i" – it is a different algorithm.) We assume M^2 and $\overline{M}^2$ have advantage $\geq 1 - 2^{-n^d}$ for an arbitrary constant d. Furthermore, we assume all four algorithms have been modified to output $\perp$ instead of 0, and $\overline{M}^1$ and $\overline{M}^2$ have been modified to output 0 instead of 1.

If $q = 1$: $L \in \mathsf{ZPP}^{\mathsf{NP}[1]}_{1/4}$ by running M^1 or $\overline{M}^1$ with probability $\frac{1}{2}$ each.

If $q = 2$: $L \in \mathsf{ZPP}^{\mathsf{NP}\|[2]}_{1/2}$ by running M^1 and $\overline{M}^1$, and if one of them outputs a bit, outputting that bit or $\perp$ otherwise.

If $q = 4$: $L \in \mathsf{ZPP}^{\mathsf{NP}\|[4]}_{1>}$ by running M^2 and $\overline{M}^2$, and if one of them outputs a bit, outputting that bit or $\perp$ otherwise.

If $q = 3$: $L \in \mathsf{ZPP}^{\mathsf{NP}\|[3]}_{3/4>}$ by running M^1 and $\overline{M}^2$ with probability $\frac{1}{2}$, or M^2 and $\overline{M}^1$ with probability $\frac{1}{2}$, and if one of them outputs a bit, outputting that bit or $\perp$ otherwise. This falls slightly short of our promise of showing $L \in \mathsf{ZPP}^{\mathsf{NP}\|[3]}_{3/4}$, but that can be fixed by noting that the proof of Theorem 3 actually shows that M^1 and $\overline{M}^1$ can have advantage $\geq \frac{1}{2} + 2^{-n^e}$ for some constant e depending on L. Then taking $d \geq e$ ensures we get advantage $\geq \frac{1}{2}\left(\frac{1}{2} + 2^{-n^e}\right) + \frac{1}{2}\left(1 - 2^{-n^d}\right) \geq \frac{3}{4}$.

4.1.2 Proof of (ii)

We just prove $\mathsf{ZPP}^{\mathsf{NP}[1]}_{>1/(k+1)} \subseteq \mathsf{ZPP}^{\mathsf{NP}\|[q]}_{q/k>}$; the "moreover" part follows by exactly the same trick (due to [3]) for strengthening $\mathsf{RP}^{\mathsf{NP}[1]}_{>0} \subseteq \mathsf{RP}^{\mathsf{NP}[1]}_{1/2>}$ to $\mathsf{RP}^{\mathsf{NP}[1]}_{>0} \subseteq \mathsf{RP}^{\mathsf{NP}[1]}_{1/2}$, which we describe in the full version [6].

For some constant c we have $L \in \mathsf{ZPP}^{\mathsf{NP}[1]}_{1/(k+1)+n^{-c}}$, witnessed by a polynomial-time randomized algorithm M (taking input x and coin tosses $s \in \{0,1\}^r$) and a language $L' \in \mathsf{NP}$. For an arbitrary constant d, we wish to show $L \in \mathsf{ZPP}^{\mathsf{NP}\|[q]}_{q/k-2^{-n^d}}$.

Fix an input x. The first step is to sample a sequence of $m = O(n^{2c+d})$ many independent strings $s^1, \ldots, s^m \in \{0,1\}^r$, so with probability $\geq 1 - 2^{-n^d}$, the sequence is *good* in the sense that on input x, M still has advantage strictly greater than $\frac{1}{k+1}$ when its coin tosses are chosen uniformly from the multiset $\{s^1, \ldots, s^m\}$. Then we design a polynomial-time randomized algorithm which, given a good sequence, outputs $L(x)$ with probability $\geq \frac{q}{k}$ after making q nonadaptive NP oracle queries, and which has zero-sided error for all sequences (good and bad). Hence, over the random $s^1, \ldots, s^m$ and the other randomness of our algorithm,

$$\mathbb{P}[\text{output is } L(x)] \;\geq\; \mathbb{P}\big[\text{output is } L(x) \,\big|\, s^1, \ldots, s^m \text{ is good}\big] - \mathbb{P}[s^1, \ldots, s^m \text{ is bad}] \;\geq\; \tfrac{q}{k} - 2^{-n^d}.$$

Henceforth fix a good sequence $s^1, \ldots, s^m$, and let z^h and $out^h \colon \{0,1\} \to \{0,1\}$ be the query string and truth table produced by $M_{s^h}(x)$ (so the output is $out^h(L'(z^h))$). We assume w.l.o.g. that out^h is nonconstant. If there is an h such that $out^h \in \{\text{id}, \text{neg}\}$, then our algorithm simply uses the NP oracle to evaluate $L'(z^h)$ and then outputs $out^h(L'(z^h)) = L(x)$. Otherwise, each out^h is one of the four functions out_{ab} (for $ab \in \{0,1\}^2$) that maps a to b and $1-a$ to $\perp$:

	out_{00}	out_{01}	out_{10}	out_{11}
0	0	1	$\perp$	$\perp$
1	$\perp$	$\perp$	0	1

For each $ab \in \{0,1\}^2$ consider the "ab" query:

$\exists\, h :\ out^h = out_{ab}$ and $L'(z^h) = a$?

If $a = 1$ then the "ab" query can be expressed as an NP oracle query: a witness consists of an h with $out^h = out_{ab}$ and a witness for $L'(z^h) = 1$. If $a = 0$ then the "ab" query can be expressed as a coNP oracle query: a nonexistence witness consists of a witness for $L'(z^h) = 1$ for each h such that $out^h = out_{ab}$. We say the "ab" query returns yes iff it indicates the existence of such an h (i.e., the NP oracle returns the bit a). If the "ab" query returns yes, we can safely output b since there exists an h such that $out^h(L'(z^h)) = out_{ab}(a) = b = L(x)$.

Our algorithm is:

1. Identify a set $P \subseteq \{0,1\}^2$ of size k for which there is guaranteed to exist an $ab \in P$ such that the "ab" query would return yes.
2. Pick a uniformly random $Q \subseteq P$ of size q.
3. For each $ab \in Q$ do the "ab" query and output b if it returns yes.
4. Finally output $\perp$ if all queries returned no.

This outputs $L(x)$ with probability $\geq \frac{q}{k}$. We just need to prove that we can indeed find such a P in step 1. Let $H = \{h \in [m] : M_{s^h} \text{ outputs } L(x)\}$ (so by assumption, $|H| > \frac{m}{k+1}$) and $H_{ab} = \{h \in [m] : out^h = out_{ab}\}$. Note that the "$ab$" query would return yes iff $H \cap H_{ab} \neq \emptyset$, and that $H \subseteq H_{0b} \cup H_{1b}$ for $b = L(x)$.

If $k = 3$: Let P contain all ab's except the one with the smallest H_{ab} (which has size $\leq \frac{m}{4}$), breaking ties arbitrarily. Then $H \cap H_{ab} \neq \emptyset$ for at least one $ab \in P$ assuming $|H| > \frac{m}{4}$.

If $k = 2$: If $|H_{00} \cup H_{10}| \leq \frac{m}{3}$ then $L(x) = 1$ assuming $|H| > \frac{m}{3}$, so we can let $P = \{01, 11\}$. Similarly, if $|H_{01} \cup H_{11}| \leq \frac{m}{3}$ then we can let $P = \{00, 10\}$. Otherwise, the smaller of H_{00}, H_{10} has size $\leq \frac{m}{3}$, and the smaller of H_{01}, H_{11} has size $\leq \frac{m}{3}$, so we can let P contain the two ab's corresponding to the larger of H_{00}, H_{10} and the larger of H_{01}, H_{11}, breaking ties arbitrarily.

If $k = 1$: If $|H_{00} \cup H_{10}| \leq \frac{m}{2}$ then $L(x) = 1$ assuming $|H| > \frac{m}{2}$, and furthermore the smaller of H_{01}, H_{11} has size $\leq \frac{m}{2}$, so we can let P contain the ab corresponding to the larger of H_{01}, H_{11}. Similarly, if $|H_{01} \cup H_{11}| < \frac{m}{2}$ then we can let P contain the ab corresponding to the larger of H_{00}, H_{10}.

References

1 Richard Beigel. Bounded Queries to SAT and the Boolean Hierarchy. *Theoretical Computer Science*, 84(2):199–223, 1991. doi:10.1016/0304-3975(91)90160-4.

2 Jin-Yi Cai and Venkatesan Chakaravarthy. On Zero Error Algorithms Having Oracle Access to One Query. *Journal of Combinatorial Optimization*, 11(2):189–202, 2006. doi:10.1007/s10878-006-7130-0.

3 Richard Chang and Suresh Purini. Amplifying $\mathrm{ZPP}^{\mathrm{SAT}[1]}$ and the Two Queries Problem. In *Proceedings of the 23rd Conference on Computational Complexity (CCC)*, pages 41–52. IEEE, 2008. doi:10.1109/CCC.2008.32.

4 Rahul Tripathi. The 1-Versus-2 Queries Problem Revisited. *Theory of Computing Systems*, 46(2):193–221, 2010. doi:10.1007/s00224-008-9126-x.

5 Nikolai Vereshchagin. Relativizability in Complexity Theory. In *Provability, Complexity, Grammars*, volume 192 of *AMS Translations, Series 2*, pages 87–172. American Mathematical Society, 1999.

6 Thomas Watson. Amplification with One NP Oracle Query. Technical Report TR18-058, Electronic Colloquium on Computational Complexity (ECCC), 2018. URL: https://eccc.weizmann.ac.il/report/2018/058/.

7 Thomas Watson. A $\mathrm{ZPP}^{\mathrm{NP}[1]}$ Lifting Theorem. In *Proceedings of the 36th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 59:1–59:16. Schloss Dagstuhl, 2019. doi:10.4230/LIPIcs.STACS.2019.59.

Separating k-Player from t-Player One-Way Communication, with Applications to Data Streams

David P. Woodruff
Carnegie Mellon University, Pittsburgh, PA, USA
dwoodruf@andrew.cmu.edu

Guang Yang
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
Conflux, Beijing, China
guang.research@gmail.com

Abstract

In a k-party communication problem, the k players with inputs $x_1, x_2, \ldots, x_k$, respectively, want to evaluate a function $f(x_1, x_2, \ldots, x_k)$ using as little communication as possible. We consider the message-passing model, in which the inputs are partitioned in an arbitrary, possibly worst-case manner, among a smaller number t of players ($t < k$). The t-player communication cost of computing f can only be smaller than the k-player communication cost, since the t players can trivially simulate the k-player protocol. But how much smaller can it be? We study deterministic and randomized protocols in the one-way model, and provide separations for product input distributions, which are optimal for low error probability protocols. We also provide much stronger separations when the input distribution is non-product.

A key application of our results is in proving lower bounds for data stream algorithms. In particular, we give an optimal $\Omega(\varepsilon^{-2}\log(N)\log\log(mM))$ bits of space lower bound for the fundamental problem of $(1\pm\varepsilon)$-approximating the number $\|x\|_0$ of non-zero entries of an n-dimensional vector x after m updates each of magnitude M, and with success probability $\geq 2/3$, in a strict turnstile stream. Our result matches the best known upper bound when $\varepsilon \geq 1/\mathsf{polylog}(mM)$. It also improves on the prior $\Omega(\varepsilon^{-2}\log(mM))$ lower bound and separates the complexity of approximating L_0 from approximating the p-norm L_p for p bounded away from 0, since the latter has an $O(\varepsilon^{-2}\log(mM))$ bit upper bound.

2012 ACM Subject Classification Theory of computation → Streaming models; Theory of computation → Complexity classes; Theory of computation → Lower bounds and information complexity

Keywords and phrases Communication complexity, multi-player communication, one-way communication, streaming complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.97

Category Track A: Algorithms, Complexity and Games

Related Version The full version hosted on arXiv `https://arxiv.org/abs/1905.07135`.

Funding This work was supported in part by the National Natural Science Foundation of China Grants No. 61433014, 61602440, 61761136014, 61872334, 61502449, and the 973 Program of China Grant No. 2016YFB1000201.

Acknowledgements We would like to thank Yuval Ishai and Eyal Kushilevitz for initiating the problem of separating worst-case partition communication complexity from streaming complexity, which was our starting point. We also thank the ICALP referees for very helpful comments which helped us revise our initial submission. D. Woodruff would also like to thank the Chinese Academy of Sciences, as well as the Simons Institute for the Theory of Computing.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 97; pp. 97:1–97:14

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

1 Introduction

Consider a k-party communication problem, in which the players have inputs $x_1, x_2, \ldots, x_k$ respectively, and want to compute a function $f(x_1, x_2, \ldots, x_k)$ of their inputs using as little communication as possible. We consider the message-passing model, in which the inputs are partitioned in an arbitrary, possibly worst-case manner among a smaller number t of players. That is, we partition $\{1, 2, \ldots, k\}$ into t subsets $S_1, S_2, \ldots, S_t$ such that $\cup_{i=1}^{t} S_i = \{1, 2, \ldots, k\}$ and $S_i \cap S_j = \emptyset$ for every $1 \leq i < j \leq t$, and let the i-th player P_i hold the sequence of inputs $y_i := \left(x_{i_1}, x_{i_2}, \ldots, x_{i_{|S_i|}}\right)$. We are still interested in computing the original function f. The total communication required must be smaller than in the original k-player setting, since the t players can simulate the protocol involving the original k players. A natural question is: *how much smaller can the communication be?*

There are many communication models that are possible, but our main motivation for looking at this question comes from applications to data streams, see below, and so we are primarily interested in the *one-way number-in-hand* model. In this model, each of the t players can only see its own input. The first player composes a message m_1 based on its input y_1 and sends m_1 to the second player. The second player takes m_1 and its input y_2 to compute a message m_2 for the third player, and so on. The t-th (also the last) player, upon receiving the message m_{t-1} from the $(t-1)$-st player, computes the output of the protocol based on m_{t-1} and its own input y_t. We sometimes abuse notation and refer to the output as m_t. The total communication cost is the maximum of $\sum_{i=1}^{t} |m_i|$, where $|m_i|$ denotes the length of the i-th message and the maximum is taken over all possible inputs $y_1, \ldots, y_t$ (which is a partition of $\{x_1, \ldots, x_k\}$) and all random coin tosses of the players. For streaming applications we are especially interested in $\max_{i \in \{1,\ldots,t\}} |m_i|$.

To explain the connection to data streams, almost all known lower bound arguments on the memory required of a data stream algorithm are proven via communication complexity, or at least can be reformulated using communication complexity. The basic idea is to partition the elements of an input stream contiguously, consisting of say k elements, into a possibly smaller number t of players. Then one argues that if there is a data stream algorithm solving the problem, then the communication problem can be solved by passing the memory contents as messages from player to player. Note that this naturally gives rise to the one-way number-in-hand model. Since the total communication cost is $t \cdot S$, where S is the size of the memory of the streaming algorithm, if the randomized t-player communication complexity of the function f is CC_t, we must have $S \geq CC_t/t$. Many lower bounds in data streams are proven already with two players. However, it is known that for some functions more players are needed to obtain stronger lower bounds, such as for estimating the frequency moments in insertion only streams (see, e.g., [3, 17] and references therein).

One cannot help but ask *how powerful is communication complexity for proving data stream lower bounds*? Another natural question is: *for a given function f, which number t of players should one partition the stream into*? Yet another question is regarding the input distribution – should it be a product distribution for which the inputs to the players are chosen independently, or should the inputs be drawn from a non-product distribution to obtain the best space lower bounds? Since we are interested in the limits of using t players for establishing lower bounds for data stream algorithms, we allow the original k inputs (which correspond to the k elements in a stream) to be partitioned in the worst possible way for a t-player communication protocol, as this will give the strongest possible lower bound.

1.1 Our Results

In this paper we study these communication questions and their connections to data streams.

We first make the simple observation that for non-product input distributions, the communication complexity can be arbitrarily smaller if we partition the k inputs into $t < k$ players. Indeed, consider the k-player set disjointness problem in which the i-th player, $1 \leq i \leq k$, has a set $S_i \subseteq [n]$, where for notational simplicity we define $[n] := \{1, 2, \ldots, n\}$ for $n \in \mathbb{N}$. The input distribution satisfies the promise that either (1) $S_i \cap S_j = \emptyset$ for every $1 \leq i < j \leq k$, or (2) there is a unique item $a \in [n]$ such that $a \in S_i$ for all $i \in [k]$, and for any other $a' \neq a$, there is at most one $i \in [k]$ for which $a' \in S_i$. It is well-known that the randomized communication complexity of this problem is $\Omega(n/k)$ [3, 8, 10], and that the bound holds even for multiple rounds of communication and players share a common blackboard. However, if we look at $t < k$ players and an arbitrary, even if the worst-case mapping of the input sets $S_1, \ldots, S_k$ to the t players, then by the pigeonhole principle there exists a player who gets two input sets S_i, S_j with $i \neq j$. Now this player can locally determine the output of the function by checking if $S_i \cap S_j = \emptyset$. Thus with $t < k$ players the problem is solvable using $O(1)$ bits per player. This simple argument shows that for non-product distributions, there can be an arbitrarily large gap between the k-player and the t-player worst-case-partitioned randomized communication complexities. Note that this example applies to a symmetric problem, meaning that the k-player set disjointness problem is invariant under any one-to-one assignment of $x_1, \ldots, x_k$ to the k players.

Perhaps surprisingly, and this is one of the main messages of our work: for symmetric functions and product input distributions, we show that for any $t < k$, for deterministic one-way communication complexity or randomized one-way communication complexity with error probability $1/\mathsf{poly}(k)$, there *is no gap* in maximum message length between the k-player and t-player communication complexities. That is, the gap is at most a multiplicative $O(1)$ factor in message length and $O(k)$ in total communication. Further, this gap is tight, as there are problems for which the input distribution is a product distribution, and the t-player communication with $1/\mathsf{poly}(k)$ error probability is $O(\log k)$ for constant $t = O(1)$, while the k-player communication with $1/\mathsf{poly}(k)$ error probability is $\Omega(k \log k)$. Thus, the answer for product input distributions is significantly different than what we saw for non-product distributions, even for symmetric functions.

We also show that for constant error protocols and under product input distributions, the gap is at most a multiplicative $O(\log k)$ factor in message length and $O(k \log k)$ in total communication. Further, we show there exists a symmetric function and input distribution which is product on any $k-1$ out of k inputs, for which this gap is best possible. We leave open the question of the existence of a symmetric function and product input distribution (on all k inputs rather than $k-1$ out of k) which realizes this gap for constant error protocols.

One takeaway message from our results is that when showing space lower bounds for data stream algorithms computing symmetric functions on product distributions, by looking at 2-player communication complexity (which is by far the most common communication setup), there is only an $O(1)$ factor loss for error probability $1/\mathsf{poly}(k)$ protocols, and an $O(\log k)$ factor loss for constant error protocols.

Data Stream Lower Bounds: As a key application of our lower bound techniques, we provide a space lower bound for $(1 \pm \varepsilon)$-approximating the *Hamming norm* in the strict turnstile model. This problem, which is also known as the L_0 *norm estimation* and denoted by T_ε, requires estimating $\|\mathbf{x}\|_0 := |\{i \mid x_i \neq 0\}|$ of a vector $\mathbf{x} = (x_1, \ldots, x_N)$ and outputting an estimate $\widetilde{F}$ for which $(1-\varepsilon)\|\mathbf{x}\|_0 \leq \widetilde{F} \leq (1+\varepsilon)\|\mathbf{x}\|_0$ with constant probability. The vector $\mathbf{x}$ is initialized to all zeros and undergoes a sequence of m updates each of the

form $(i, v) \in [N] \times [\pm M]$, where $[\pm M] := \{0, \pm 1, \ldots, \pm M\}$ and each update (i, v) causes $x_i \leftarrow x_i + v$. In the strict turnstile model $x_i \geq 0$ holds for all i and at all points in the stream. We obtain an $\Omega\left(\varepsilon^{-2} \log(N) \log\log(mM)\right)$ bits of space lower bound for $(1 \pm \varepsilon)$-approximating the Hamming norm. This lower bound matches the best known upper bound $O\left(\varepsilon^{-2} \log(N) \left(\log(1/\varepsilon) + \log\log(mM)\right)\right)$ [12] for any $\varepsilon \geq 1/\mathsf{polylog}(mM)$. Note that $\varepsilon \geq 1/\mathsf{polylog}(mM)$ is required in order to obtain polylogarithmic space, and so is the most common setting of parameters. Perhaps surprisingly, there is an upper bound of $O\left(\varepsilon^{-2} \log(mM)\right)$ bits of space for $(1 \pm \varepsilon)$-approximating L_p for $p > 0$ [13] (improving an earlier $O\left(\log^2 N\right)$ bound of [9]; see also a time-efficient version in [11]), and thus we provide a strict separation in the complexities for $p = 0$ and $p > 0$. The Hamming norm has many applications, as it corresponds to estimating the number of distinct values, and can be used to estimate set union and intersection sizes (see [7] where it was introduced).

Technical Overview: We first illustrate the idea behind showing there is no gap between k-player and 2-player deterministic one-round communication complexity. The first player P_1 of the k-player protocol pretends to be Alice, the first player of the 2-player protocol, to create the message m_1 as Alice would do and sends it to the second player P_2 of the k-player protocol. Having received this message m_1, P_2 enumerates over all possible inputs of P_1 until finding one which would cause P_1 to send m_1. Since the protocol is deterministic and it evaluates a function defined on a product domain, meaning that it is a total function on a domain of the form $S_1 \times S_2 \times \cdots \times S_k$, the function value must be the same as long as P_1's input results in the same message m_1 to be sent. So P_2 can arbitrarily pick one of those inputs as his guess for P_1. Now P_2 has a guess x for P_1's input together with his own input y, and P_2 can simulate Alice in the 2-player protocol. This is feasible because the 2-player protocol works under any partitioning of the inputs. Then P_2 sends to the third player P_3 the message that Alice would send to Bob in the 2-player protocol, given that Alice had input (x, y). In case when every player P_i cannot figure out how many input items have been processed from his own input and the received message m_{i-1}, which is important for his simulation of the 2-player protocol, an additional logarithmic-many-bits index carrying this piece of information should be passed together with the simulated messages. In this way, the entire k-player protocol can be simulated and the per player communication equals to the communication of the 2-player protocol between Alice and Bob, sometimes plus the additional logarithmic many bits for the index. Moreover, both protocols are deterministic.

For the randomized case with a product input distribution, we first consider 2-player protocols with error probability $1/\mathsf{poly}(k)$. We would like to run the same simulation as for deterministic protocols, except now it is unclear how the second player P_2 can reconstruct a valid input x for the first player P_1 from the first message m_1. A natural thing would be for P_2 to choose the input x_1 to P_1 for which the probability of sending m_1, given that P_1's input is x_1, is greatest. This is not correct though, since the overall probability of P_1 holding x_1 and sending m_1 may be less than the $1/\mathsf{poly}(k)$ error bound and the protocol could afford to be always wrong on such a combination of x_1 and m_1. Thus we need some balancing between two probabilities: i) the first player P_1 sends m_1 on input x_1; and ii) the protocol output is correct given that P_1 has input x_1 and sends m_1.

The above naturally suggests that we should impose an input product distribution μ. Then it must be that for a good fraction of x, weighted according to μ, the k-player protocol is correct when the first player has input x_1 and sends message m_1. Thus we can sample x from the conditional distribution on μ given that message m_1 is sent. Here, for correctness, it is crucial that μ is a product distribution; this ensures for most settings of remaining player's inputs (weighted according to μ), for most choices of x_1 (weighted according to μ)

giving rise to m_1, the function evaluated on the inputs is the same, and x_1 can be sampled independently of remaining inputs. Once we have sampled x_1, and given that the second player has private input x_2 in the k-player protocol, we can then have the second player pretend to be Alice of a randomized 2-player protocol with input (x_1, x_2), similar to the deterministic case. Ultimately, we will show that under distribution μ we obtain a protocol with total communication at most $O(k)$ times that of the 2-player protocol with error probability $1/\mathsf{poly}(k)$ (and an $O(1)$ multiplicative blowup in maximum message length, times that of the 2-player protocol), where the factor k comes from the number of invocations of the 2-player protocol.

We illustrate the optimality of the randomized reduction above by looking at the Sum-Equal problem studied by Viola [16]: in this problem each of k players holds an input x_i mod p, where $p = \Theta\left(k^{1/4}\right)$ is a prime, and they wish to determine whether $\sum_i x_i = 0$ or $1 \mod p$. Viola shows this problem has randomized communication complexity $\Theta(k \log k)$, for both randomized protocols with constant error probability as well as deterministic protocols (and thus also randomized protocols with $1/\text{poly}(k)$ error probability). Moreover, for randomized protocols with $1/\text{poly}(k)$ error probability, Viola's $\Omega(k \log k)$ lower bound holds even for a product distribution on the inputs (where if $\sum_i x_i \mod p \notin \{0, 1\}$ the output can be arbitrary). We observe that under any partition of the inputs into 2-players Alice and Bob, the problem can be solved with $O(\log k)$ bits with probability $1 - 1/\text{poly}(k)$ just by running an equality test on the sum modulo p of Alice and the negated sum modulo p of Bob. Thus, this illustrates that the factor $O(k)$ gap for protocols for product input distributions with $1/\text{poly}(k)$ error probability is *optimal*.

On the other hand, for constant error protocols and a product input distribution, there is a 2-player $O(1)$ bit upper bound in the public coin model which comes from running an equality test with constant error probability (since we measure error with respect to an input distribution, equality has an $O(1)$ upper bound with constant error). We note that the k-player protocol has communication $\Omega(k \log k)$ for constant error protocols, which gives the $\Omega(k \log k)$ factor gap we claimed. The only downside is that the $\Omega(k \log k)$ lower bound holds for an input distribution which is product on $k - 1$ out of k players, rather than all k players. We leave it as an open question to give an optimal separation for product input distributions for constant error probability.

Given the importance of Viola's problem in showing separations, we next show a *direct sum theorem* for his problem, showing its communication complexity increases to $\Omega(kr \log k)$ for solving a constant fraction of r independent copies. To show the direct sum theorem for Viola's problem, one issue is that, unlike for two players where the technique of *information complexity* often provides direct sum theorems, for k-players the analogues are much weaker. A natural route would be to take Viola's *corruption bound*, argue it implies a high information bound, and then apply standard direct sum theorems for information. This approach does not give an information cost lower bound on private coin protocols, though one can fix it for two players using [5], which improves upon a bound in [6]. However, for k players similarly strong bounds are unknown. Another natural approach is to use the fact that if a problem has a corruption bound, then one immediately has a direct sum for it [4]. Again though, this is only for two players or the *number on forehead* model, and not for our setting.

Instead, our proof is inspired by Viola's rectangle argument for a single copy of the Sum-Equal problem, where each rectangle, restricted to the first $k - 1$ players, is a product distribution on which the protocol generates a message to the k-th player. We use a rectangle argument on multiple copies where the output is now a binary vector instead of a single bit. The main obstacle is that we must consider the Hamming distance between the protocol

output and the correct answer in a vector space, which is much more involved than studying the error probability for a single instance. The intuition of our proof is that for every large rectangle, there must be linearly many copies that appear (almost) uniformly random in the last player's view. The above argument is fairly intricate, and involves several levels of conversion: i) a large rectangle implies large conditional entropy in many players' inputs; ii) the large entropy of all copies implies we have min-entropy at least 1 on many copies; iii) a random variable of min-entropy at least 1 can always be decomposed into a convex combination of uniform distributions over two elements; iv) the summation of sufficiently many independent random variables that are each drawn from a uniform-over-two-element distribution turns out to be nearly uniform, and hence many Sum-Equal copies look uniform to the last player.

Thus, the last player can hardly outperform a random guess. Note that it is insufficient to prove uniformity for many copies individually (which is not too hard using the same idea as in Viola's proof), since such a situation could be simulated with a much smaller rectangle with very small error. We instead perform our rectangle argument inductively to show most copies appear almost uniform, even if conditioned on previous copies. For space considerations this induction is mostly deferred to the full version.

This direct sum technique has further applications. One application is to proving a lower bound for approximating the Hamming norm in a strict turnstile stream. Using a result of [2], to show lower bounds for streaming algorithms in the strict turnstile model, it suffices to show lower bounds in the simultaneous communication model, where each player simultaneously sends a message to a referee who outputs the answer. While our direct sum theorem holds in this more restrictive model, we also need to consider a composition of the gap-Hamming problem on top of the Sum-Equal instances as well as an augmented index version of the composed problem. In the augmented problem we additionally give a referee an index i and the answers to all copies j, with $j > i$. Similar augmentation has been studied for L_p-norms [13]. This allows us to reduce our communication problem to Hamming norm approximation, and ultimately prove our data stream lower bound.

2 Preliminaries

A function $f : \Sigma^k \to \Gamma$ is called a *k-party symmetric function* if for every $(x_1, x_2, \ldots, x_k) \in \Sigma^k$ and for every permutation σ over $\{1, 2, \ldots, k\}$, there is $f(x_1, \ldots, x_k) = f\left(x_{\sigma(1)}, \ldots, x_{\sigma(k)}\right)$. A k-dimensional vector space S is called a *product space* if it can be represented as $S = S_1 \times S_2 \times \cdots \times S_k$. A distribution μ is called a *product distribution* if it is obtained by taking the product of k independent distributions, i.e., $\mu = \mu_1 \times \mu_2 \times \cdots \times \mu_k$.

In the t-player communication complexity model, there are t computationally unbounded players, e.g., $P_1, \ldots, P_t$, required to compute a function $f : X_1 \times \cdots \times X_t \to Y$, where f is usually a t-party symmetric function. Each player P_i is given a private input $x_i \in X_i$ and follows a fixed protocol to exchange messages. For every input $(x_1, \ldots, x_t)$, the message transcript is denoted by $\Pi_t(x_1, \ldots, x_t)$ when all players follow the protocol Π_t (when Π_t is randomized, $\Pi_t(x_1, \ldots, x_t)$ is a random variable taking probabilities over players' random coins). A deterministic protocol Π_t computes f if there is a function Π_{out} such that $\Pi_{out}\left(\Pi_t^{(t)}(x_1, \ldots, x_t), x_t\right) \equiv f$, where $\Pi_t^{(t)}(x_1, \ldots, x_t)$ denotes P_t's view under the execution of Π_t on input $(x_1, \ldots, x_t)$ and for simplicity we let $\Pi_{out}(x_1, \ldots, x_t) := \Pi_{out}\left(\Pi_t^{(t)}(x_1, \ldots, x_t), x_t\right)$. A δ-error randomized protocol Π_t for f requires the existence of Π_{out} such that for all inputs $(x_1, \ldots, x_t)$, $\Pr\left[\Pi_{out}(x_1, \ldots, x_t) = f(x_1, \ldots, x_t)\right] \geq 1 - \delta$. The *communication cost* of Π_t is the maximum size of $\Pi_t(x_1, \ldots, x_t)$ over all $x_1, \ldots, x_t$ and all

random coins. The *t-player deterministic communication complexity* (resp. *t-player δ-error randomized communication complexity*), denoted by $\mathbf{DCC}_t(f)$ (resp. $\mathbf{RCC}_{t,\delta}(f)$), is the cost of the best t-player deterministic (resp. δ-error randomized) protocol Π_t for f.

Given a k-party function $f : X_1 \times \cdots \times X_k \to Y$ and $t < k$, we define $\mathbf{DCC}_t(f)$ and $\mathbf{RCC}_{t,\delta}(f)$ under a *worst-case partition* of inputs. That is, let $f_t(z_1, \ldots, z_t) = f(x_1, \ldots, x_k)$ be defined for every partition $i_0 = 0 \le i_1 \le \cdots \le i_t = k$ and $z_j := (x_{i_{j-1}+1}, \ldots, x_{i_j})$, and the t-player communication complexity of f is defined with respect to the worst choice of f_t, i.e., $\mathbf{DCC}_t(f) := \max_{f_t} \mathbf{DCC}_t(f_t)$ and $\mathbf{RCC}_{t,\delta}(f) := \max_{f_t} \mathbf{RCC}_{t,\delta}(f_t)$.

Given a t-party function f and its input distribution μ, we let $\mathbf{DCC}^{\mu}_{t,\delta}(f)$ denote the communication cost of the best t-player deterministic protocol Π_t computing f such that $\Pr_{x\sim\mu}[\Pi_{out}(x) \neq f(x)] \le \delta$. Similarly we define $\mathbf{RCC}^{\mu}_{t,\delta}(f)$ for randomized protocols.

In the restricted *one-way communication model* [15, 1, 14], the i-th player sends exactly one message to the $(i+1)$-st player for $i \in [t-1]$ following Π_t, and then P_t announces the output of Π_t as specified by Π_{out}. Note that in this setting there are only $k-1$ messages sent by $P_1, \ldots, P_{k-1}$, and we do not count the final output announced by P_t in the communication in order to best correspond to streaming algorithms. This is also known as a *sententious* protocol in previous work, e.g., [16]. We denote the *t-player one-way communication complexities of f* by $\overrightarrow{\mathbf{DCC}}_t(f)$ and $\overrightarrow{\mathbf{RCC}}_{t,\delta}(f)$, respectively.

In the *common reference string model* (aka *CRS model*), there is a sequence of public random coins, which is by default a uniformly random binary string, accessible to all players. The obvious advantage of communication in the CRS model is that players have access to the same random string and thus save the cost of synchronizing their private coins.

A streaming algorithm is an algorithm that scans the input $(x_1, \ldots, x_m) \in \Sigma^m$ as m stream input items in sequence, updates its internal memory of size $s = o(m \log |\Sigma|)$ (i.e., a streaming automaton with 2^s states, where the space cost of updating the internal memory is not accounted for), and finally outputs a function $f(x_1, \ldots, x_m)$ evaluated on all input items. If the best deterministic (resp. δ-error randomized) streaming algorithm computes f with s bits of memory and t passes over the data stream, then we say the *deterministic* (resp. *δ-error*) *streaming complexity* of f is st, denoted by $\mathbf{DSC}(f) = st$ (resp. $\mathbf{RSC}_\delta(f) = st$). In a popular and standard setting, a streaming algorithm scans the input stream in a *single pass* and only processes every input item once. The necessary amount of memory required by such single-pass algorithms is called the *single-pass deterministic/δ-error streaming complexity* and denoted by $\overrightarrow{\mathbf{DSC}}(f)$ and $\overrightarrow{\mathbf{RSC}}_\delta(f)$ respectively.

Note that every streaming algorithm can be naturally interpreted as a communication protocol where each party holds some (possibly an empty set of) input items on the stream and the messages capture the memory updates. The connection between streaming complexity and communication complexity trivially follows in the following lemma.

▶ **Lemma 1.** *For every function f and error tolerance δ, for every $k \in \mathbb{N}$, it holds that:*

$$\mathbf{DSC}(f) \ge \frac{1}{k} \cdot \mathbf{DCC}_k(f), \quad \mathbf{RSC}_\delta(f) \ge \frac{1}{k} \cdot \mathbf{RCC}_{k,\delta}(f)$$

Furthermore, similar relations hold for $\overrightarrow{\mathbf{DSC}}$, $\overrightarrow{\mathbf{RSC}}_\delta$ and $\overrightarrow{\mathbf{DCC}}_k$, $\overrightarrow{\mathbf{RCC}}_{k,\delta}$.

3 Communication Complexity for Functions on Non-Product Spaces

▶ **Theorem 2.** *For every $t \ge 2$, there is a t-party symmetric function f defined on $D \subseteq \{0,1\}^n = \left(\{0,1\}^{n/t}\right)^t$ such that for $\delta < 1/4$, $\overrightarrow{\mathbf{DCC}}_{t-1}(f) \le t-1$ but $\mathbf{RCC}_{t,\delta}(f) = \Omega(n/t)$. If $t = O(1)$, then $\overrightarrow{\mathbf{DCC}}_{t-1}(f) = O(1)$ and $\mathbf{RSC}_\delta(f) \ge \frac{1}{t} \cdot \mathbf{RCC}_{t,\delta}(f) = \Omega(n)$.*

Proof. Consider the t-party set disjointness problem $\textsc{Disj}_{n/t,t}$ defined as follows: there are t players $P_1, \ldots, P_t$ such that every player P_i holds a private indicator vector $\mathbf{x}_i \in \{0,1\}^{n/t}$ which represents a subset of $[n/t]$, i.e., $\textsc{Disj}_{n/t,t}(\mathbf{x}_1, \ldots, \mathbf{x}_t) = \vee_{j=1}^{n/t} (\wedge_{i=1}^{t} x_{i,j})$, where $x_{i,j}$ denotes the j-th coordinate of $\mathbf{x}_i$. We consider the domain D such that the vectors $\mathbf{x}_1, \ldots, \mathbf{x}_t \in \{0,1\}^{n/t}$ are either (1) pairwise disjoint, or (2) sharing a unique element $j \in [n/t]$. Let f be the function that computes $\textsc{Disj}_{n/t,t}$ on domain D.

On the one hand, it is easy to verify that $\overrightarrow{\mathbf{DCC}}_{t-1}(f) \leq t-1$. Indeed, at least one of the $t-1$ players obtains two distinct indicator vectors and hence can itself decide the output of f. The communication is 1 bit per player to pass the result, and hence the total communication is bounded by $t-1$ since there are $t-1$ players.

On the other hand, the $\Omega(n/t)$ lower bound for $\mathbf{RCC}_{t,\delta}(f)$ follows from the known lower bound for multi-player set disjointness (see [3], which was improved to optimal in [8, 10]). The lower bound for $\mathbf{RSC}_{\delta}(f)$ immediately follows by Lemma 1. ◀

4 Deterministic Communication and Streaming Complexity

We first show that 2-player one-way communication complexity is equivalent to the streaming complexity of single-pass streaming algorithms in the deterministic setting.

▶ **Theorem 3.** *For every symmetric function f, $\overrightarrow{\mathbf{DCC}}_2(f) \leq \overrightarrow{\mathbf{DSC}}(f) \leq \overrightarrow{\mathbf{DCC}}_2(f) + \log n$.*

Proof. Obviously, $\overrightarrow{\mathbf{DSC}}(f) \geq \overrightarrow{\mathbf{DCC}}_2(f)$ since a 2-player communication protocol simulates a streaming algorithm. It remains to prove $\overrightarrow{\mathbf{DSC}}(f) \leq \overrightarrow{\mathbf{DCC}}_2(f) + \log n$.

Suppose the input stream is $(x_1, \ldots, x_n) \in \Sigma^n$, and for every partition into $(x_1, \ldots, x_i)$ and $(x_{i+1}, \ldots, x_n)$ there is a deterministic 2-player one-way protocol Π_2^i computing f. We design the deterministic single-pass streaming algorithm A for f by simulating 2-player one-way communication protocols under different partitions. The memory usage of A is therefore bounded by the maximum communication cost of the simulated 2-player protocols plus an index in $[n]$ recording the number of processed items. Notice that when processing the item x_{i+1}, A has already processed $x_1, \ldots, x_i$ and has (m_i, i) in memory. A can thus reconstruct a compatible guess of $x_1'', \ldots, x_i''$ that would induce exactly the message m_i as in Π_2^i, and then sets the memory to be $(m_{i+1}, i+1)$ where m_{i+1} is the message sent in Π_2^{i+1} when P_1 has $(x_1'', \ldots, x_i'', x_{i+1})$ and P_2 has $(x_{i+2}, \ldots, x_n)$. A repeats this process for every $i = 1, \ldots, n-1$ and at the end it outputs $f(x_1, \ldots, x_n)$.

Therefore, we complete the proof with $\overrightarrow{\mathbf{DCC}}_2(f) \leq \overrightarrow{\mathbf{DSC}}(f) \leq \overrightarrow{\mathbf{DCC}}_2(f) + \log n$. ◀

▶ **Corollary 4.** *For every k-party symmetric function f,*

$$(k-1) \cdot \overrightarrow{\mathbf{DCC}}_2(f) \leq \overrightarrow{\mathbf{DCC}}_k(f) \leq (k-1) \cdot \left(\overrightarrow{\mathbf{DCC}}_2(f) + \log k \right)$$

Proof. Combining Lemma 1 and Theorem 3, it follows that

$$\overrightarrow{\mathbf{DCC}}_k(f) \leq (k-1) \cdot \overrightarrow{\mathbf{DSC}}(f) \leq (k-1) \cdot \left(\overrightarrow{\mathbf{DCC}}_2(f) + \log k \right)$$

The other direction $\overrightarrow{\mathbf{DCC}}_k(f) \geq (k-1) \cdot \overrightarrow{\mathbf{DCC}}_2(f)$ holds by giving $z_j = \emptyset$ to every player $j \in \{2, \ldots, k-1\}$ in the k-player case, when the problem degenerates to 2-player communication but the same message has to be passed $k-1$ times. ◀

Such a linear separation naturally extends to the communication complexity of t-player versus k-player protocols, as long as $2 \leq t < k$. Thus, the deterministic communication complexity grows *linearly* in the number of parties.

We remark that if every player must get a non-trivial input, i.e., at least one input element to the function, the linear growth remains for some but not all problems. For example, the communication complexity of the parity of k bits is linear in the number of players. However, to decide whether k elements in $[k]$ are distinct, the 2-player protocol requires communication $\log\binom{k}{k/2} \approx k - \log\sqrt{k}$, whereas the k-player worst-case communication grows sublinearly, i.e. for k players the communication is no more than $\sum_{i=1}^{k-1}\log\binom{k}{i} \ll (k-1)\cdot\log\binom{k}{k/2}$.

5 Communication Complexity for Functions on a Product Space

5.1 Separations for Randomized Communication Complexity

In this section, we consider the communication cost of randomized multi-player protocols defined on product input distributions and present a $k\log k$ versus $t\log t$ separation between k-player and t-player communication complexity.

First we introduce the Sum-Equal problem (as used in Viola's work [16]).

The *k-player* Sum-Equal *over integers*, denoted by Sum-Equal$_k$, requires deciding whether $\sum_{i=1}^{k} x_i = 0$, where each player P_i is given an integer x_i as well as k. In the CRS model, an additional public random string is also known to all players. The *k-player* Sum-Equal *over* $\mathbb{Z}_m$, denoted by Sum-Equal$_{k,m}$, is defined similarly as Sum-Equal$_k$, except that the input items are drawn from $\mathbb{Z}_m$ and the summation is over $\mathbb{Z}_m$, for a publicly known m.

▶ **Lemma 5** ([16], Theorem 15 and Theorem 29). *For every $k \in \mathbb{N}$, $0 \le \delta \le 1/3$, and in the CRS model, the k-player δ-error communication complexity of* Sum-Equal *satisfies:*

(a) *For every $m \in \mathbb{N}$, $\overrightarrow{\mathbf{RCC}}_{k,\delta}(\text{Sum-Equal}_{k,m}) = O(k\log(k/\delta))$.*

(b) *For every prime $p \in (k^{1/4}, 2k^{1/4})$, $\mathbf{RCC}_{k,\delta}(\text{Sum-Equal}_{k,p}) = \Omega(k\log k)$.*

In particular, $\mathbf{RCC}_{k,\delta}(\text{Sum-Equal}_{k,p}) = \Theta(k\log k)$ in the CRS model if $\delta = \Omega(1/\mathsf{poly}(k))$.

We remark that Viola's lower bound for Sum-Equal$_{k,p}$ is proved for a non-product distribution μ_H whose support covers exactly a $2/p$ fraction of the whole (product) input space. Thus if a k-player protocol solves Sum-Equal$_{k,p}$ with error $\delta \le 1/k$ on a uniform distribution μ over the whole input space, then its error with respect to μ_H is bounded by $\frac{1/k}{2/p} < k^{-3/4}$. By Lemma 5, the $\Omega(k)$ separation in Corollary 6 naturally follows.

▶ **Corollary 6.** *For prime $p \in (k^{1/4}, 2k^{1/4})$ and $\delta \le 1/\mathsf{poly}(k)$, there is a product distribution μ such that $\mathbf{RCC}^{\mu}_{k,\delta}(\text{Sum-Equal}_{k,p}) = \Omega(k\log k)$, $\overrightarrow{\mathbf{RCC}}_{2,\delta}(\text{Sum-Equal}_{k,p}) = O(\log k)$.*

For a larger error tolerance, say δ is a constant, we have a stronger separation between k-party communication and t-party communication. However, the hard distribution is slightly non-product, that is, it is a product distribution on any $k-1$ out of the k players.

▶ **Corollary 7.** *For every $k \in \mathbb{N}$, there is a k-party symmetric function f such that*

(a) *For any product distribution μ, for every $2 \le t \le k$ and $0 \le \delta \le 1/3$, $\overrightarrow{\mathbf{RCC}}^{\mu}_{t,\delta}(f) = O(t\log(t/\delta))$. In particular, $\overrightarrow{\mathbf{RCC}}^{\mu}_{2,\delta}(f) = O(\log(1/\delta))$.*

(b) *There exists a distribution μ_H, which is product on any $k-1$ out of k players, for which $\mathbf{RCC}^{\mu}_{k,\delta}(f) = \Omega(k\log k)$ as long as $\delta \le 1/3$.*

For $\delta \ge 1/\mathsf{poly}(t)$, the gap between $\mathbf{RCC}^{\mu}_{k,\delta}(f)$ and $\overrightarrow{\mathbf{RCC}}^{\mu}_{t,\delta}(f)$ is bounded as below:

$$\mathbf{RCC}^{\mu}_{k,\delta}(f) \,/\, \overrightarrow{\mathbf{RCC}}^{\mu}_{t,\delta}(f) = \Omega\left(\frac{k\log k}{t\log t}\right)$$

The outline of the proof of Corollary 7 was given in Section 1. That is, the upper bound in part (a) follows from applying $k = t$ in the first part of Lemma 5, while the lower bound in part (b) follows from the second part of Lemma 5. We defer the proofs to the full version.

5.2 Tightness of the Communication Complexity Separation

The following theorem and corollary show tightness of our separations.

▶ **Theorem 8.** *For every k-party function $f : \Sigma^k \to \Gamma$, product distribution μ over Σ^k, and error tolerance $\delta < 1/3$, if the optimal δ-error 2-player one-way protocol for f does not degenerate to the deterministic case, then the following holds:*

$$\overrightarrow{\mathbf{RCC}}^{\mu}_{k,\delta}(f) \,/\, \overrightarrow{\mathbf{RCC}}_{2,\delta}(f) \leq O\left(k \cdot \left(1 + \frac{\log k}{\log(1/\delta)}\right)\right) = \begin{cases} O(k \log k) & \text{if } \delta = \Omega(1) \\ O(k) & \text{if } \delta = 1/k^{\Omega(1)} \end{cases}$$

Proof sketch. We present the major steps and leave the complete proof to the full version.

First we let Π_0 be the optimal δ-error 2-player one-way protocol Π_0 that computes f with communication $C = \overrightarrow{\mathbf{RCC}}_{2,\delta}(f)$, and construct a new protocol Π_2 by taking $M = O\left(1 + \frac{\log k}{\log(1/\delta)}\right)$ repetitions of Π_0 such that the error probability of Π_2 is reduced to $\delta^2/(16k^2)$. Note that Π_2 is still a 2-player one-way protocol but has communication $O(CM)$.

Second we prove that for every product input distribution μ over Σ^k, the k-party function f can be evaluated by a randomized k-player one-way protocol Π_k with communication $O(k \cdot CM)$ and error $\delta/2$ with respect to μ. The idea is that given μ, each player P_i: 1) assumes that the received message m_{i-1} from P_{i-1} will lead to a correct answer with probability $\geq 1 - \frac{\delta}{4k}$; 2) samples a possible input $x'_1, \ldots, x'_{i-1}$ of previous players $P_1, \ldots, P_{i-1}$ on which with probability $\geq 1 - \frac{\delta}{4k}$ the protocol is correct conditioned on m_{i-1} being sent and $(x'_1 \ldots, x'_{i-1}, x_i, \ldots, x_k)$ being the actual input (here we use that μ is a product distribution); 3) and finally sends a message m_i of length $O(CM)$ as in Π_2 where Alice has input $(x'_1, \ldots, x'_{i-1}, x_i)$. By a union bound the error probability of Π_k is bounded by $\delta/2$ with respect to μ. The fact that μ is a product distribution is used in the second step where the sampling process relies on that previous players' inputs are independently distributed from that of future players.

Thus we finish the proof and conclude that $\overrightarrow{\mathbf{RCC}}^{\mu}_{k,\delta}(f) \leq O(kCM)$. ◀

Notice that in the proof of Theorem 8, every message in Π_k has the length bounded by $O(CM)$, which gives an upper bound for the single-pass streaming complexity.

▶ **Corollary 9.** *For every k-party function f and product input distribution μ, and for every $\delta < 1/3$, $\mathbf{RSC}^{\mu}_{\delta}(f) \leq \overrightarrow{\mathbf{RSC}}^{\mu}_{\delta}(f) \leq O\left(1 + \frac{\log k}{\log(1/\delta)}\right) \cdot \overrightarrow{\mathbf{RCC}}_{2,\delta}(f)$.*

6 A Direct Sum for Viola's Problem

We next turn to our direct sum theorem for Viola's problem, which is a crucial building block for our streaming application.

▶ **Theorem 10.** *Let $F : \left(\mathbb{Z}_p^m\right)^k \to \{0,1\}^m$ be the k-party function computing m independent copies of* Sum-Equal$_{k,p}$*, where p is a prime between $k^{1/4}$ and $2k^{1/4}$. For every error tolerance $\delta \in (0, 1/9)$, we say a protocol Π is* correct with probability $1 - \delta$ *if there is a reconstruction function G such that for every fixed $i \in [m]$ and input $x \in \left(\mathbb{Z}_p^m\right)^k$, $G\big(i, \Pi_{out}(x)\big)$ equals the output of the i-th instance of* Sum-Equal$_{k,p}$ *with probability at least $1 - \delta$, over the internal randomness of Π. Then the communication cost of any Π which is correct with probability $1 - \delta$, is $\Omega(mk \log k)$.*

We give a sketch of the proof of Theorem 10 here, and defer the full proof to the full version.

Proof sketch of Theorem 10. First we fix the randomness used in the protocol Π and convert it into a deterministic protocol Π' that has δ error with respect to a specific input distribution $\mathcal{H}$. Here $\mathcal{H} = (X_1, \ldots, X_{k-1}, X_k + v)$ for independent $X_1, \ldots, X_{k-1}$ uniformly distributing over $\mathbb{Z}_p^m$, $X_k = -\sum_{j=1}^{k-1} X_j$ and v uniformly sampled from $\{0,1\}^m$. Note that $\mathcal{H}_{-k} := (X_1, \ldots, X_{k-1})$ is uniform over $\left(\mathbb{Z}_p^m\right)^{k-1}$.

We next recall the intuition behind rectangle arguments in multi-player number-in-hand communication complexity: every k-player (number-in-hand) deterministic protocol with communication at most c partitions the inputs into $C = 2^c$ sets $R^1, R^2, \ldots, R^C$, where each R^i is a *rectangle* in the form of $R^i = R_1^i \times R_2^t \times \ldots \times R_k^i$ such that every input in R^i induces exactly the same transcript π_i. We will use the rectangle argument to show that Π' uses communication $c \geq \Omega(1) \cdot mk \log k$.

The main step is the following claim (with proof sketched later in this subsection):

▷ Claim 11. If $c < \frac{1-9\delta}{135} \cdot mk \log k$, then for every rectangle R satisfying $\Pr[\mathcal{H}_{-k} \in R_{-k}] \geq 1/(3C) = 1/(3 \times 2^c)$, there must be $L \subseteq [m]$ and $\ell := |L| \geq 9\delta m$ such that conditioned on $X_{-k} \in R_{-k}$, the distribution of $X_k^{(L)}$, which is X_k restricted on L, is ℓ/p-close to the uniform distribution over $\mathbb{Z}_p^\ell$.

Using Claim 11, it is easy to show $\Pr[\Pi'(\mathcal{H}) \text{ errs on } \leq 3\delta m \text{ coordinates}] \leq 2/3$, which contradicts that Π' has δ error with respect to $\mathcal{H}$ and $\delta < 1/9$. Therefore, the communication cost of Π', and hence of Π, must be $\geq \frac{1-9\delta}{135} \cdot mk \log k = \Omega(mk \log k)$. ◀

Proof sketch of Claim 11. This claim is proved using induction on the size of L. Suppose the claim is true for (w.l.o.g.) the first $\leq \ell - 1$ indices, we prove it for the next one. More specifically, we show that the last player P_k gets nearly no information about the ℓ-th copy when the input distribution follows $\mathcal{H}$ and X_{-k} falls into a sufficiently large rectangle $R_{-k} = R_1 \times \cdots \times R_{k-1}$. That is, for $X_{-k} \sim \left(\mathbb{Z}_p^m\right)^{k-1}$ and $X_k = -\sum_{j=1}^{k-1} X_j$, the marginal distribution $X_k^{(\ell)} \mid X_{-k} \in R_{-k}$ is statistically close to uniform.

The proof outline is as follows: first, let $\mathcal{E}_x$ denote the event that the first $k-1$ players have x on their first $\ell - 1$ coordinates, i.e. $X_{-k}^{[\ell-1]} = x$. Second, we consider frequently appearing x conditioned on $\mathcal{H}_{-k} \in R_{-k}$ such that $\Pr[\mathcal{E}_x \mid \mathcal{H}_{-k} \in R_{-k}] \geq \frac{1}{2p^{1+(\ell-1)(k-1)}}$ (the missed probability measure is at most $\frac{1}{2p}$ since there are $\leq p^{(\ell-1)(k-1)}$ different choices of x), and let $J_x \subseteq [k-1]$ be the set of players whose input falls into R_{-k} with "significant" probability conditioned on $\mathcal{E}_x$. Specifically, we prove that J_x must have size $|J_x| \geq 0.5k - 1$ for $J_x := \left\{ j \in [k-1] \;\middle|\; \Pr\left[X_j \in R_j \mid \mathcal{E}_x\right] \geq 2^{-1-2c/k} \right\}$. Third, for every player $j \in J_x$, we consider the set $I_{j,x}$ of coordinates such that for every $i \in I_{j,x}$, the conditional min-entropy of $X_j^{(i)}$ is large given that player j's input X_j is consistent with x and falls into R_j. In particular, for $I_{j,x} := \left\{ i \in [m] \mid \mathsf{H}_\infty\left[X_j^{(i)} \mid X_j \in R_j, \mathcal{E}_x\right] \geq 1 \right\}$, there is $|I_{j,x}| > m - \ell - \frac{2(1-9\delta)}{15} m + 1$.

Finally we apply Chebyshev's inequality and a Chernoff bound together with a standard averaging argument to conclude that there is a fixed coordinate, w.l.o.g. we call it ℓ, such that with probability $\geq 1 - e^{-\Omega(k)}$, the conditional min-entropy $\mathsf{H}_\infty\left[X_j^{(\ell)} \mid X_j \in R_j, \mathcal{E}_x\right] \geq 1$ for $\geq k/30$ players $j \in [k-1]$. As a result, the last player P_k's input $X_k^{(\ell)} = -\sum_{j=1}^{k-1} X_j^{(\ell)}$ is a convex combination of random variables where each of them is the summation of $\geq k/30$ uniform-over-two-elements variables. Repeating a very similar argument as in [16], we conclude that $X_k^{(\ell)}$ is $e^{-\Omega(\sqrt{k})}$ close to uniform.

The overall error probability of above arguments is bounded by $1/p$, which sums up to $\leq \ell/p$ for $X_k^{(L)}$ via a standard union bound. ◁

7 Lower bound for Hamming Norm Estimation

In this section we present a space lower bound for single-pass streaming algorithms for $(1\pm\varepsilon)$-approximating the Hamming norm L_0, which is denoted by T_ε as in Section 1.1. Recall that the underlying vector is N-dimensional and there are m updates each of magnitude $[\pm M]$.

▶ **Theorem 12.** *For error tolerance $\varepsilon < 1/3$ and $\varepsilon = \max\left\{\Omega\left(\sqrt{\frac{\log k}{k}}\right), \frac{1}{N^{0.49}}\right\}$, any single-pass streaming algorithm solving T_ε with probability $\geq 2/3$ in the strict turnstile model must use $\Omega\left(\varepsilon^{-2}\log(N)\log\log(mM)\right)$ bits of space.*

Proof sketch. We present a proof sketch here, with the detailed proof left to the full paper. First we introduce the $\mathrm{GHSE}_{n,k}$ problem, which is a composition of the n-dimensional gap Hamming weight problem $\textsc{Gap-Hamming}_n$ over the results of n copies of k-player $\textsc{Sum-Equal}_k$ instances, i.e., the result of $\mathrm{GHSE}_{n,k}$ is 1 if there are $\geq (1+\varepsilon)n/2$ underlying $\textsc{Sum-Equal}$ instances outputting 1, and 0 if $\leq (1-\varepsilon)n/2$ instances outputting 1.

The hard problem for our lower bound is the augmented index version of $\mathrm{GHSE}_{n,k}$, which we denote by $\textsc{Aug-Index-GHSE}^t_{n,k}$. In particular, $\textsc{Aug-Index-GHSE}^t_{n,k}$ has $t = \Theta(\log n)$ many $\mathrm{GHSE}_{n,k}$ instances embedded, where the last player P_k is given an index $i \in [t]$ together with the results of $\mathrm{GHSE}^{(i+1)}_{n,k}, \ldots, \mathrm{GHSE}^{(t)}_{n,k}$, and P_k is required to output the result of $\mathrm{GHSE}^{(i)}_{n,k}$. Following the reduction in Theorem 4.1 of [2] it suffices to prove our space lower bound in the simultaneous communication model, where each of $P_1, \ldots, P_{k-1}$ sends a single message to the referee P_k.

In the reduction from $\textsc{Aug-Index-GHSE}^t_{n,k}$ to T_ε, the input integers to underlying $\textsc{Sum-Equal}_k$ instances are processed as updates to distinct elements. Furthermore, every $\textsc{Sum-Equal}_k$ instance of $\mathrm{GHSE}^{(j)}_n$ embedded in the $\textsc{Aug-Index-GHSE}^t_{n,k}$ problem is given frequency 100^{j-1}, i.e., is counted as 100^{j-1} distinct elements. Thus the universe has $N := n + 100\cdot n + \cdots + 100^{t-1}\cdot n \leq 100^t n/99$ distinct elements in total, and the final Hamming norm is a weighted sum $F := \sum_{j=1}^t 100^{j-1} f_j$, where f_j is the Hamming weight of $\textsc{Sum-Equal}_k$ instances of $\mathrm{GHSE}^{(j)}_n$ for every $j \in [t]$. An algorithm solving T_ε will give a $(1\pm\varepsilon)$-estimate $\widetilde{F}$ of F, such that $(1-\varepsilon)F \leq \widetilde{F} \leq (1+\varepsilon)F$. From the estimate $\widetilde{F}$ we need to determine the result of $\mathrm{GHSE}^{(i)}_n$ for the given index i. Since the referee can precisely remove the influence of $\mathrm{GHSE}^{(i+1)}_{n,k}, \ldots, \mathrm{GHSE}^{(t)}_{n,k}$ using the auxiliary input before computing $\widetilde{F}$, it suffices to consider the case $i = t$ and the estimation of f_t. Indeed we prove that $\widetilde{F}$ is also a good approximation to $100^{t-1}f_t$ with high probability, as long as the additive error $\sum_{j=1}^{t-1} 100^{j-1} f_j$ is significantly less than the variance of $100^{t-1}f_t$. More specifically,

$$\mathbf{RCC}^{sim}_{k,1/3}(\mathrm{T}_\varepsilon) \geq \mathbf{RCC}^{sim}_{k,0.4}\left(\textsc{Aug-Index-GHSE}^t_{n,k}\right) \tag{1}$$

for our specified input distribution, which induces variance $O(n)$ on every f_j while our gap in advantage is $\Omega(n)$.

Then we prove that the communication cost of solving the augmented index version of t copies of $\mathrm{GHSE}_{n,k}$ is equal to simultaneously solving $\Omega(t)$ many copies.

$$\mathbf{RCC}^{sim}_{k,0.4}\left(\textsc{Aug-Index-GHSE}^t_{n,k}\right) \geq \Omega\left(t\cdot\mathbf{RCC}^{sim}_{k,0.01}(\mathrm{GHSE}_{n,k})\right) \tag{2}$$

The proof relies on the direct sum property of one-way communication for the GHSE problem. The intuition is that all necessary information for computing $\mathrm{GHSE}^{(1)}_{n,k}, \ldots, \mathrm{GHSE}^{(t)}_{n,k}$ must be included in the messages to the referee, since every instance $\mathrm{GHSE}^{(i)}_{n,k}$ can be determined at the referee's position by changing the referee's input alone (without tampering the messages).

Next we prove an $\Omega\left(\varepsilon^{-2}k\log\log k\right)$ lower bound for $\mathbf{RCC}^{sim}_{k,0.1}\left(\mathrm{GHSE}_{n,k}\right)$ and $mM = \mathsf{poly}(k)$. Consider the input $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ to the $\mathrm{GHSE}_{n,k}$ problem, where each player P_j gets $\mathbf{x}_j = \left(\mathbf{x}_j^{(1)}, \ldots, \mathbf{x}_j^{(n)}\right) \in \mathbb{Z}^n$, and for every $i \in [n]$, $Z^{(i)} := \textsc{Sum-Equal}_k\left(\mathbf{x}_1^{(i)}, \ldots, \mathbf{x}_k^{(i)}\right)$ denotes the result of the i-th Sum-Equal instance and the range is $\{\pm 1\}$. Let $\mathrm{HSE}(\mathbf{x}) := \sum_{i=1}^{k} Z^{(i)}$ denote the bias of the underlying vector for the $\textsc{Gap-Hamming}_n$ problem embedded in $\mathrm{GHSE}_{n,k}(\mathbf{x})$. Recall that $\mathrm{GHSE}_{n,k}$ distinguishes $\mathrm{HSE}(\mathbf{x}) \geq \varepsilon n$ and $\mathrm{HSE}(\mathbf{x}) \leq -\varepsilon n$, where the gap becomes $\sqrt{n'}$ for $\mathrm{GHSE}_{n',k}$ and $n' = 1/\varepsilon^2$. With random universal hash functions specified by the public randomness, we prove that

$$\mathbf{RCC}^{sim}_{k,0.01}\left(\mathrm{GHSE}_{n',k}\right) \geq \mathbf{RCC}^{sim}_{k,0.1}\left(\textsc{Aug-Index-Sum-Equal}_k^{n''}\right) \tag{3}$$

where $n'' = \Theta\left(n'\right)$ and $\textsc{Aug-Index-Sum-Equal}_k^{n''}$ is the augmented index version of n'' instances of the $\textsc{Sum-Equal}_k$ problem.

Furthermore, the lower bound holds for a distribution μ over $\mathbb{Z}^{n'\times k}$ such that for $\mathbf{x} \sim \mu$ the conditional expectation satisfies $\mathbf{Var}\left(\mathrm{HSE}(\mathbf{x})\right) \leq n'$, $\mathbf{E}\left[\mathrm{HSE}(\mathbf{x}) \mid \mathrm{GHSE}_{n',k}(\mathbf{x}) = 1\right] = 10\sqrt{n'}$ and $\mathbf{E}\left[\mathrm{HSE}(\mathbf{x}) \mid \mathrm{GHSE}_{n',k}(\mathbf{x}) = 0\right] = -10\sqrt{n'}$. More specifically, let each player specify independent hash functions for every $\textsc{Sum-Equal}_k$ instance, and send the majority of those hash values to the referee. The referee can guess the input and corresponding hash value of any specific $\textsc{Sum-Equal}_k$ instance, such that the conditional distribution of the majority of hash values has a $\Theta\left(1/\sqrt{n''}\right)$ bias under correct guesses. Therefore by taking $n' = \Theta\left(n''\right)$ independent instances of the majority of hash values and conditioned on the correctness of the guesses, the expected number of agreements of the majority and the guessed hash value has a gap of $\Theta\left(n'/\sqrt{n''}\right) = \Theta\left(\sqrt{n'}\right)$, while in both cases the variance is linear in n'. For convenience we shift $\mathrm{HSE}(\mathbf{x})$ to $\pm 10\sqrt{n'}$ by padding and hence the vector of majority instances becomes an input to $\textsc{Gap-Hamming}_{n'}$.

For $\mathbf{RCC}^{sim}_{k,0.1}\left(\textsc{Aug-Index-Sum-Equal}_k^{n''}\right)$, i.e., k-player 0.1-error simultaneous communication complexity of $\textsc{Aug-Index-Sum-Equal}_k^{n''}$, the lower bound follows Theorem 13.

▶ **Theorem 13.** *Let* Π *be an* δ*-error randomized simultaneous communication protocol for* $\textsc{Aug-Index-Sum-Equal}_k^{m'}$*, where* $m' \leq \frac{k\log\log k}{20\log k}$ *and the error tolerance* $\delta < 1/6$*. Then* Π *must have simultaneous communication cost* $\mathbf{RCC}^{sim}_{k,\delta}(\Pi) = \Omega\left(m'k\log\log k\right)$*. Furthermore, the lower bound holds when the inputs to the* $\textsc{Sum-Equal}_k$ *problems are drawn from* $\left([a]^{m'}\right)^{k-1} \times [\pm ka]^{m'}$ *and the sum of inputs to each copy of* $\textsc{Sum-Equal}_k$ *is promised to be* 0 *or* q*, where* $a = O\left(\log k\right)$ *and* $q = 2^{O(a)} \leq k^{1/8}$ *is a multiple of all integers in* $[a]$*.*

Here we present the proof intuition of Theorem 13, while the proof appears in the full paper. Suppose that in a simultaneous communication protocol, a player P_1 encodes multiple instances of $\textsc{Sum-Equal}_k$ independently in a message, say t_1 bits for $\textsc{Sum-Equal}_k^{(1)}$, t_2 bits for $\textsc{Sum-Equal}_k^{(2)}$, and so on. Then many $\textsc{Sum-Equal}_k$ instances will be irrecoverable if the message length $\sum_{i=1}^{m'} t_i$ is significantly less than necessary for handling m' instances in parallel, say $\sum_{i=1}^{m'} t_i \leq 0.1\cdot m'\cdot\mathbf{RCC}^{sim}_{k,\delta}\left(\textsc{Sum-Equal}_k\right)/k$, which means the $\textsc{Aug-Index-Sum-Equal}_k^{m'}$ cannot be solved with small error. Of course the full argument is much more involved, since the information in different Sum-Equal instances can be combined in the message, which we deal with via a dedicated rectangle argument for conditional distributions. Combining (1), (2), (3), and Theorem 13, we get $\mathbf{RCC}^{sim}_{k,1/3}\left(\mathrm{T}_\varepsilon\right) \geq \Omega\left(tn''k\log\log k\right)$. Recalling Theorem 4.1 of [2] and $t = \Theta\left(\log n\right) = \Omega\left(\log N\right)$, $n'' = \Theta\left(n'\right) = \Theta\left(\varepsilon^{-2}\right)$, $mM = \mathsf{poly}(k)$, we conclude that $\overrightarrow{\mathbf{RSC}}_{k,1/3}\left(\mathrm{T}_\varepsilon\right) = \Omega\left(\varepsilon^{-2}\log(N)\log\log(mM)\right)$. ◀

References

1. Farid Ablayev. Lower bounds for one-way probabilistic communication complexity and their application to space complexity. *Theoretical Computer Science*, 157(2):139–159, 1996.
2. Yuqing Ai, Wei Hu, Yi Li, and David P. Woodruff. New Characterizations in Turnstile Streams with Applications. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 20:1–20:22, 2016.
3. Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, June 2004.
4. Paul Beame, Toniann Pitassi, Nathan Segerlind, and Avi Wigderson. A Direct Sum Theorem for Corruption and the Multiparty NOF Communication Complexity of Set Disjointness. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*, pages 52–66, 2005.
5. Mark Braverman and Ankit Garg. Public vs Private Coin in Bounded-Round Information. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 502–513, 2014.
6. Joshua Brody, Harry Buhrman, Michal Koucký, Bruno Loff, Florian Speelman, and Nikolay K. Vereshchagin. Towards a Reverse Newman's Theorem in Interactive Information Complexity. *Algorithmica*, 76(3):749–781, 2016 (also CCC 2013).
7. Graham Cormode, Mayur Datar, Piotr Indyk, and S. Muthukrishnan. Comparing Data Streams Using Hamming Norms (How to Zero In). *IEEE Trans. Knowl. Data Eng.*, 15(3):529–540, 2003.
8. André Gronemeier. Asymptotically Optimal Lower Bounds on the NIH-Multi-Party Information Complexity of the AND-Function and Disjointness. In *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings*, pages 505–516, 2009.
9. Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, 2006.
10. T. S. Jayram. Hellinger strikes back: A note on the multi-party information complexity of AND. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. APPROX '09 / RANDOM '09, Berkeley, CA, USA, August 21 - 23, 2009*, pages 562–573, 2009.
11. Daniel M. Kane, Jelani Nelson, Ely Porat, and David P. Woodruff. Fast moment estimation in data streams in optimal space. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 745–754, 2011.
12. Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 41–52, 2010.
13. Daniel M. Kane, Jelani Nelson, and David P. Woodruff. On the Exact Space Complexity of Sketching and Streaming Small Norms. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1161–1178, 2010.
14. Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. *Computational Complexity*, 8(1):21–49, 1999.
15. Christos H. Papadimitriou and Michael Sipser. Communication complexity. *Journal of Computer and System Sciences*, 28(2):260–269, 1984.
16. Emanuele Viola. The communication complexity of addition. *SODA*, 2013.
17. David P. Woodruff and Qin Zhang. Tight bounds for distributed functional monitoring. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 941–960, 2012.

Construction of Optimal Locally Recoverable Codes and Connection with Hypergraph

Chaoping Xing
School of Electronics, Information and Electrical Engineering, Shanghai Jiaotong University, Shanghai, 200240, P. R. China

Chen Yuan
Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
chenyuan@cwi.nl

Abstract

Locally recoverable codes are a class of block codes with an additional property called locality. A locally recoverable code with locality r can recover a symbol by reading at most r other symbols. Recently, it was discovered by several authors that a q-ary optimal locally recoverable code, i.e., a locally recoverable code achieving the Singleton-type bound, can have length much bigger than $q+1$. In this paper, we present both the upper bound and the lower bound on the length of optimal locally recoverable codes. Our lower bound improves the best known result in [12] for all distance $d \geq 7$. This result is built on the observation of the parity-check matrix equipped with the Vandermonde structure. It turns out that a parity-check matrix with the Vandermonde structure produces an optimal locally recoverable code if it satisfies a certain expansion property for subsets of $\mathbb{F}_q$. To our surprise, this expansion property is then shown to be equivalent to a well-studied problem in extremal graph theory. Our upper bound is derived by an refined analysis of the arguments of Theorem 3.3 in [6].

2012 ACM Subject Classification Theory of computation → Error-correcting codes

Keywords and phrases Locally Repairable Codes, Hypergraph

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.98

Category Track A: Algorithms, Complexity and Games

Funding *Chen Yuan*: This research is supported by the European Union Horizon 2020 research and innovation programme under grant agreement No. 74079 (ALGSTRONGCRYPTO).

Acknowledgements We sincerely thank Prof. J. Verstraëte for his linking our condition (8) with the problem in extremal graph theory. He also provided us some references for latest results on extremal graph theory. We would also like to express our great gratitude to Profs. V. Guruswami, Q. Xiang and M. Lu for discussions and help.

1 Introduction

Motivated by applications in distributed and cloud storage systems, locally recoverable codes have been studied extensively in recent years. Informally speaking, a locally recoverable code (LRC for short) is a block code with an additional property called locality. For a locally recoverable code C of length n, dimension k and locality r, it was shown in [4] that the minimum distance $d(C)$ of C is upper bounded by

$$d(C) \leqslant n - k - \left\lceil \frac{k}{r} \right\rceil + 2. \tag{1}$$

The bound (1) is called the Singleton-type bound for locally recoverable codes. A code achieving the above bound is usually called optimal.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 98; pp. 98:1–98:13


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

1.1 Known results

Construction of optimal locally recoverable codes, i.e., block codes achieving the bound (1) is of both theoretical interest and practical importance. This is a challenging task and has attracted great attention in the last few years. In the literature, there are a few constructions available and some classes of optimal locally recoverable codes are known. A class of codes constructed earlier and known as pyramid codes [8] are shown to be codes that are optimal. In [14], Silberstein *et al* proposed a two-level construction based on the Gabidulin codes combined with a single parity-check $(r+1, r)$ code. Another construction [16] used two layers of MDS codes, a Reed-Solomon code and a special $(r+1, r)$ MDS code. A common shortcoming of these constructions relates to the size of the code alphabet which in all the papers is an exponential function of the code length, complicating the implementation. There was an earlier construction of optimal locally recoverable codes given in [13] with alphabet size comparable to code length. However, the construction in [13] only produces a specific value of the length n, i.e., $n = \lceil \frac{k}{r} \rceil (r+1)$. Thus, the rate of the code is very close to 1. There are also some existence results given in [13] and [15] with less restrictions on the locality r. But both results require large alphabet which is an exponential function of the code length.

A recent breakthrough construction was given in [15]. This construction naturally generalizes Reed-Solomon construction which relies on the alphabet of cardinality comparable to the code length n. The idea behind the construction is very nice. The only shortcoming of this construction is restriction on the locality r. Namely, $r+1$ must be a divisor of either $q-1$ or q, or $r+1$ is equal to a product of a divisor of $q-1$ and a divisor of q for certain q, where q is the code alphabet. This construction was extended via automorphism group of rational function fields by Jin, Ma and Xing [10] and it turns out that there is more flexibility on the locality and the code length can be $q+1$. For some particular locality such as $r = 2, 3, 5, 7, 11$ or 23, it was shown that there exist q-ary optimal locally recoverable codes with length up to $q + 2\sqrt{q}$ via elliptic curves [11]. All these results are aimed at the optimal LRC with large distance.

Unlike classical MDS codes, it is surprising to discover that the optimal LRCs can have super-linear code length in alphabet size q. Barg et.al, [1] gave optimal LRCs by using algebraic surfaces of length $n \approx q^2$ when the distance $d = 3$ and $r \leqslant 4$. This inspired the construction of the optimal LRC with unbounded length and distance $d = 3, 4$ [12]. Furthermore, it was shown in [6] that an optimal LRC with $d \geq 5$ must have length upper bounded in terms of alphabet size q. More precisely, they showed that the length of an optimal q-ary linear LRC with distance $d \geqslant 5$ and locality r is upper bonded by $O\left(dq^{3+\frac{4}{d-4}}\right)$. As for the lower bound, they presented an explicit construction of optimal LRCs with code length $\Omega_r\left(q^{1+\frac{1}{\lfloor (d-3)/2 \rfloor}}\right)$ provided that $d \leq r+2$, where Ω_r means that the implied constant depends on r. One can see that there is still a huge gap between the lower bound and the upper bound. Following this discovery, there are several works dedicated to constructing the maximum length of optimal LRCs. The paper [9] aimed at the optimal LRC with small distance $d = 5$ or 6. In particular, for $d = 6$, the results given in [9] are obtained subject to the constraint that q is even.

1.2 Our results, comparisons and a conjecture

The main result of this paper can be summarized as follows.

▶ **Theorem 1.** *Suppose that $r \geqslant d-2$ and $(r+1)|n$. Then*

(i) *there exists an explicit construction of optimal locally recoverable codes with length $n = q^{2-o(1)}$, minimum distance d and locality r for $d = 7, 8$;*

(ii) *there exists an explicit construction of optimal locally recoverable codes with length* $n = q^{\frac{3}{2}-o(1)}$*, minimum distance* d *and locality* r *for* $d = 9, 10$*;*

(iii) *there exist optimal locally recoverable codes with length* $n = \Omega_{r,d}\left(q(q\log q)^{\frac{1}{\lfloor (d-3)/2 \rfloor}}\right)$*, minimum distance* d *and locality* r *for* $d \geq 11$*; and*

(iv) *there exists an explicit construction of optimal locally recoverable code with length* $n = \Omega_{r,d}\left(q^{1+\frac{1}{\lfloor (d-3)/2 \rfloor}}\right)$*, minimum distance* d *and locality* r *for a constant* $d \geq 11$*. Moreover, the complexity of this construction is upper bounded by* $O(n^d)$*.*

The first three results are derived from extremal graph theory (see Section 5). The last one is derived from the probabilistic arguments (see Section 4).

The first two results improve on the result in [6] which only achieves $n = \Omega(q^{3/2})$ for $d = 7, 8$ and $n = \Omega(q^{4/3})$ for $d = 9, 10$. The third one outperforms the result in [6] by a $(\log q)^{\frac{1}{\lfloor (d-3)/2 \rfloor}}$ multiplicative factor. In addition, for $d = 6$, we are able to remove the constraint required in [9] that q is even.

Although it was proved in [6] that the length of an optimal locally recoverable code is upper bounded by $q^{3+O\left(\frac{1}{d}\right)}$, both the constructions in [6] and this paper show from different angles that the length of an optimal locally recoverable code only achieve $q^{1+O\left(\frac{1}{d}\right)}$. Furthermore, via an upper bound from extremal graph theory, our construction in this paper can achieve at most $O\left(q^{1+\frac{2}{\lfloor (d-1)/2 \rfloor}}\right)$ (see Section 5). Thus, we make the following conjecture.

▶ **Conjecture 2.** *Every optimal locally recoverable code with minimum distance* d *and locality* r *has length upper bounded by* $q^{1+O\left(\frac{1}{d}\right)}$*.*

In addition to the above lower bound on length of optimal locally recoverable codes, we also provide an improved upper bound by refining the analysis of the arguments of Theorem 3.3 in [6].

▶ **Theorem 3** (Informal)**.** *Let* C *be an optimal* $[n, k, d]_q$*-linear locally repairable code with the locality* r*. If* $d \geqslant 5$*, then*

$$n \leq \begin{cases} O(q^3) & \text{if } d \bmod r+1 > 5 \text{ or } d \bmod r+1 < 2, \\ O(q^2) & \text{if } 2 \leq d \bmod r+1 \leq 5. \end{cases} \tag{2}$$

1.3 Our techniques

For minimum distance $d \geq 7$, the only optimal locally recoverable code with super-linear code length was given in [6]. In this paper, we present another construction for optimal LRCs for $d \geqslant 5$. Our idea comes from generalized Reed-Solomon codes where parity-check matrices have the Vandermonde structure. This idea was already employed in [9] for $d = 5, 6$. Similar to [9], we divide a parity-check matrix into disjoint blocks, each block with $r + 1$ columns. We require that each block of this matrix has a Vandermonde matrix structure. In order that the parity-check matrix with this structure produces an optimal locally recoverable code, elements in these blocks must satisfy certain expansion property. This property allows us to relate optimality of a locally recoverable code to a well-studied problem in extremal graph theory. With the help of extremal graph theory, we succeed to improve all of the best known results in [6] for $d \geq 7$.

Furthermore, by a random or probabilistic argument, we show an existence result. Moreover, for constant d the probabilistic method for the existence result can be converted into a deterministic algorithm via method of conditional probabilities. Thus, we obtain an algorithmic construction in polynomial time, i.e., Theorem 1(iv). The result of Theorem 1(iv) matches the result given in [6]. However, our parity-check matrix is more structured and this may lead to some other applications.

1.4 Organization

The paper is organized as follows. In Section 2, we briefly introduce locally recoverable codes and some basic notations on graph theory. Section 3 presents a necessary and sufficient condition for which a Vandermonde-type parity-check matrix produces an optimal locally recoverable code in terms of certain expansion property for subsets of $\mathbb{F}_q$. In Section 4, we first show an existence result via a probabilistic method. Then this probabilistic method is converted into an algorithmic construction in polynomial time. In Section 5, we show that the necessary and sufficient condition derived in Section 2 is equivalent to a central problem in extremal graph theory. By applying the known results from extremal graph theory, we obtain the desired results. Finally, in Section 6, we prove a general upper bound on the optimal LRC.

2 Preliminaries

2.1 Locally recoverable codes

Let q be a prime power and $\mathbb{F}_q$ be the finite field with q elements and denote by $[n]$ the set $\{1, 2, \ldots, n\}$. In this paper, we consider linear locally recoverable codes only. An $[n, k, d]$ linear code C is a k-dimensional subspace of $\mathbb{F}_q^n$ with minimum (Hamming) distance d. The (Euclidean) dual code of C, denoted by $C^\perp$, is defined by $C^\perp = \{\mathbf{b} \in \mathbb{F}_q^n : \ \mathbf{c} \cdot \mathbf{b} = 0 \text{ for all } \mathbf{c} \in C\}$, where $\mathbf{c} \cdot \mathbf{b}$ denotes the standard inner product of the two vectors $\mathbf{b}$ and $\mathbf{c}$.

Informally speaking, a block code is said to have locality r if every coordinate of a given codeword can be recovered by accessing at most r other coordinates of this codeword. There are several equivalent definitions of locally recoverable codes. A formal definition of a locally recoverable code with locality r is given as follows.

▶ **Definition 4.** A q-ary block code C of length n is called a locally recoverable code or locally repairable code (LRC for short) with locality r if for any $i \in [n]$, there exists a subset $R_i \subseteq [n] \setminus \{i\}$ of size r such that for any $\mathbf{c} = (c_1, \ldots, c_n) \in C$, c_i can be recovered by $\{c_j\}_{j \in R_i}$, i.e., for any $i \in [n]$, there exists a subset $R_i \subseteq [n] \setminus \{i\}$ of size r such that for any $\mathbf{u}, \mathbf{v} \in C$, $\mathbf{u}_{R_i \cup \{i\}} = \mathbf{v}_{R_i \cup \{i\}}$ if and only if $\mathbf{u}_{R_i} = \mathbf{v}_{R_i}$. The set R_i is called a recovering set of i.

In literature, there are various definitions for locally recoverable code and all of them are equivalent. For example, we have the following two definitions that are equivalent to Definition 4. For the sake of completeness, we give a proof.

▶ **Lemma 5.** *A q-ary code C of length n is a locally recoverable code if and only if one of the followings holds.*

- **(i)** *For any $i \in [n]$, there exists a subset $R_i \subseteq [n] \setminus \{i\}$ of size r such that position i of every codeword $\mathbf{c} \in C$ is determined by $\mathbf{c}_{R_i}$, i.e, there is a function $f_i(x_1, \ldots, x_r)$ (independent of $\mathbf{c}$ and only dependent on i) such that $c_i = f_i(\mathbf{c}_{R_i})$, where $\mathbf{c}_{R_i}$ stands for the projection of $\mathbf{c}$ at R_i.*
- **(ii)** *For any $i \in [n]$, there exists a subset $R_i \subseteq [n] \setminus \{i\}$ of size r such that*

$$C_{R_i}(i, \alpha) \cap C_{R_i}(i, \beta) = \emptyset$$

for any $\alpha \neq \beta \in \mathbb{F}_q$, where $C(i, \alpha) = \{\mathbf{c} \in C : \ c_i = \alpha\}$ and $C_{R_i}(i, \alpha)$ denotes the projection of $C(i, \alpha)$ on R_i.

The Singleton (upper) bound in (1) is given in terms of minimum distance d. We can also rewrite this bound in terms of dimension k.

▶ **Lemma 6.** *Let n, k, d, r be positive integers with $(r+1)|n$. If the Singleton-type bound* (1) *is achieved, then*

$$n - k = \frac{n}{r+1} + d - 2 - \left\lfloor \frac{d-2}{r+1} \right\rfloor. \tag{3}$$

Conversely, if $d - 2 \not\equiv r \pmod{r+1}$ and the equlity (3) *is satisfied, then the Singleton-type bound* (1) *is achieved.*

The proof is straightforward and can be found in [6].

▶ Remark 7. If $d - 2 \equiv r \pmod{r+1}$, one can verify that (3) implies that $r|k$. In this case, by [4, Corollary 10] one cannot achieve the Singleton-type bound (1) with equality and one must have $d \leqslant n - k - \left\lceil \frac{k}{r} \right\rceil + 1$. Therefore in this case we say an LRC attaining this latter bound as optimal.

▶ **Corollary 8.** *If $r \geqslant d-2$, then an $[n, k, d]$ locally recoverable code with locality r is optimal if*

$$n - k - \frac{n}{r+1} = d - 2. \tag{4}$$

Proof. As $r \geqslant d - 2$, $\left\lfloor \frac{d-2}{r+1} \right\rfloor = 0$. Hence, (3) and (4) are equivalent. ◀

The locality of a locally recoverable code C can be determined by a parity-check matrix of C as follows. Assume that $(r+1)|n$. Let $m = \frac{n}{r+1}$ and let D_i be $(n-k-m) \times (r+1)$ matrices. Put

$$H = \left(\begin{array}{c|c|c|c} \mathbf{1} & \mathbf{0} & \cdots & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{1} & \cdots & \mathbf{0} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \mathbf{0} & \mathbf{0} & \cdots & \mathbf{1} \\ \hline D_1 & D_2 & \cdots & D_m \end{array} \right), \tag{5}$$

where $\mathbf{1}$ and $\mathbf{0}$ stand for the all-one row vector and the zero row vector of length $r+1$, respectively. Let C be the code with H as a parity-check matrix. Then it is clear that the dimension of C is at least k. Furthermore, we claim that the locality of C is r. Indeed, let $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ be a codeword of C, then $\sum_{j=1+(r+1)i}^{(r+1)(i+1)} c_j = 0$ for $0 \leqslant i \leqslant m-1$ as $H\mathbf{c}^T = \mathbf{0}$. Hence, a coordinate c_j with $j \in \{1 + (r+1)i, \ldots, (r+1)(i+1)\}$ for some $0 \leqslant i \leqslant m-1$ can be repaired by $\mathbf{c}_{R_j}$ with $R_j = \{1 + (r+1)i, \ldots, (r+1)(i+1)\} \setminus \{j\}$.

In conclusion, to see if a linear code C with a parity-check matrix H of the form (5) is an optimal locally recoverable code, it is sufficient to check if the minimum distance of C satisfies (4) for $r \geqslant d - 2$.

2.2 Graphs

A undirected graph G is a pair $G = (V, E)$, where V is a finite set and E is a set consisting of some subsets of size 2 of V. An element of V is called a vertex and an element of E is called an edge. A subgraph G' of a graph G is a graph whose vertex set and edge set are subsets of those of G. We say that G has a cycle $(v_1, \ldots, v_m)$ if $\{v_i, v_{i+1}\} \in E$ for $i = 1, \ldots, m-1$ and $\{v_m, v_1\} \in E$. The following Lemma 9 provides a simple but useful way to determine if G contains a cycle. The proof can be found in any textbook about graph theory (see [3] for instance).

▶ **Lemma 9.** *An undirected graph G contains a cycle if $|E| \geq |V|$.*

Apart from the above usual definition of graphs, we also require some results on hypergraph in this paper. A hypergraph is a generalization of a graph in which an edge can join any number of vertices. Formally, a hypergraph H is a pair $H = (X, E)$ where X is a set of elements called vertices, and E is a set of non-empty subsets of X called hyperedges or edges. Therefore, E is a subset of $2^X \setminus \{\emptyset\}$, where 2^X stands for the power set of X.

▶ **Definition 10** (r-uniform Hypergraph (or r-graph for short))**.** *A hypergraph $H = (X, E)$ is called r-uniform if every hyperedge in E has size r. In other words, every hyperedge of an r-uniform hypergraph connects exactly r vertices.*

There are several ways to define cycles in a hypergraph that coincide with the definition of cycles in the usual graph. In this paper, we use the Berge cycle as the generalization of cycles in the usual graph.

▶ **Definition 11** (Berge cycle)**.** A r-uniform hypergraph $H = (X, E)$ contains a Berge k-cycle $(v_1, \ldots, v_k)$ if there exist k hyperedges $e_1, \ldots, e_k \in E$ such that $\{v_{i-1}, v_i\} \subseteq e_i$ for $i = 2, \ldots, k$ and $\{v_1, v_k\} \subseteq e_1$.

3 A criterion on minimum distance

It follows from Corollary 8 that for $d \leq r + 2$, a locally recoverable code with parity-check matrix H in (5) is optimal provided that any $d - 1$ columns of H are linearly independent and each D_i is a $(d - 2) \times (r + 1)$ matrix.

Let $\mathbb{F}_q$ be a finite field and put $m = \frac{n}{r+1}$. Assume that $A_1, \ldots, A_m$ are subsets of $\mathbb{F}_q$, each of size $r + 1$. Let $A_i = \{a_{i,1}, \ldots, a_{i,r+1}\}$ for $i = 1, \ldots, m$. Let $\mathbf{a}_{i,j} = (a_{i,j}, a_{i,j}^2, \ldots, a_{i,j}^{d-2})$ and put $D_i = (\mathbf{a}_{i,1}^T, \mathbf{a}_{i,2}^T, \ldots, \mathbf{a}_{i,r+1}^T)$. Thus, D_i is a Vandermonde-type matrix. Let $\mathbf{e}_1, \ldots, \mathbf{e}_m$ be the standard basis of vector space $\mathbb{F}_q^m$, i.e., all components of $\mathbf{e}_i$ are 0 except that the i-th component is 1. Then, we can rewrite H as follow.

$$H = \begin{pmatrix} \mathbf{e}_1^T & \cdots & \mathbf{e}_1^T & \cdots & \mathbf{e}_m^T & \cdots & \mathbf{e}_m^T \\ \mathbf{a}_{1,1}^T & \cdots & \mathbf{a}_{1,r+1}^T & \cdots & \mathbf{a}_{m,1}^T & \cdots & \mathbf{a}_{m,r+1}^T \end{pmatrix}. \tag{6}$$

We now present a sufficient and necessary condition under which any $d - 1$ columns of the matrix H in (6) are linearly independent.

▶ **Theorem 12.** *For $d \geqslant 5$, then any $d-1$ columns of H defined in* (6) *are linearly independent if and only if $|\bigcup_{i \in S} A_i| \geq r|S| + 1$ for any $S \subseteq [m]$ of size no more than $t = \lfloor \frac{d-1}{2} \rfloor$.*

Proof. We first prove the "if" direction. Let $\mathbf{h}_{i,j}$ be the (i, j)th column of H, i.e., $\mathbf{h}_{i,j} = (\mathbf{e}_i, \mathbf{a}_{i,j})^T$ for $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant r+1$. Choose any $d-1$ columns $\{\mathbf{h}_{i,j}\}_{1 \leqslant i \leqslant m; j \in S_i}$ of H, where S_i are subsets of $[r+1]$ satisfying $\sum_{i=1}^m |S_i| = d-1$. Let $A_i' = \{a_{i,j} \in A_i : j \in S_i\}$, i.e., A_i' is the subset of A_i where each element is associated with one of the $d-1$ columns. Let H' be the $(n - k - m) \times (d - 1)$ matrix consisting of these $d - 1$ columns. We are going to show that H' has rank $d - 1$. We assume that S_i is either empty or of size at least 2. Otherwise, the unique column selected from D_i with $|S_i| = 1$ must be linearly independent from the rest $d - 2$ columns. We can consider the linear independence of the rest $d - 2$ columns instead. Now, we assume that there are at most t non-empty sets S_i. Let $A = \{a_{i,j}\}_{1 \leqslant i \leqslant m; j \in S_i}$.

Assume that $A = \{a_1, \ldots, a_s\}$ has s distinct elements. If $s = d-1$, then by elementary row operations, one can find a $(d-1) \times (d-1)$ Vandermonde submatrix of the form

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ \mathbf{a}_1^T & \mathbf{a}_2^T & \cdots & \mathbf{a}_{d-1}^T \end{pmatrix}$$

of H', where $\mathbf{a}_i = (a_i, a_i^2, \ldots, a_i^{d-2})$. Thus, the rank of H' is $d-1$.

We proceed to the case where $s < d-1$. By permuting the columns of H', we obtain a matrix of the following form:

$$H_1 = \left(\begin{array}{cccc|ccc} \mathbf{e}_{i_1}^T & \mathbf{e}_{i_2}^T & \cdots & \mathbf{e}_{i_s}^T & \mathbf{e}_{i_{s+1}}^T & \cdots & \mathbf{e}_{i_{d-1}}^T \\ \mathbf{a}_1^T & \mathbf{a}_2^T & \cdots & \mathbf{a}_s^T & \mathbf{a}_{s+1}^T & \cdots & \mathbf{a}_{d-1}^T \end{array} \right),$$

where $1 \leqslant i_1 \leqslant i_2 \leqslant \cdots \leqslant i_{d-1} \leqslant m$ and $\{a_{s+1}, \ldots, a_{d-1}\}$ is a subset of A. Thus, a_j belongs to A_{i_j} for $1 \leqslant j \leqslant d-1$. By elementary column operations, we can erase $\mathbf{a}_{s+i}^T$ since it also appears in one of the first s columns. Hence, H_1 is equivalent to

$$H_2 = \left(\begin{array}{cccc|ccc} \mathbf{e}_{i_1}^T & \mathbf{e}_{i_2}^T & \cdots & \mathbf{e}_{i_s}^T & \mathbf{e}_{i_{s+1}}^T - \mathbf{e}_{k_{s+1}}^T & \cdots & \mathbf{e}_{i_{d-1}}^T - \mathbf{e}_{k_{d-1}}^T \\ \mathbf{a}_1^T & \mathbf{a}_2^T & \cdots & \mathbf{a}_s^T & \mathbf{0}^T & \cdots & \mathbf{0}^T \end{array} \right),$$

where $\{k_{s+1}, \ldots, k_{d-1}\}$ is a subset of $\{i_1, \ldots, i_s\}$. Since H_2 is an upper left triangular block matrix, showing that H_2 is a full-rank matrix is equivalent to showing both $(\mathbf{a}_1^T, \mathbf{a}_2^T, \ldots, \mathbf{a}_s^T)$ and $(\mathbf{e}_{i_{s+1}}^T - \mathbf{e}_{k_{s+1}}^T, \cdots, \mathbf{e}_{i_{d-1}}^T - \mathbf{e}_{k_{d-1}}^T)$ have full rank. Note that $(\mathbf{a}_1^T, \mathbf{a}_2^T, \ldots, \mathbf{a}_s^T)$ is a $(d-2) \times s$ Vandermonde matrix and hence it has full rank s. It remains to show that $\mathbf{e}_{i_{s+1}} - \mathbf{e}_{k_{s+1}}, \ldots, \mathbf{e}_{i_{d-1}} - \mathbf{e}_{k_{d-1}}$ are linearly independent. Suppose they were linearly dependent. Then there exist elements $\lambda_{s+1}, \ldots, \lambda_{d-1} \in \mathbb{F}_q$ which are not all zero such that

$$\sum_{j=s+1}^{d-1} \lambda_j (\mathbf{e}_{i_j} - \mathbf{e}_{k_j}) = 0.$$

Let P be the subset of $\{s+1, \ldots, d-1\}$ such that $\lambda_i \neq 0$ if and only if $i \in P$. It follows that

$$\sum_{i \in P} \lambda_i (\mathbf{e}_{j_i} - \mathbf{e}_{k_i}) = 0. \tag{7}$$

Let $U = \{j_i : i \in P\}$, $V = \{k_i : i \in P\}$ and $W = U \cup V$. As both U and V are subsets of $\{i \in [m] : |S_i| \geqslant 2\}$, we have $|W| \leqslant t = \lfloor \frac{d-1}{2} \rfloor$. Since λ_i is nonzero for all $i \in P$, every $\ell \in W$ must appear at least twice in the multiset consisting of elements of U and V. Otherwise, $\mathbf{e}_\ell$ could not be cancelled in (7). This implies $|W| \leq |P|$.

On the other hand, for each $a_i \in A$, there is exactly one subset A'_{k_i} containing a_i since the first s columns have s distinct $\mathbf{a}_i$. Furthermore, let $t_i = |\{\ell \in U : a_i \in A'_\ell\}|$. It follows that $\sum_{a_i \in A} t_i = |P|$ and a_i belongs to $t_i + 1$ subsets in $\{A'_\ell : \ell \in W\}$. This implies

$$\left| \bigcup_{\ell \in W} A_\ell \right| \leq \sum_{\ell \in W} |A_\ell| - \sum_{i=1}^{s} t_i = (r+1)|W| - |P|$$

since $A'_\ell \subseteq A_\ell$. Combining with the condition $|\bigcup_{\ell \in W} A_\ell| \geq r|W| + 1$ forces $|W| \geq |P| + 1$. A contradiction occurs and we complete the proof of the "if" direction.

We proceed to the "only if" direction. First, we claim that $|A_i \cap A_j| \leq 1$ for any $i \neq j$. Otherwise, we may assume that $A_i \cap A_j$ contains two distinct elements a_1 and a_2. Thus, H contains the four linearly dependent columns $(\mathbf{e}_i, \mathbf{a}_1)^T, (\mathbf{e}_i, \mathbf{a}_2)^T, (\mathbf{e}_j, \mathbf{a}_1)^T$ and $(\mathbf{e}_j, \mathbf{a}_2)^T$.

We prove the "only if" part by contradiction. Without loss of generality, we assume that the first s subsets $A_1, \ldots, A_s$ do not satisfy the condition, i.e. $|\bigcup_{i=1}^{s} A_i| \leq sr$, where s satisfies $s \leqslant t$. Define an undirected graph $G = ([s], E)$ such that $\{i, j\} \in E$ if and only if $A_i \cap A_j \neq \emptyset$. By inclusion-exclusion principle, we have

$$rs \geqslant \left|\bigcup_{i=1}^{s} A_i\right| \geq \sum_{i=1}^{s} |A_i| - \sum_{(i,j)\in E} 1 = s(r+1) - |E|.$$

This implies $|E| \geq s$. By Lemma 9, there exists a cycle in this undirected graph. Without loss of generality, we may assume that $(1, \ldots, \ell)$ is a cycle, i.e., $\{i, i+1\} \in E$ for $i = 1, \ldots, \ell - 1$ and $\{\ell, 1\} \in E$. By the definition of E, A_i and A_{i+1} contains a common element $\{a_{j_i}\}$. Then, we can pick two columns $(\mathbf{e}_i, \mathbf{a}_{j_{i-1}})^T$ [1] and $(\mathbf{e}_i, \mathbf{a}_{j_i})^T$ from the i-th block D_i for $i = 1, \ldots, \ell$. These 2ℓ columns are linearly dependent since

$$\sum_{i=1}^{\ell} \Big((\mathbf{e}_i, \mathbf{a}_{j_{i-1}}) - (\mathbf{e}_i, \mathbf{a}_{j_i}) \Big) = \sum_{i=1}^{\ell} (0, \mathbf{a}_{j_{i-1}} - \mathbf{a}_{j_i}) = \mathbf{0}.$$

The proof is completed. ◀

By Theorem 12, we immediately obtain the following result.

▶ **Theorem 13.** *If $t = \lfloor \frac{d-1}{2} \rfloor \geqslant 2$ and $(r+1)|n$, then there exists a q-ary optimal linear LRC with length n, minimum distance d and locality r provided that there are $m = \frac{n}{r+1}$ sets $A_1, \ldots, A_m \subseteq \mathbb{F}_q$ such that*

$$\begin{array}{ll} |A_i| = r+1 & \text{for } 1 \leq i \leq m, \\ |\bigcup_{i\in S} A_i| \geq |S|r + 1 & \text{for any } S \subseteq [m] \text{ of size at most } t. \end{array} \tag{8}$$

▶ Remark 14. We point out that there is another way to look at (8) as one reviewer suggests. Define an unbalanced bipartite expander graph where the vertex u_i on the left hand side are associated with sets A_i and the vertex v_j on the right hand side are associated with an element a_j in $\mathbb{F}_q$. u_i and v_j are adjacent if and only if a_j is contained in A_i. The expansion property (8) is now equivalent to the existence of good unbalanced bipartite graph [5]. However, the parameters discussed in this paper are not in the scope of explicit construction of good unbalanced bipartite graph, i.e., the expansion property in (8) is too strong for all known explicit constructions.

4 Random and algorithmic constructions

In the previous section, we converted the construction of optimal LRCs into a problem of finding subsets of $\mathbb{F}_q$ satisfying (8). In this section, we first present a probabilistic construction of subsets satisfying (8). In addition, we can derandomize this probabilistic construction into a deterministic construction in polynomial time if d is constant. The case $t = 2$, i.e., $d = 5$ and 6, is equivalent to the design of constant weight codes [9]. In this section, we assume $t \geq 3$. Since the algebraic structure is not important for the union of set, we replace $\mathbb{F}_q$ with $[q]$ from now on.

▶ **Theorem 15.** *There exist $m = \left\lceil \frac{q^{1+\frac{1}{t-1}}}{2t^2(r+1)^{2+\frac{2}{t-1}}} \right\rceil$ sets $A_1, \ldots, A_m$ satisfying (8) provided q is large enough.*

[1] Define $\mathbf{a}_{j_0} = \mathbf{a}_{j_\ell}$ for simplicity.

Proof. Let $X_i = \{x_{i,1}, \ldots, x_{i,r+1}\}, i = 1, \ldots, 2m$ be the set picked uniformly at random over all $(r+1)$-sized subsets of $[q]$. Define the binary random variable Y_S such that $Y_S = 1$ if $|\bigcup_{i \in S} X_i| \leq |S|r$ and 0 otherwise. Our goal is to bound the expectation $E\left[\sum_{S \subseteq [2m], |S| \leq t} Y_S\right]$. Without loss of generality, we may assume that $S = \{1, \ldots, a\}$ for some $1 < a \leq t$. We order the random variables in $X_i, i = 1, \ldots, a$, i.e., $x_{1,1}, \ldots, x_{1,r+1}, \ldots, x_{a,1}, \ldots, x_{a,r+1}$. We want to bound the probability of the event $Y_S = 1$, i.e., at least a elements repeated in this sequence. Given an element $x_{i,j}$, the probability that $x_{i,j} \neq x_{i',j'}$ for some $x_{i',j'}$ prior to $x_{i,j}$ is at least $1 - \frac{(i-1)(r+1)+j}{q} \geq 1 - \frac{a(r+1)}{q}$. Taking over all sets of size at least a in this sequence, the probability of $Y_S = 1$ is at most

$$\sum_{i=a}^{a(r+1)} \binom{a(r+1)}{i} \left(\frac{a(r+1)}{q}\right)^i \leq \sum_{i=a}^{a(r+1)} \frac{\left(a(r+1)\right)^i}{i!} \left(\frac{a(r+1)}{q}\right)^i \leq \frac{1.1}{a!} \left(\frac{a^2(r+1)^2}{q}\right)^a.$$

for $q \geq 10a^2(r+1)^2$. It follows that

$$\begin{aligned} E\left[\sum_{S \subset [2m], |S| \leq t} Y_S\right] &= \sum_{i=2}^{t} \sum_{S \subset [2m], |S|=i} \Pr[Y_S = 1] \leq \sum_{i=2}^{t} \binom{2m}{i} \frac{1.1}{i!} \left(\frac{i^2(r+1)^2}{q}\right)^i \\ &\leq \sum_{i=2}^{t} 1.1 (\frac{1}{i!})^2 \left(\frac{2mi^2(r+1)^2}{q}\right)^i \leq \sum_{i=2}^{t} 1.1 \left(\frac{1}{i!}\right)^2 \left(\frac{q}{(r+1)^2}\right)^{\frac{i}{t-1}} \\ &\leq 1.1 \times 1.5 \left(\frac{1}{t!}\right)^2 \left(\frac{q}{(r+1)^2}\right)^{\frac{t}{t-1}} \leq \frac{2}{4t^2} \left(\frac{q}{(r+1)^2}\right)^{\frac{t}{t-1}} \leq m. \end{aligned}$$

for $q \geq t^{2t} 3^t (r+1)$ and $t \geq 3$. The second inequality is due to $\binom{2m}{i} \leq \frac{(2m)^i}{i!}$ and the third inequality is due to

$$\left(\frac{1}{i!}\right)^2 \left(\frac{q}{(r+1)^2}\right)^{\frac{i}{t-1}} \geq 3 \left(\frac{1}{(i-1)!}\right)^2 \left(\frac{q}{(r+1)^2}\right)^{\frac{i-1}{t-1}}.$$

That means there exists $2m$ $(r+1)$-sized sets $A_1, \ldots, A_{2m}$ such that there are at most m subsets $S \subseteq [2m]$ with $|\bigcup_{i \in S} A_i| \leq |S|r$. For each of these m subsets S, remove one set from $A_i, i \in S$. The desired result follows as we remove at most m sets. ◀

Theorem 15 is an existence proof. However, if t is a constant, it is possible to turn this argument into an algorithm via the method of conditional probabilities.

▶ **Theorem 16.** *There exists a polynomial-time deterministic algorithm to find m sets in Theorem 15 provided that t is a constant.*

Proof. We follow the same notation in Theorem 15. Let $X_i = \{x_{i,1}, \ldots, x_{i,r+1}\}$ be a random set of size $r+1$. Our goal is to minimize $E[\sum_{S \subset [2m], |S| \leq t} Y_S]$ by fixing the set X_i one by one. Since

$$\begin{aligned} E\left[\sum_{S \subseteq [2m], |S| \leq t} Y_S\right] &= \sum_{A \subset [q], |A|=r+1} E\left[\sum_{S \subseteq [2m], |S| \leq t} Y_S | X_1 = A\right] \Pr[X_1 = A] \\ &= \frac{1}{\binom{q}{r+1}} \sum_{A \subset [q], |A|=r+1} E\left[\sum_{S \subseteq [2m], |S| \leq t} Y_S | X_1 = A\right], \end{aligned}$$

there exists a set A such that $E\left[\sum_{S \subseteq [2m], |S| \leq t} Y_S | X_1 = A\right] \leq E\left[\sum_{S \subseteq [2m], |S| \leq t} Y_S\right]$. If $r+1$ is a constant, we only need to enumerate all subsets of size $r+1$ in polynomial time. However, if $r+1$ is not a constant, we enumerate $x_{1,1} \in X_1$ instead of the whole set, i.e., minimizing

$E\left[\sum_{S\subseteq[2m],|S|\leq t} Y_S|x_{1,1}=a_{1,1}\right]$ for $a_{1,1}\in[q]$. Given a subset $S\subseteq[2m]$ of size t, let us show how to compute $E[Y_S|x_{1,1}=a_{1,1}]$. Without loss of generality, we assume $S=\{1,\ldots,t\}$. We list $t(r+1)$ random elements $x_{1,1}=a_{1,1},x_{1,2},\ldots,x_{1,r+1},\ldots,x_{t,1},\ldots,x_{t,r+1}$. For large enough q, it suffices to approximate $E[Y_S|x_{1,1}=a_{1,1}]$ by counting the number of sequences where there are exact t repetitions. There are $\binom{(r+1)t}{t}$ combinations of these t positions. Let $R\subseteq[t]\times[r+1]$ be any set of size t representing the t positions. we first remove these t positions from the sequence. The remaining tr positions in the sequence must have distinct elements and there are $\prod_{i=0}^{rt-1}(q-i)$ ways to pick these tr elements. Assume that we assign $1,\ldots,rt$ to these rt positions. To obtain our final result, we multiply it by $\prod_{i=0}^{rt-1}(q-i)$. It remains to fill our sequence by adding back the t positions in R. For each $(i,j)\in R$, we enumerate all possible choices of $x_{i,j},(i,j)\in R$ and find out the number of combinations that there are exact t repetitions in the resulting sequence. There are at most q^t ways to do the enumeration. Then, we obtain the exact value of $E[Y_S|x_{1,1}=a_{1,1}]$. Observe that there are at most $\sum_{i=2}^{t}\binom{n}{i}$ subsets S. Thus, this expectation can be computed in polynomial time as t is a constant. We do it $r+1$ times so as to fix all elements in X_1. Given $A_1,\ldots,A_k$, our goal is to find $X_{k+1}=A_{k+1}$ to minimize the expectation

$$E\left[\sum_{S\subseteq[2m],|S|\leq t} Y_S|X_1=A_1,\ldots,X_k=A_k\right]\leq E\left[\sum_{S\subseteq[2m],|S|\leq t} Y_S\right].$$

It can be done in the same way as $X_1,\ldots,X_{k-1}$ are already fixed. After we fix all these $2m$ sets, we will obtain $A_1,\ldots,A_{2m}$ with the same property as Theorem 15 claims. Then, we enumerate all t-sized subsets $S\subseteq[q]$ and do the same as Theorem 15 does. The resulting subsets are the output of our algorithm. The number of these subsets is at least m. Since t is constant, all this operation is done in polynomial time. The proof is completed. ◀

The following is a direct consequence of Theorem 12, Theorem 15 and Theorem 16.

▶ **Theorem 17.** *For $d\geqslant 5$, put $t=\left\lfloor\frac{d-1}{2}\right\rfloor$. If $r\geqslant d-2$, $(r+1)|n$ and q is sufficiently large, then there exists a q-ary $[n,k,d]$ optimal locally recoverable code with locality r and $n\geq\frac{q^{1+\frac{1}{t-1}}}{2t^2(r+1)^{1+\frac{2}{t-1}}}$. The parity check matrix of this code has the form of (6). Moreover, if d is a constant, there exists a deterministic algorithm running in polynomial time to construct this code.*

5 The connection with extremal graph theory

To our surprise, it turns out that finding a collection of sets satisfying (8) is equivalent to constructing an $(r+1)$-uniform hypergraph avoiding short Berge cycle. The latter is one of the central problems in extremal graph theory and this problem is extremely difficult.

▶ **Lemma 18.** *There exist m sets satisfying (8) if and only if there exists an $(r+1)$-hypergraph $H=([q],E)$ with $|E|=m$ that does not have any Berge ℓ-cycles for all $\ell\leq t$.*

Proof. To see the equivalence of these two problems, we define an $(r+1)$-hypergraph as follows: Let $H=(V,E)$ with $V=[q]$ and $E=\{A_1,\ldots,A_m\}$. It is clear that H is an $(r+1)$-hypergraph. Assume that there exists $k\leq t$ subsets $A_{i_1},\ldots,A_{i_k}$ does not satisfy the condition that $|\bigcup_{j=1}^{k}A_{i_j}|\geq rk+1$. The same argument in Theorem 12 implies that there exists a cycle $(1,2,\ldots,\ell)$ such that $j\in A_{i_j}\cap A_{i_{j+1}}$. That means $\{j-1,j\}\subseteq A_{i_j}$ for $j=2,\ldots,\ell$ and $\{1,\ell\}\subseteq A_{i_1}$. By the definition of Berge cycle, the $(r+1)$-hypergraph

H contains this Berge ℓ-cycle $(1, 2, \ldots, \ell)$. On the other hand, assume that there exists a Berge ℓ-cycle in H. Denote the ℓ edges of this cycle $A_{i_1}, \ldots, A_{i_\ell}$. The results follows since $|A_{i_j} \cap A_{i_{j+1}}| \geq 1$ for $i = 1, \ldots, \ell - 1$ and $|A_{i_1} \cap A_{i_\ell}| \geq 1$. ◀

The equivalence of both the problems allow us to make use of known results in this area. Let $\mathcal{F}$ be a family of $r+1$-graph. Denote by $ex_{r+1}(n, \mathcal{F})$ the maximum number of edges in an $(r+1)$-hypergraph that does not contain any subgraphs in $\mathcal{F}$. Denote by BC_k the set of k-cycles. Let $\mathcal{B}_k = \{BC_2, \ldots, BC_k\}$. One upper bound on $ex_{r+1}(n, \mathcal{B}_t)$ is obtained by reducing this problem to an $m \times n$ bipartite graph with girth more than $2t$ and apply the result in [7].

▶ **Proposition 19** ([19]). *$ex_{r+1}(n, \mathcal{B}_t)$ is upper bounded by*

(i) $\frac{n}{r}(\frac{n}{r+1})^{\frac{2}{t-1}} + \frac{n}{r+1}$ *if t is odd,*

(ii) $\frac{n}{r(r+1)}n^{\frac{2}{t}} + \frac{n}{r+1}$ *if t is even.*

Since these two problems are equivalent, Proposition 19 gives an upper bound on the number m of sets A_i. For $t = 3, 4$, the following two propositions show that this upper bound is asymptotically tight. However, constructing such hypergraph requires sophisticated knowledge in this area which is beyond the scope of this paper. We summarize the results as follows.

▶ **Proposition 20** ([18]). *There exists explicit construction of $(r+1)$-hypergraph $H = ([q], E)$ with $|E| = q^{2-o(1)}$ that contains no subgraph in $\mathcal{B}_3$.*

▶ **Proposition 21** ([17]). *There exists explicit construction of $(r+1)$-hypergraph $H = ([q], E)$ with $|E| = \Theta(q^{\frac{3}{2}-o(1)})$ that contains no subgraph in $\mathcal{B}_4$.*

Determining the exact value of $ex_{r+1}(n, \mathcal{B}_t)$ for $r \geq 2$ and $t \geq 3$ is extremely difficult. A major open problem in this area is whether $ex_{r+1}(n, \mathcal{B}_t) = \Omega(n^{1+\frac{2}{t}})$. A tighter lower bound for general t can be obtained from H-free random process [2]. The method in [2] can also be applied to hypergraph and add a log factor above the probabilistic method in Theorem 15. Again this technique is beyond our scope.

▶ **Proposition 22** ([18]). *$ex_{r+1}(n, \mathcal{B}_t) = \Omega_{r,t}(n(n \log n)^{\frac{1}{t-1}})$.*

Theorem 1 summarizes all above results in the language of codes.

6 An upper bound on the length of optimal LRC

In this section, we derive an upper bound on the length of optimal LRC. Our upper bound holds for all optimal LRC. The proof of our upper bound is a refined analysis of the argument of Theorem 3.3 in [6]. Recall the following Lemma in [6].

▶ **Lemma 23** (Lemma 3.1 [6]). *Let C be an $[n, k, d]_q$ linear optimal LRC with locality r. Then, there exist $\frac{n}{r+1}$ disjoint recovery sets, each of size $r+1$ provided that*

$$\frac{n}{r+1} \geq \left(d - 2 - \left\lfloor \frac{d-2}{r+1} \right\rfloor\right)(3r+2) + \left\lfloor \frac{d-2}{r+1} \right\rfloor + 1. \tag{9}$$

We use above lemma to force the optimal LRC to have disjoint recovery sets.

▶ **Theorem 24.** *Let C be an optimal $[n, k, d]_q$-linear locally repairable codes of locality r with $(r+1)|n$ and $n = \Omega(dr^2)$ satisfying (9) in Lemma 23. If $d \geqslant 5$, then*

$$n \leq \begin{cases} O(q^3) & \text{if } d \bmod r+1 > 5 \text{ or } d \bmod r+1 < 2, \\ O(q^2) & \text{if } 2 \leq d \bmod r+1 \leq 5. \end{cases} \tag{10}$$

Proof. Put $h = d - 2 - \lfloor \frac{d-2}{r+1} \rfloor$. Then $n - k = \frac{n}{r+1} + h$ and $h \leq d-2$. We first follow the same line of proof in Theorem 3.3, [6]. It allows us to assume that the parity-check matrix of optimal LRC C is as follows:

$$H = \left(\begin{array}{c|c|ccc|c} \mathbf{1} & \mathbf{0} & \cdots & \cdots & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \cdots & \cdots & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \cdots & \cdots & \mathbf{1} \\ \hline \multicolumn{6}{c}{A} \end{array} \right), \tag{11}$$

where A is an $h \times n$ matrix over $\mathbb{F}_q$. The submatrix consisting of the first ℓ rows of H is a block diagonal matrix. Let $\mathbf{h}_{i,j}$ be the $((i-1)(r+1)+j)$-th column of H, i.e.,

$$\mathbf{h}_{i,j} = (\underbrace{0, \ldots, 0}_{i-1}, 1, \underbrace{0, \ldots, 0}_{\ell - i}, \mathbf{v}_{i,j})^T \tag{12}$$

for some $\mathbf{v}_{i,j} \in \mathbb{F}_q^h$, where T stands for transpose. Define

$$\mathbf{h}'_{i,j} := \mathbf{h}_{i,j} - \mathbf{h}_{i,r+1} = (\underbrace{0, \ldots, 0}_{\ell}, \mathbf{v}_{i,j} - \mathbf{v}_{i,r+1})^T$$

for $i \in [\ell]$ and $j \in [r]$. It is clear that there are only $h = d - 2 - \lfloor \frac{d-2}{r+1} \rfloor$ nonzero components in $\mathbf{h}'_{i,j}$. Assume that $d - 6 = a(r+1) + b$ for some $0 \leq b \leq r$. Then, we claim that the first $ar + b$ columns $\mathbf{h}'_{i,j}$ are linearly independent. This is because the linear combination of these $ar + b$ columns leads to

$$\sum_{i=1}^{a} \sum_{j=1}^{r} \lambda_{i,j} (\mathbf{h}_{i,j} - \mathbf{h}_{i,r+1}) + \sum_{j=1}^{b} \lambda_{a+1,j} (\mathbf{h}_{a+1,j} - \mathbf{h}_{a+1,r+1}) \neq 0$$

as it is a linear combination of $ar + b + a + 1 = d - 5$ columns of H.

Since these $ar + b$ columns $\mathbf{h}'_{i,j}$ are linearly independent, there exist $ar + b$ indices where these $ar + b$ vectors $\mathbf{h}'_{i,j}$ span the whole space. Without loss of generality, we assume they are the last $ar + b$ indices. To simply our argument, we denote by S the index set of first $ar + b$ columns and $\bar{S}$ the index set of the rest columns. For each $(x, y) \in \bar{S}$, there exist $\lambda_{x_i, y_j} \in \mathbb{F}_q$ such that $\bar{\mathbf{h}}'_{x,y} = \mathbf{h}'_{x,y} - \sum_{(i,j) \in S} \lambda_{x_i, y_j} \mathbf{h}'_{i,j}$ gives a vector whose value on last $ar + b$ indices are all zero. The number of nonzero components of $\mathbf{h}'_{x,y}$ is at most

$$h - (ar + b) = d - 2 - \lfloor \frac{d-2}{r+1} \rfloor - (d-6) + \lfloor \frac{d-6}{r+1} \rfloor = 4 + \lfloor \frac{d-6}{r+1} \rfloor - \lfloor \frac{d-2}{r+1} \rfloor.$$

On the other hand, let $\bar{\mathbf{h}}'_{x_1,y_1}$ and $\bar{\mathbf{h}}'_{x_2,y_2}$ be any two vectors of $\bar{\mathbf{h}}'_{x,y}, (x, y) \in \bar{S}$. Observe that they lie in the space spanned by $ar + b$ columns $\mathbf{h}'_{i,j}$ and $\mathbf{h}'_{x_1,y_1}, \mathbf{h}'_{x_2,y_2}$ which in turn contained in the space spanned by the first $d - 5$ columns $\mathbf{h}_{i,j}$ together with $\mathbf{h}_{x_1,y_1}, \mathbf{h}_{x_1,r+1}, \mathbf{h}_{x_2,y_2}, \mathbf{h}_{x_2,r+1}$. This implies that $\bar{\mathbf{h}}'_{x_1,y_1}$ and $\bar{\mathbf{h}}'_{x_2,y_2}$ are linearly independent. Let H_1 be the matrix whose columns consist of $\bar{\mathbf{h}}'_{x,y}, (x, y) \in \bar{S}$. Remove all zero rows in H_1 and denote the resulting matrix H_2. It is clear that any two columns of H_2 are linearly independent and H_2 has at most $4 + \lfloor \frac{d-6}{r+1} \rfloor - \lfloor \frac{d-2}{r+1} \rfloor$ rows. Let C_1 be a linear code whose parity-check matrix is H_2. We divide our discussion into two cases.

1. $\lfloor \frac{d-6}{r+1} \rfloor - \lfloor \frac{d-2}{r+1} \rfloor = 0$, i.e., $d \bmod r + 1 > 5$ or $d \bmod r + 1 < 2$.
   Then, C_1 has length $N \geq \frac{rn}{r+1} - d$, dimension $k \geq N - 4$ and distance $d \geq 3$. In the worst scenario, we assume that $k = N - 4$ and $d = 3$. Applying the Hamming bound on code C_1 gives $q^{N-4} \leq \frac{q^N}{N(q-1)}$. This implies $N \leq q^3$, i.e., $n \leq \frac{r+1}{r}(d + q^3)$.

2. $\lfloor \frac{d-6}{r+1} \rfloor - \lfloor \frac{d-2}{r+1} \rfloor = -1$ i.e., $2 \leq d \bmod r+1 \leq 5$.
Then, C_1 has length $N \geq \frac{rn}{r+1} - d$, dimension $k \geq N-3$ and distance $d \geq 3$. In the worst scenario, we assume that $k = N-3$ and $d = 3$. Applying the Hamming bound on code C_1 gives $q^{N-3} \leq \frac{q^N}{N(q-1)}$. This implies $N \leq q^2$, i.e., $n \leq \frac{r+1}{r}(d+q^2)$.

The proof is completed. ◀

References

1. Alexander Barg, Kathryn Haymaker, Everett W. Howe, Gretchen L. Matthews, and Anthony Várilly-Alvarado. Locally recoverable codes from algebraic curves and surfaces. *CoRR*, abs/1701.05212, 2017. `arXiv:1701.05212`.
2. Tom Bohman and Peter Keevash. The early evolution of the H-free process. *Inventiones mathematicae*, 181(2):291–336, August 2010.
3. Belá Bollobás. *Modern Graph Theory*. Springer, New York, 1998.
4. Parikshit Gopalan, Cheng Huang, Huseyin Simitci, and Sergey Yekhanin. On the Locality of Codeword Symbols. *IEEE Trans. Information Theory*, 58(11):6925–6934, 2012.
5. Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced Expanders and Randomness Extractors from Parvaresh–Vardy Codes. *J. ACM*, 56(4):20:1–20:34, July 2009.
6. Venkatesan Guruswami, Chaoping Xing, and Chen Yuan. How Long Can Optimal Locally Repairable Codes Be? In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20-22, 2018 - Princeton, NJ, USA*, pages 41:1–41:11, 2018.
7. Shlomo Hoory. The Size of Bipartite Graphs with a Given Girth. *J. Comb. Theory, Ser. B*, 86(2):215–220, 2002.
8. Cheng Huang, Minghua Chen, and Jin Li. Pyramid Codes: Flexible Schemes to Trade Space for Access Efficiency in Reliable Data Storage Systems. In *Sixth IEEE International Symposium on Network Computing and Applications (NCA 2007), 12 - 14 July 2007, Cambridge, MA, USA*, pages 79–86, 2007.
9. Lingfei Jin. Explicit construction of optimal locally recoverable codes of distance 5 and 6 via binary constant weight codes. *CoRR*, abs/1808.04558, 2018. `arXiv:1808.04558`.
10. Lingfei Jin, Liming Ma, and Chaoping Xing. Construction of optimal locally repairable codes via automorphism groups of rational function fields. *CoRR*, abs/1710.09638, 2017. `arXiv:1710.09638`.
11. Xudong Li, Liming Ma, and Chaoping Xing. Optimal locally repairable codes via elliptic curves. *CoRR*, abs/1712.03744, 2017. `arXiv:1712.03744`.
12. Yuan Luo, Chaoping Xing, and Chen Yuan. Optimal locally repairable codes of distance 3 and 4 via cyclic codes. *CoRR*, abs/1801.03623, 2018. `arXiv:1801.03623`.
13. N. Prakash, Govinda M. Kamath, V. Lalitha, and P. Vijay Kumar. Optimal linear codes with a local-error-correction property. In *Proceedings of the 2012 IEEE International Symposium on Information Theory, ISIT 2012, Cambridge, MA, USA, July 1-6, 2012*, pages 2776–2780, 2012.
14. Natalia Silberstein, Ankit Singh Rawat, Onur Ozan Koyluoglu, and Sriram Vishwanath. Optimal locally repairable codes via rank-metric codes. In *Proceedings of the 2013 IEEE International Symposium on Information Theory, Istanbul, Turkey, July 7-12, 2013*, pages 1819–1823, 2013.
15. Itzhak Tamo and Alexander Barg. A Family of Optimal Locally Recoverable Codes. *IEEE Trans. Information Theory*, 60(8):4661–4676, 2014.
16. Itzhak Tamo, Dimitris S. Papailiopoulos, and Alexandros G. Dimakis. Optimal Locally Repairable Codes and Connections to Matroid Theory. *IEEE Trans. Information Theory*, 62(12):6661–6671, 2016.
17. Craig Timmons and Jacques Verstraëte. A counterexample to sparse removal. *European Journal of Combinatorics*, 44:77–86, 2015.
18. Jacques Verstraëte. Personal communication.
19. Jacques Verstraëte. *Extremal problems for cycles in graphs*, pages 83–116. Springer International Publishing, Cham, 2016.

Improvements in Quantum SDP-Solving with Applications

Joran van Apeldoorn
QuSoft, CWI, The Netherlands
apeldoor@cwi.nl

András Gilyén
QuSoft, CWI, The Netherlands
gilyen@cwi.nl

Abstract

Following the first paper on quantum algorithms for SDP-solving by Brandão and Svore [13] in 2016, rapid developments have been made on quantum optimization algorithms. In this paper we improve and generalize all prior quantum algorithms for SDP-solving and give a simpler and unified framework.

We take a new perspective on quantum SDP-solvers and introduce several new techniques. One of these is the quantum operator input model, which generalizes the different input models used in previous work, and essentially any other reasonable input model. This new model assumes that the input matrices are embedded in a block of a unitary operator. In this model we give a $\widetilde{\mathcal{O}}\left(\left(\sqrt{m}+\sqrt{n}\gamma\right)\alpha\gamma^4\right)$ algorithm, where n is the size of the matrices, m is the number of constraints, γ is the reciprocal of the scale-invariant relative precision parameter, and α is a normalization factor of the input matrices. In particular for the standard sparse-matrix access, the above result gives a quantum algorithm where $\alpha = s$. We also improve on recent results of Brandão et al. [12], who consider the special case when the input matrices are proportional to mixed quantum states that one can query. For this model Brandão et al. [12] showed that the dependence on n can be replaced by a polynomial dependence on both the rank and the trace of the input matrices. We remove the dependence on the rank and hence require only a dependence on the trace of the input matrices.

After we obtain these results we apply them to a few different problems. The most notable of which is the problem of shadow tomography, recently introduced by Aaronson [2]. Here we simultaneously improve both the sample and computational complexity of the previous best results. Finally we prove a new $\widetilde{\Omega}(\sqrt{m}\alpha\gamma)$ lower bound for solving LPs and SDPs in the quantum operator model, which also implies a lower bound for the model of Brandão et al. [12].

2012 ACM Subject Classification Theory of computation → Quantum computation theory

Keywords and phrases quantum algorithms, semidefinite programming, shadow tomography

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.99

Category Track A: Algorithms, Complexity and Games

Related Version The full version of this paper is available at: `arXiv:1804.05058`.

Funding *Joran van Apeldoorn*: Supported by the Netherlands Organization for Scientific Research, grant number 617.001.351.
András Gilyén: Supported by ERC Consolidator Grant 615307-QPROGRESS and partially supported by QuantERA project QuantAlgo 680-91-034.

Acknowledgements We thank the authors of [12] for sending work-in-progress versions of their paper, and Fernando Brãndao, Tongyang Li and Xiaodi Wu for personal communication. A.G. thanks Robin Kothari and Nathan Wiebe for useful discussions. We thank Jamie Sikora for useful discussions about applications and for suggesting the state discrimination problem. We are grateful to Ronald de Wolf and Sander Gribling for useful discussions, and advice about the manuscript.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 99; pp. 99:1–99:15

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

1 Semidefinite programs

In this paper we consider *semidefinite programs* (SDPs). SDPs have many applications in optimization, notable examples include approximation of NP-hard problems like MAXCUT [21] and polynomial optimization through the Sum-Of-Squares hierarchy [24, 29]. SDPs have also found applications in quantum information theory. Examples include POVM measurement design [18] and finding the winning probability of non-local games [16].

We consider the basic (primal) form of an SDP as follows:

$$\begin{aligned} \mathrm{OPT} = \max \quad & \mathrm{Tr}\left(CX\right) \\ \text{s.t.} \quad & \mathrm{Tr}\left(A_j X\right) \leq b_j \quad \text{for all } j \in [m], \\ & X \succeq 0, \end{aligned} \tag{1}$$

where $[m] := \{1, \ldots, m\}$. The input to the problem consists of $n \times n$ Hermitian constraint matrices $A_1, \ldots, A_m$, an objective matrix C, and reals $b_1, \ldots, b_m$. For normalization purposes we assume $\|C\|, \|A_j\| \leq 1$. The number of constraints is m (we do not count the standard $X \succeq 0$ constraint for this). The variable X of this SDP is an $n \times n$ positive semidefinite (psd) matrix. We assume that $A_1 = I$ and $b_1 = R$, giving a known bound on the trace of a solution: $\mathrm{Tr}\left(X\right) \leq R$. A primal SDP also has a *dual*. For a primal SDP of the above form (1) the dual SDP is

$$\begin{aligned} \mathrm{OPT} = \min \quad & b^T y \\ \text{s.t.} \quad & \sum_{j=1}^{m} y_j A_j - C \succeq 0, \\ & y \geq 0. \end{aligned} \tag{2}$$

We assume that the dual optimum is attained and that an explicit $r \geq 1$ is known such that at least one optimal dual solution y exists $\|y\|_1 \leq r$. These assumptions imply that strong duality holds, justifying the use of OPT for both optimal values. The parameters r and R correspond to the scale of the SDP, without these bounds we could scale the SDP such that a larger error can be allowed. In some cases $r \cdot R$ may be quite large, but there are natural problems, where it is constant, cf. zero-sum games [4]. Finally, note that linear programs (LPs) correspond to the case where all constraint matrices are diagonal.

In this paper we build on the observation that a normalized psd matrix can be naturally represented as a quantum state. Since operations on quantum states can sometimes be cheaper to perform on a quantum computer than operations on classical descriptions of matrices, this can give rise to faster algorithms for solving SDPs on a quantum computer [13].

We say an algorithm is an ε-approximate *quantum SDP-solver* if for all input numbers $g \in \mathbb{R}$ and $\zeta \in (0, 1)$, with success probability $1 - \zeta$, all of the following hold:

(i) The algorithm finds a vector $y' \in \mathbb{R}^{m+1}$ and a number $z \in \mathbb{R}$ defining its output

$$X := z \frac{e^{-\sum_{j=1}^{m} y'_j A_j + y'_0 C}}{\mathrm{Tr}\left(e^{-\sum_{j=1}^{m} y'_j A_j + y'_0 C}\right)}. \tag{3}$$

The output X is an *ε-feasible primal solution* with objective value at least $g - \varepsilon$, i.e.,

$$\forall j \in [m]\colon \mathrm{Tr}\left(XA_j\right) \leq b_j + \varepsilon,$$

and $\mathrm{Tr}\left(XC\right) \geq g - \varepsilon$. If the algorithm cannot find such an output X, then it correctly concludes that no feasible solution exist (if we set $\varepsilon = 0$).

(ii) The algorithm finds a $y \in \mathbb{R}^{m+1}$ that is an *ε-feasible solution* to the dual problem with objective value at most $g + \varepsilon$, i.e.,

$$\sum_{j=1}^{m} y_j A_j - C \succeq -\varepsilon I, \tag{4}$$

and $b^T y \leq g + \varepsilon$, or it correctly concludes that no feasible y exists (if we set $\varepsilon = 0$).

(iii) The algorithm determines whether $\mathrm{OPT} \leq g - \varepsilon$ or $\mathrm{OPT} \geq g + \varepsilon$. If $\mathrm{OPT} \in (g - \varepsilon, g + \varepsilon)$ then it may output either. (Note that this essentially follows from (i)-(ii).)

Notice that this solves a decision version of the problem. However, we can easily find an approximation of OPT using binary search on g if we have an ε-approximate SDP-solver. Since ε is scale depended we actually care about the dependence on the scale invariant parameter $\gamma := Rr/\varepsilon$. An algorithm that only satisfies (i) will be called an *ε-approximate SDP primal oracle*. For such an algorithm the relevant scale invariant parameter is $\gamma := R/\varepsilon$. Due to the form of the objective value constraint in the first point, and to simplify statements like (3), we write $A_0 := -C$ and $b_0 := -g$.

Notation

We use the following definition for $\tilde{\mathcal{O}}$:

$$\widetilde{\mathcal{O}}_{d,e}\left(f(a,b,c)\right) := \mathcal{O}\left(f(a,b,c) \cdot \mathrm{polylog}(f(a,b,c),d,e)\right).$$

We define $\tilde{\Omega}$ in a similar way and $\tilde{\Theta}$ as the intersection of the two. We write δ_{ij} for the Kronecker delta function and e_j for the jth basis vector in the standard basis when the dimension of the space is clear from context. For a Hermitian matrix H we write $\mathrm{Spec}(H)$ for its spectrum (set of eigenvalues). For a function $f : \mathbb{R} \to \mathbb{R}$ we write $f(H)$ for the matrix we get by applying f to the eigenvalues of H, i.e.,

$$f(H) = U \begin{bmatrix} f(\lambda_1) & & \\ & \ddots & \\ & & f(\lambda_n) \end{bmatrix} U^{-1} \text{ where } H = U \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} U^{-1}.$$

2 SDP-solving frameworks

In this section we present two frameworks for SDP-solving, providing the basis of our quantum algorithms. First we present an algorithm to implement a primal oracle, and then the Arora-Kale framework, which is used for finding a good approximation of the optimal value and an almost feasible solution to the dual. These together implement a full SDP-solver.

Both frameworks have a very similar iterative structure, with iteration count $\widetilde{\mathcal{O}}_n\left(\gamma^2\right)$, where $1/\gamma$ is the relevant scale-invariant precision parameter (as we will see the value of γ differs by a factor r in the two cases). The main difference is that the iterative step of the primal oracle framework requires only a simple search, whereas in the Arora-Kale framework one needs to solve a slightly more complex task. Both algorithms start with $y = 0$, and in each step only a constant number of indices of y are incremented, thus in both cases we will work with a y vector that is non-negative and $\widetilde{\mathcal{O}}_n\left(1/\theta^2\right)$-sparse.

2.1 An SDP primal oracle

For the primal oracle we use the same algorithm as Brandão et al. [12] following the proof of Lemma 4.6 of Lee, Raghavendra and Steurer [25]. A few small reductions are required to apply this technique. To be able to work with density operators instead of X, the b_js in the constraints $1 \dots m$ are scaled down by a factor R, such that every solution X' to the new SDP has trace at most 1. Then, we add one new variable denoted by ω such that

$$\rho := \begin{bmatrix} X' & 0 \\ 0 & \omega \end{bmatrix}.$$

Now $\mathrm{Tr}(\rho) = 1$ and $\rho \succeq 0$ imply that $\mathrm{Tr}(X') \leq 1$, and we get a new SDP that is equivalent to the previous one. It can be shown that in our input models this reduction does not introduce more than a constant factor overhead in the complexity; note that this reduction also illustrates that for an SDP primal oracle $\frac{\varepsilon}{R}$ is the relevant scale-invariant precision parameter.

The following meta-algorithm for an SDP primal oracle assumes access to (some form of) a description of the SDP after the above-discussed reduction, and provides an output as in Eq. (3) of (i)

1. Let $y = 0 \in \mathbb{R}^{m+1}$ and $\theta = \frac{\varepsilon}{2R}$.
2. Repeat $\frac{\ln(n)}{\theta^2}$ times the following:
   a. Define $\rho := e^{-\sum_{j=0}^{m} y_j A_j} / \mathrm{Tr}\left(e^{-\sum_{j=0}^{m} y_j A_j}\right)$.
   b. Find an index j such that $\mathrm{Tr}(A_j \rho) \geq b_j$ or conclude correctly that for all j, $\mathrm{Tr}(A_j \rho) \leq b_j + \theta$.
   c. If no j is found, then we are done and output y and $z = R\mathrm{Tr}(X')$, where $\mathrm{Tr}(X')$ is the probability[1] of measuring ρ to be in the subspace corresponding to the variable X'.
   d. Otherwise update $y \leftarrow y + \theta e_j$.
3. Conclude that there is no solution for $\theta = 0$.

2.2 The Arora-Kale framework

Similarly to previous work [13] we build our results on the Arora-Kale framework. For a detailed description see the original paper by Arora and Kale [7]. For our application, the following broad overview suffices.[2]

Now we describe the Arora-Kale meta-algorithm. This algorithm assumes access to (some form of) a description of the SDP, such that the first constraint is $\mathrm{Tr}(X) \leq R$, i.e., $A_1 = I$ and $b_1 = R$. It provides an output as in Eq. (4) of (ii). (Remember that we set $A_0 = -C$ and $b_0 = -g$.)

1. Let $y = 0 \in \mathbb{R}^{m+1}$ and set $\theta = \frac{\varepsilon}{6Rr}$.
2. Repeat $\frac{\ln(n)}{\theta^2}$ times the following:
   (a) Define $\rho := e^{-\sum_{j=0}^{m} y_j A_j} / \mathrm{Tr}\left(e^{-\sum_{j=0}^{m} y_j A_j}\right)$.

[1] Note that a θ-approximation of $\mathrm{Tr}\left(X'\right)$ is easy to compute by means of amplitude estimation if ρ can be efficiently prepared as a quantum state – which is the case in our algorithms.

[2] For more discussion on general (quantum) SDP-solvers along this line, see [5].

(b) Find a $\tilde{y}$ in the polytope

$$\mathcal{P}_\delta(\rho) := \Big\{\tilde{y} \in \mathbb{R}^{m+1} : b^T\tilde{y} \leq 0, \sum_{j=0}^{m} \tilde{y}_j \mathrm{Tr}\left(A_j\rho\right) \geq -\delta, \tilde{y} \geq 0,\ \tilde{y}_0 = \frac{1}{2r},\ \|\tilde{y}\|_1 \leq 1\Big\}.$$

for $\delta = \theta$; if cannot find such a $\tilde{y}$ then conclude that none exists for $\delta = 0$.

(c) If no such $\tilde{y}$ exists, then conclude that OPT $> g$ and stop.

(d) If such a $\tilde{y}$ exists, then update $y \leftarrow y + \theta\tilde{y}$.

3. Conclude OPT $\leq g + \varepsilon$ and output $\frac{2r\theta}{\ln(n)}y + \frac{\varepsilon}{R}e_1 - e_0$ as a dual solution.

Note that in the above meta-algorithm, up to a constant factor, the θ parameter is essentially $\gamma^{-1} = \frac{\varepsilon}{rRs}$, illustrating that γ^{-1} is the relevant scale-invariant precision parameter for SDP-solving, for a more detailed discussion see [5].

Brandão and Svore [13] observed that $\rho := e^{-\sum_j y_j A_j}/\mathrm{Tr}\left(e^{-\sum_j y_j A_j}\right)$ is a quantum Gibbs state and this state can be prepared efficiently on a quantum computer, allowing fast trace estimation, in particular resulting in a quadratic speedup in n compared to classical methods.

A procedure that solves step (b) is called a *θ-oracle*, not to be confused with the input oracles. In the rest of this paper we will assume that the cost of updating the y vector is no more than the cost of a θ-oracle call. This is justified, because we use the geometric approach of Apeldoorn et al. [5, Lemma 16] for implementing a 3-sparse γ^{-1}-oracle. Their oracle returns a 3-sparse vector, and thus requires updating only 3 entries of y, which can be done efficiently using an efficient data structure.

3 Input models & Subroutines

We consider three input models: the *sparse matrix model*, the *quantum state model*, and the *quantum operator model*. The first two models were already studied in previous work. The quantum operator model is a common generalization of the other two models, and in fact any other reasonable model for SDPs, as we show.

In all models we assume quantum oracle access to the numbers b_j via the input oracle O_b satisfying[3] for all $j \in [m]$:

$$O_b|j\rangle|0\rangle = |j\rangle|b_j\rangle.$$

For all input oracles we assume we can apply both the oracle and its inverse[4] in a controlled fashion.

Sparse matrix model

In the *sparse matrix model* the input matrices are assumed to be s-row sparse for a known bound $s \in [n]$, meaning that there are at most s non-zero elements per row. The model is close to the classical model for sparse matrices. Access to the A_j matrices is provided by

[3] For simplicity we assume the bitstring representation has at most $\mathcal{O}\left(\log(nmRr/\varepsilon)\right)$ bits.

[4] When we talk about samples, e.g. in Section 4.1 of the full version of this paper [3], then we do not assume we can apply the inverse operation.

two oracles, similar to previous work on Hamiltonian simulation in [9]. The first of the two oracles is a unitary O_{sparse}, which serves the purpose of sparse access. This oracle calculates the **index** : $[m] \times [n] \times [s] \to [n]$ function, which for input (j, k, ℓ) gives the column index of the ℓth non-zero element in the kth row of A_j. We assume this oracle computes the index "in place":

$$O_{\text{sparse}}|j, k, \ell\rangle = |j, k, \mathbf{index}(j, k, \ell)\rangle. \tag{5}$$

(In the degenerate case where the kth row has fewer than ℓ non-zero entries, $\mathbf{index}(j, k, \ell)$ is defined to be ℓ together with some special symbol indicating this case.)

We also need another oracle O_A, returning a bitstring[3] representation of $(A_j)_{ki}$ for every $j \in [m]$ and $k, i \in [n]$:

$$O_A|j, k, i, z\rangle = |j, k, i, z \oplus (A_j)_{ki}\rangle. \tag{6}$$

This model corresponds directly to a classical way of accessing sparse matrices.

Quantum state model

In contrast to the sparse matrix model, the *quantum state model* is inherently quantum and has no classical counterpart for SDPs. In this model we assume that each A_j has a fixed decomposition of the form

$$A_j = \mu_j^+ \varrho_j^+ - \mu_j^- \varrho_j^- + \mu_j^I I$$

for (subnormalized) density operators $\varrho_j^\pm$, non-negative reals $\mu_j^\pm$ and real number $\mu_j^I \in \mathbb{R}$.

▶ **Definition 1** (Subnormalized density operators & Purification)**.** *A* subnormalized density operator ϱ *is a psd matrix of trace at most* 1*. A* purification *of a subnormalized density operator* ϱ *is a* 3*-register pure state such that tracing out the third register*[5] *and projecting on the subspace where the second register is* $|0\rangle$ *yields* ϱ*.*

We write "ϱ*" and "*ς*" for subnormalized density operators to distinguish them from normalized density operators, for which we write "*ρ*" and "*σ*".*

We assume access to an oracle O_μ that takes as input an index j and outputs binary representations[3] of μ_j^+, μ_j^- and μ_j^I.

Furthermore we assume access to a state-preparing oracle $O_{|\cdot\rangle}$ that prepares purifications[5] $|\psi_j^\pm\rangle$ of $\varrho_j^\pm$:

$$O_{|\cdot\rangle}|j\rangle|\pm\rangle|0\rangle = |j\rangle|\pm\rangle|\psi_j^\pm\rangle.$$

Finally we assume that a bound $B \in \mathbb{R}_+$ is known such that

$$\forall j : \mu_j^+ + \mu_j^- \leq B.$$

Note that a tight upper bound B can easily be found using $\mathcal{O}(\sqrt{m})$ quantum queries to O_μ by means of maximum finding [17].

[5] For simplicity we assume that for a d-dimensional density operator a purification has at most polylog(d) qubits.

Quantum operator model

Motivated by recent work [27, 20] we propose a new input model that we call the *quantum operator model*. In this model the input matrices are given by a unitary that implements a block-encoding:

▶ **Definition 2** (Block encoding). *Suppose that A is a w-qubit operator, $\alpha, \varepsilon \in \mathbb{R}_+$ and $k \in \mathbb{N}$, then we say that the $(a+w)$-qubit unitary U is an (α, a, ε)-block-encoding of A, if*

$$\left\| A - \alpha(\langle 0|^{\otimes a} \otimes I) U (|0\rangle^{\otimes a} \otimes I) \right\| \leq \varepsilon.$$

Roughly speaking this means that A is represented by a unitary

$$U \approx \begin{pmatrix} A/\alpha & . \\ . & . \end{pmatrix}.$$

In the quantum operator model we assume access to an oracle O_U that acts as follows:

$$O_U|j\rangle|\psi\rangle = |j\rangle(U_j|\psi\rangle).$$

Where U_j is an $(\alpha, a, 0)$-block-encoding[6] of A_j, for some fixed[7] $\alpha \in \mathbb{R}$ and for $a = \mathcal{O}(\log(nmRr/\varepsilon))$.

In Section 5 of the full version of this paper [3] we show that the sparse input model can be reduced to the quantum operator model with $\alpha = s$ and that the quantum state model can be reduced to it with $\alpha = B$. We also argue that if we can perform a measurement corresponding to $A_j \succeq 0$ using a ancilla qubits, i.e., accept a state ρ with probability $\mathrm{Tr}(A_j\rho)$, then we can implement a $(1, a+1, 0)$-block-encoding of A_j. In this way this input model is a common generalization of all reasonable input models for SDPs, since at the very least an input model should allow you to calculate $\mathrm{Tr}(A_j X)$.

Therefore if one can perform a POVM measurement on a quantum computer with a measurement operator M, one can also implement a block-encoding of M, and use it as an input matrix in our operator model. Similarly, being able to perform Hamiltonian simulation with a Hermitian matrix H gives access to H as a block-encoding, as shown by [28, 20]. Recent work [22] introduced QROM data structures that allow the efficient creation of a superposition over matrix elements of a matrix M. It turns out [15] that the corresponding block-encoding can be implemented with $\widetilde{\mathcal{O}}_{n,\gamma}(1)$ QROM calls, such that even for non-sparse matrices one has $\alpha \leq \sqrt{n}$.

Computational cost

We analyze the query complexity of algorithms and subroutines, i.e., the number of queries to controlled versions of the input oracles and their inverses. We denote the optimal quantum query complexity of an ε-approximate *quantum SDP-solver* with success probability $2/3$ by $T_{SDP}(\varepsilon)$ (this is a "meta quantity", which becomes concrete once the input model is specified). We only consider success probability $2/3$ to simplify the notation and proofs. However in all cases an ε-approximate SDP-solver with success probability $1-\zeta$ can easily be constructed using $\mathcal{O}(\log(1/\zeta)T_{SDP}(\varepsilon))$ queries.

[6] If n is not a power of 2, then we simply define A_j to be zero on the additional $2^w - n$ dimensions.

[7] Having a single normalization parameter α is not a serious restriction as it is easy to make a block-encoding more subnormalized so that every A_j gets the same normalization, cf. Lemma 14 of the full version of this paper [3].

In our algorithms we assume access to a quantum-read/classical-write RAM (known as QCRAM), and assume one read/write operation has a constant gate complexity[8]; the size of the QCRAM that we use is typically $\widetilde{\mathcal{O}}_{n,m}\left(\left(\frac{Rr}{\varepsilon}\right)^2\right)$ bits. Most often in our results the number of non-query elementary operations, i.e., two-qubit gates and QCRAM calls, matches the query complexity up to polylog factors. In particular, if not otherwise stated, in our results a T-query quantum algorithm uses at most $\widetilde{\mathcal{O}}_{n,m}(T)$ elementary operations.

Subroutines

We work with two major subroutines which need to be implemented according to the specific input model. First, the algorithms require an implementation of a Gibbs-sampler.

▶ **Definition 3** (Gibbs-sampler). *A θ-precise* Gibbs-sampler *on bounded input vectors $y \in \mathbb{R}^{m+1}_{\geq 0}$ is a quantum circuit that works under the promise that the support of y has size at most d, and $\|y\|_1 \leq K$. It takes as input a data structure storing the vector y, and for any input satisfying the promise, it creates as output a purification of a θ-approximation of the Gibbs state*

$$e^{-\sum_{j=0}^{m} y_j A_j} / \mathrm{Tr}\left(e^{-\sum_{j=0}^{m} y_j A_j}\right).$$

The minimum cost of such a circuit is denoted by $T_{Gibbs}(K, d, 4\theta)$ (this is again a "meta quantity", which becomes concrete once the input model is specified).

For technical reasons we also allow Gibbs-samplers that require a random classical input seed $S \in \{0,1\}^a$ for some $a = \mathcal{O}(\log(1/\theta))$. In this case the output should be a θ-approximation of the Gibbs state with high probability ($\geq 4/5$) over a uniformly random input seed S.

We use the approximate Gibbs states in order to estimate the quantity $\mathrm{Tr}(A_j\rho)$.

▶ **Definition 4** (Trace estimator). *A* (θ, σ)-trace estimator *is a quantum circuit that as input takes a quantum state ρ and index j. It outputs a sample from a random variable $Z_j \in \mathbb{R}$ which is an estimator of $\mathrm{Tr}(A_j\rho)$ with bias at most θ:*

$$|\mathrm{Tr}(A_j\rho) - \mathbb{E}[Z_j]| \leq \theta,$$

and the standard deviation of Z_j is at most σ. We write $T^{\sigma}_{Tr}(\theta)$ for the minimum cost of such a circuit (this is again a "meta quantity" as in the above definition).

4 Prior work

Classical SDP-solvers roughly fall into two categories: those with logarithmic dependence on R, r and $1/\varepsilon$, and those with polynomial dependence on these parameters but better dependence on m and n. In the first category the best known algorithm [26] at the time of writing has complexity

$$\widetilde{\mathcal{O}}_{Rr/\varepsilon}\left(m(m^2 + n^{\omega} + mns)\right).$$

where $\omega \in [2, 2.38]$ is the yet unknown exponent of matrix multiplication.

[8] Note that read/write operations of a QRAM or QCRAM of size S can be implemented using $\widetilde{\mathcal{O}}(S)$ two-qubit gates, so this assumption could hide a factor in the gate complexity which is at most $\widetilde{\mathcal{O}}(S)$.

In the second category Arora and Kale [7] gave an alternative framework for solving SDPs, using a matrix version of the "multiplicative weights update" method, see Section 2. Their framework can be tuned for specific types of SDPs, allowing for near linear-time algorithms in the case of for example the Goemans-Williamson SDP for the approximation of the maximum cut in a graph [21].

In 2016 Brandão and Svore [13] used the Arora-Kale framework to implement a general quantum SDP-solver in the sparse matrix model. They observed that the matrix

$$\rho := \frac{e^{-\sum_{j=0}^{m} y_j A_j}}{\operatorname{Tr}\left(e^{-\sum_{j=0}^{m} y_j A_j}\right)},$$

that is used for calculations in the Arora-Kale framework is in fact a $\log(n)$-qubit Gibbs state and can be efficiently prepared as a quantum state on a quantum computer. Using this they achieved a quantum speedup in terms of n. Combining this with a Grover-like speedup allowed for a speedup in terms of m as well, leading to an ε-approximate quantum SDP solver with complexity

$$\widetilde{\mathcal{O}}\left(\sqrt{mn}s^2\left(\frac{Rr}{\varepsilon}\right)^{32}\right).$$

They also showed an $\Omega(\sqrt{m}+\sqrt{n})$ quantum query lower bound for solving SDPs when all other parameters are constant. This left as open question whether a better lower bound, matching the $\sqrt{mn}$ upper bound, could be found. The upper bound for the sparse input model was subsequently improved by van Apeldoorn et al. [5] to

$$\widetilde{\mathcal{O}}\left(\sqrt{mn}s^2\left(\frac{Rr}{\varepsilon}\right)^{8}\right).$$

van Apeldoorn et al. also gave an $\Omega(\sqrt{\max(n,m)}\min(n,m)^{3/2})$ lower bound, albeit for non-constant parameters R and r. This bound implies that there is no general quantum SDP-solver that has a $o(nm)$ dependence on n and m and logarithmic dependence on R, r and $1/\varepsilon$. They also showed that every SDP-solver whose efficiency relies on outputting sparse dual solutions (including their algorithm and that of Brandão and Svore [13]) is limited, since problems with a lot of symmetry (like maxflow-mincut) in general require non-sparse dual solutions. Furthermore, they showed that for many combinatorial problems (like MAXCUT) R and r increase linearly with n and m.

Very recently Brandão et al. [11] gave an improved SDP-solver for the quantum state input model[9] that has a complexity bound with logarithmic dependence on n:

$$T_{SDP}(\varepsilon) = \widetilde{\mathcal{O}}_n\left(\sqrt{m}\ \operatorname{poly}\left(\frac{Rr}{\varepsilon}, B, \max_{j\in\{0,\dots,m\}}[\operatorname{rank}(A_j)]\right)\right).$$

Brandão et al. also applied their algorithm to the problem of *shadow tomography*, giving the first non-trivial application of a quantum SDP-solver.

Subsequently these results where further improved by the introduction of the Fast Quantum OR lemma by the same authors [12]. Approaches prior to [12] searched for a violated constraint in the SDP using Grover-like techniques, resulting in a multiplicative

[9] This model was already introduced in the first version of [13] together with a similar complexity statement, but there were some unresolved issues in the proof, that were only fixed by the contributions of [11].

complexity of Gibbs-sampling and searching. The Fast Quantum OR lemma can be used to separate the search phase from the initial Gibbs-state preparation phase. This led to the improved complexity bound [12] of

$$\widetilde{\mathcal{O}}_n\left(\left(\sqrt{m}+\mathrm{poly}(\max_{j\in\{0\ldots m\}}[\mathrm{rank}(A_j)])\right)\ \mathrm{poly}\left(\frac{Rr}{\varepsilon},B\right)\right).$$

We thank the authors of [12] for sending us an early draft of [12] introducing the Fast Quantum OR Lemma, which enabled us to work on these improvements. During the correspondence the application of the Fast OR lemma to the sparse matrix model was independently suggested by Brandão et al. [30] and by us.

5 Our results

In this paper we present multiple results. The main contribution consists of multiple improvements to the algorithms for SDP solving, based on combining various recent quantum algorithmic developments. Although some of these improvements require quite technical proofs, they come from simple new perspectives and ideas, often combining previous works in new ways. We also apply the resulting algorithms to a few problems in convex optimization. Finally, we prove a new lower bound that fits the novel input models that we work with.

5.1 Improvements to the quantum algorithms

In this paper we build on the Arora-Kale framework for SDP-solving in a similar fashion as [5, 13] and also use results from [11, 25] to construct a primal oracle. We improve on the previous results about quantum SDP-solving in three different ways:

Two-Phase Quantum Search and Minimum Finding

We give a computationally more efficient version of the Gentle Quantum Search Lemma [2] using the Fast Quantum OR Lemma from [12]. We also extend this to minimum finding to get our *Two-Phase Quantum Minimum Finding* (Lemma 7 of the full version of this paper [3]). As independently observed by the authors of [12] the Fast Quantum OR Lemma gives a speed-up for SDP primal oracles in general. Moreover, using Two-Phase Quantum Minimum Finding, we show how to improve the upper bound on the complexity of general SDP-solving from

$$T_{SDP}(\varepsilon)=\widetilde{\mathcal{O}}_n\left(\sqrt{m}T^{\sigma}_{Tr}((4\gamma)^{-1})T_{Gibbs}(\gamma,\gamma^2,\gamma^{-1})\gamma^3\sigma\right) \tag{7}$$

as implied in previous work [13, 5] to

$$T_{SDP}(\varepsilon)=\widetilde{\mathcal{O}}_n\left(\left(\sqrt{m}T^{\sigma}_{Tr}((4\gamma)^{-1})+T_{Gibbs}(\gamma,\gamma^2,\gamma^{-1})\right)\gamma^4\sigma^2\right), \tag{8}$$

where $\gamma=\Theta(Rr/\varepsilon)$. For the complexity of SDP primal oracles, the same upper bounds hold.

Quantum operator model and efficient data structures

We introduce the quantum operator input model unifying prior approaches. We show that both the sparse model and the quantum state model can be reduced to the quantum operator model with a constant overhead and with the choices of $\alpha=s$ and $\alpha=B$ respectively.

Moreover, we show that for $\sigma = \Theta(1)$, we have that

$$T^{\sigma}_{Tr}((4\gamma)^{-1}) = \widetilde{\mathcal{O}}_{\gamma}(\alpha),$$

in the quantum operator model.

The complexity of Gibbs-sampling in the sparse matrix input model previously was [5]:

$$T_{Gibbs}(K, d, \theta) = \widetilde{\mathcal{O}}_{\theta}\left(\sqrt{n}Ks^2d^2\right).$$

By considering the operator model, in which we can show how to simulate a linear combination of Hamiltonians efficiently, we can improve this to

$$T_{Gibbs}(K, d, \theta) = \widetilde{\mathcal{O}}_{\theta,d}\left(\sqrt{n}K\alpha\right).$$

This result is based on the idea of gradually building up an efficient data structure for state preparation, following ideas of [22]. This demonstrates that these data structures can be used efficiently even if one does not assume preprocessed data. Moreover, it shows that working in the operator model does not only unify prior approaches but also inspires more efficient quantum algorithms due to its conceptual clarity.

Table 1 Summary of the role of our various improvements, the theorem numbers refer to the numbers in the full version of the paper [3]. The main new results are on the bottom, the other complexity statements represent partial results following from only applying some of the improvements. We present the results for the sparse matrix and quantum state input models for comparison to prior work. However, note that our results presented for sparse input hold more generally for the quantum operator input model; to get the corresponding results one should just replace s by α in the table. Thereby similar bounds hold in the case of the quantum state input model too, after replacing s by B, which can be beneficial when $B^{2.5}\gamma^{2.5} \geq \sqrt{n}$. Notation: $\mathrm{rk} = \max_{j\in\{0,\dots,m\}} \mathrm{rank}(A_j)$ and $\gamma = \frac{Rr}{\varepsilon}$.

	Without OR lemma / Two-Phase Search	
	Sparse input	Quantum state input
Previous Gibbs-sampling	$\widetilde{\mathcal{O}}\left(\sqrt{mn}s^2\gamma^8\right)$ [5]	$\widetilde{\mathcal{O}}_n\left(\sqrt{m}\mathrm{poly}\left(\gamma, B, \mathrm{rk}\right)\right)$ [11]
Improved Gibbs-sampling	$\widetilde{\mathcal{O}}\left(\sqrt{mn}s\gamma^4\right)$ Corollary 18	$\widetilde{\mathcal{O}}_n\left(\sqrt{m}B^{3.5}\gamma^{6.5}\right)$ Corollary 25
	With OR lemma / Two-Phase Search	
	Sparse input	Quantum state input
Previous Gibbs-sampling	$\widetilde{\mathcal{O}}\left(\left(\sqrt{m}+\sqrt{n}s\gamma^5\right)s\gamma^4\right)$ Theorem[10] 8 + [5]	$\widetilde{\mathcal{O}}_n\left(\left(\sqrt{m}+\mathrm{poly}(\mathrm{rk})\right)\mathrm{poly}\left(\gamma, B\right)\right)$ [12]
Improved Gibbs-sampling	$\widetilde{\mathcal{O}}\left(\left(\sqrt{m}+\sqrt{n}\gamma\right)s\gamma^4\right)$ Theorem 17	$\widetilde{\mathcal{O}}_n\left((\sqrt{m}+B^{2.5}\gamma^{3.5})B\gamma^4\right)$ Theorem 24

Gibbs sampling for the quantum state model

We develop a new method for Gibbs-sampling in the quantum state model. As noted in [12] this model has the nice property that it is relatively easy to find the important eigenspaces of the input matrices. We introduce a new technique for finding these important eigenspaces

that, in contrast to the approach in [12], does not introduce a dependence on the rank of the input matrices in the complexity. In particular we improve the complexity bound of [12]

$$T_{Gibbs}(K,d,\theta) = \mathcal{O}\left(\text{poly}(K,B,d,1/\theta,\max_{j\in\{0\ldots m\}}[\text{rank}(A_j)])\right),$$

to

$$T_{Gibbs}(K,d,\theta) = \widetilde{\mathcal{O}}_{d,\theta,n}\left((KB)^{3.5}\right),$$

both making the polynomial dependence explicit and improving it.

An important consequence of this improvement is that we do not get a dependence on the rank of the input matrices in the complexity of SDP solving, unlike Brandão et al. [12]. Since the most natural use for the quantum state model is when the A_j matrices naturally correspond to quantum states, B is often just 1. However, if the states are highly mixed, then the rank is about n, eliminating the speedup over the sparse input model when a rank dependence is present. Finally note that this Gibbs-sampling method is only beneficial if $\sqrt{n} \leq (KB)^{2.5}$, otherwise the reduction to the quantum operator model with $\alpha = B$ gives a better algorithm.

For the quantum operator input model the above improvements lead to the complexity bound

$$T_{SDP}(\varepsilon) = \widetilde{\mathcal{O}}\left(\left(\sqrt{m}+\sqrt{n}\gamma\right)\alpha\gamma^4\right), \tag{9}$$

where $\gamma := \frac{Rr}{\varepsilon}$. Note that the $\Omega(\sqrt{n}+\sqrt{m})$ lower bound of [13] also applies to the quantum operator model due to our reductions, matching the above upper bound (9) up to polylog factors in n and m when γ and α are constant. For the quantum state input model our improved Gibbs-sampler yields the complexity bound

$$T_{SDP}(\varepsilon) = \widetilde{\mathcal{O}}_n\left(\left(\sqrt{m}+B^{2.5}\gamma^{3.5}\right)B\gamma^4\right).$$

In both cases, the same bound holds for an SDP primal oracle but with $\gamma := R/\varepsilon$.

5.2 Applications

In Section 4 of the full version of this paper [3] we give some applications of quantum SDP-solvers. Due to the large error dependence the last two of these are not of practical interest. However they do show that a theoretical improvement in one of the parameters is possible over classical computers and more specified quantum algorithms might improve the error dependence.

Shadow tomography of quantum states

We extend the idea of applying SDP-solving to the problem of shadow tomography: given an unknown, n-dimensional quantum state ρ, find ε-additive approximations of the expectation values $\text{Tr}(E_1\rho),\ldots,\text{Tr}(E_m\rho)$ of several binary measurement operators. This problem was introduced by Aaronson in [2], he gave an efficient algorithm in terms of the number of samples from ρ. In particular he proved that $\widetilde{\mathcal{O}}\left(\log^4(m)\log(n)/\varepsilon^5\right)$ samples suffice. Brandão et al. [12] applied their SDP-solver to get a more efficient algorithm in terms of computation time when the measurements E_i are given in the quantum state model, while keeping the sample complexity as low as $\text{poly}(\log(m),\log(n),1/\varepsilon,B)$. We simultaneously improve on both results, giving a sample bound of $\widetilde{\mathcal{O}}\left(\log^4(m)\log(n)/\varepsilon^4\right)$ while also improving the best

known time complexity [2, 12] of the implementation for all input models. Finally we show that if we can efficiently implement the measurements $\mathrm{Tr}(E_1\rho), \ldots, \mathrm{Tr}(E_m\rho)$ on a quantum computer, then we can also efficiently represent $E_1, \ldots, E_m$ using the quantum operator input model, hence the computational complexity can be stated in terms of the number of measurements needed.

Quantum state discrimination

We apply the SDP-solvers to the problem of quantum state discrimination: given a set of quantum states, what is the best POVM for discriminating between the states? We consider the case of minimizing the total error in the measurements. In this case we get an algorithm with running time $\widetilde{\mathcal{O}}\left(\sqrt{k}\ \mathrm{poly}(d, 1/\varepsilon)\right)$ in the sparse input model, where k is the number of states and d is the dimension of the states. Due to the quantum state model for SDP-solving, we can also solve the problem when the states that need to be discriminated are actually given as quantum states, rather than classical descriptions of density operators.

Optimal design

We apply our sparse SDP-solver to the problem of E-optimal design: given a set of k experiments, find the optimal distribution of the experiments that minimizes the variance in our knowledge of a d-dimensional system. Our final bound is $\widetilde{\mathcal{O}}\left((\sqrt{k}+\sqrt{d})\mathrm{poly}(1/\varepsilon, P)\right)$, where P is a parameter that depends on the standard deviation of the experiments.

5.3 Lower bounds

We end the paper with proving new lower bounds. Lower bounds on the quantum query complexity of SDP-solving for the sparse input model were presented in previous works [13, 5]. We add to this by giving $\widetilde{\Omega}(\sqrt{m}B/\varepsilon)$ and $\widetilde{\Omega}(\sqrt{m}\alpha/\varepsilon)$ bounds for the quantum state model and quantum operator model respectively. These lower bounds show that the $\sqrt{m}$ factor and the polynomial dependence on the parameters B, α, and $1/\varepsilon$ are necessary.

Compared to problems with a discrete input, proving lower bounds on continuous-input quantum problems gives rise to extra challenges and often requires more involved techniques, see for example the work of Belovs [8] on generalizations of the adversary method. Due to these difficulties, fewer results are known in this regime. Examples of known continuous-input lower-bound results include phase-estimation related problems (cf. Bessen [10]) and the complexity-theoretic version of the no-cloning theorem due to Aaronson [1]. Recently, a new hybrid-method based approach was developed by Gilyén et al. [19] in order to handle continuous-input oracles, which they use for proving a lower bound for gradient computation. We use their techniques to prove our lower bounds, combined with efficient reductions between input models stemming from the smooth-functions of Hamitonians techniques developed in the work of van Apeldoorn et al. [5].

6 Subsequent work

A few months after the first version of our paper was posted on the arXiv, Kerenidis and Prakash [23] gave a quantum interior point algorithm for solving LPs and SDPs. They work in an input model where the input matrices are stored in QROM, which input model is also covered by our quantum operator input model. However, it is hard to compare their complexities to ours, because their final complexity statement depends polynomially on the condition number of the matrices that the interior point method encounters, and they do not

give explicit bounds for these condition numbers. Also they have two accuracy parameters, while one accuracy parameter only appears as a logarithmic factor, their complexity depends polynomially on the other.

Very recently Apeldoorn et al. [6] and Chakrabarti et al. [14] developed improved quantum algorithms for general black-box convex optimization. Since we work in a model where we are given access directly to the constraints defining the problem, our results are incomparable.

References

1 Scott Aaronson. Quantum Copy-Protection and Quantum Money. In *Proceedings of the 24th IEEE Conference on Computational Complexity (CCC)*, pages 229–242, 2009. arXiv: 1110.5353 doi:10.1109/CCC.2009.42.

2 Scott Aaronson. Shadow Tomography of Quantum States. In *Proceedings of the 50th ACM Symposium on Theory of Computing (STOC)*, pages 325–338, 2018. arXiv: 1711.01053 doi:10.1145/3188745.3188802.

3 Joran van Apeldoorn and András Gilyén. Improvements in Quantum SDP-Solving with Applications. arXiv: 1804.05058, 2018.

4 Joran van Apeldoorn and András Gilyén. Quantum algorithms for zero-sum games. arXiv: 1904.03180, 2019.

5 Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Quantum SDP-Solvers: Better upper and lower bounds. In *Proceedings of the 58th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 403–414, 2017. arXiv: 1705.01843 doi:10.1109/FOCS.2017.44.

6 Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Convex optimization using quantum oracles. arXiv: 1809.00643, 2018.

7 Sanjeev Arora and Satyen Kale. A Combinatorial, Primal-Dual Approach to Semidefinite Programs. *Journal of the ACM*, 63(2):12:1–12:35, 2016. Earlier version in STOC'07. doi:10.1145/2837020.

8 Aleksandrs Belovs. Variations on Quantum Adversary. arXiv: 1504.06943, 2015.

9 Dominic W. Berry, Andrew M. Childs, and Robin Kothari. Hamiltonian Simulation with Nearly Optimal Dependence on all Parameters. In *Proceedings of the 56th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 792–809, 2015. arXiv: 1501.01715 doi:10.1109/FOCS.2015.54.

10 Arvid J. Bessen. Lower bound for quantum phase estimation. *Physical Review A*, 71(4):042313, 2005. arXiv: quant-ph/0412008 doi:10.1103/PhysRevA.71.042313.

11 Fernando G. S. L. Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M. Svore, and Xiaodi Wu. Exponential Quantum Speed-ups for Semidefinite Programming with Applications to Quantum Learning, 2017. First arXiv version. arXiv:1710.02581v1.

12 Fernando G. S. L. Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M. Svore, and Xiaodi Wu. Quantum SDP Solvers: Large Speed-ups, Optimality, and Applications to Quantum Learning, 2018. arXiv:1710.02581v2.

13 Fernando G. S. L. Brandão and Krysta M. Svore. Quantum Speed-ups for Solving Semidefinite Programs. In *Proceedings of the 58th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 415–426, 2017. arXiv: 1609.05537 doi:10.1109/FOCS.2017.45.

14 Shouvanik Chakrabarti, Andrew M. Childs, Tongyang Li, and Xiaodi Wu. Quantum algorithms and lower bounds for convex optimization. arXiv: 1809.01731, 2018.

15 Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. The power of block-encoded matrix powers: improved regression techniques via faster Hamiltonian simulation. arXiv: 1804.01973, 2018.

16 Richard Cleve, Peter Høyer, Benjamin Toner, and John Watrous. Consequences and limits of nonlocal strategies. In *Proceedings of the 19th IEEE Conference on Computational Complexity (CCC)*, pages 236–249, 2004. arXiv: quant-ph/0404076 doi:10.1109/CCC.2004.1313847.

17 Christoph Dürr and Peter Høyer. A Quantum Algorithm for Finding the Minimum. arXiv: quant-ph/9607014, 1996.

18 Yonina C. Eldar. A Semidefinite Programming Approach to Optimal Unambiguous Discrimination of Quantum States. *IEEE Transactions on Information Theory*, 49:446–456, 2003. arXiv: quant-ph/0206093 doi:10.1109/TIT.2002.807291.

19 András Gilyén, Srinivasan Arunachalam, and Nathan Wiebe. Optimizing quantum optimization algorithms via faster quantum gradient computation. In *Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1425–1444, 2019. arXiv: 1711.00465 doi:10.1137/1.9781611975482.87.

20 András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st ACM Symposium on Theory of Computing (STOC)*, 2019. (to appear) arXiv: 1806.01838 doi:10.1145/3313276.3316366.

21 Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995. Earlier version in STOC'94. doi:10.1145/227683.227684.

22 Iordanis Kerenidis and Anupam Prakash. Quantum Recommendation Systems. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 49:1–49:21, 2017. arXiv: 1603.08675 doi:10.4230/LIPIcs.ITCS.2017.49.

23 Iordanis Kerenidis and Anupam Prakash. A Quantum Interior Point Method for LPs and SDPs. arXiv: 1808.09266, 2018.

24 Jean Lasserre. Global Optimization with Polynomials and the Problem of Moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001. doi:10.1137/S1052623400366802.

25 James R. Lee, Prasad Raghavendra, and David Steurer. Lower Bounds on the Size of Semidefinite Programming Relaxations. In *Proceedings of the 47th ACM Symposium on Theory of Computing (STOC)*, pages 567–576, 2015. arXiv: 1411.6317 doi:10.1145/2746539.2746599.

26 Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *Proceedings of the 56th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1049–1065, 2015. arXiv: 1508.04874 doi:10.1109/FOCS.2015.68.

27 Guang Hao Low and Isaac L. Chuang. Hamiltonian Simulation by Qubitization. arXiv: 1610.06546, 2016.

28 Guang Hao Low and Isaac L. Chuang. Hamiltonian Simulation by Uniform Spectral Amplification. arXiv: 1707.05391, 2017.

29 Pablo A. Parrilo. *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD thesis, California Institute of Technology, 2000. URL: https://thesis.library.caltech.edu/1647/.

30 Xiaodi Wu. Personal communication. Email, November 2017.

Minimizing GFG Transition-Based Automata

Bader Abu Radi
School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel
bader.aburadi@gmail.com

Orna Kupferman
School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel
orna@cs.huji.ac.il

Abstract

While many applications of automata in formal methods can use nondeterministic automata, some applications, most notably synthesis, need deterministic or *good-for-games* automata. The latter are nondeterministic automata that can resolve their nondeterministic choices in a way that only depends on the past. The *minimization* problem for nondeterministic and deterministic Büchi and co-Büchi word automata are PSPACE-complete and NP-complete, respectively. We describe a polynomial minimization algorithm for good-for-games *co-Büchi* word automata with *transition-based* acceptance. Thus, a run is accepting if it traverses a set of designated transitions only finitely often. Our algorithm is based on a sequence of transformations we apply to the automaton, on top of which a minimal quotient automaton is defined.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory; Theory of computation → Automata over infinite objects

Keywords and phrases Minimization, Deterministic co-Büchi Automata

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.100

Category Track B: Automata, Logic, Semantics, and Theory of Programming

1 Introduction

Automata theory is one of the longest established areas in Computer Science. A classical problem in automata theory is *minimization*: generation of an equivalent automaton with a minimal number of states. For automata on finite words, the picture is well understood: For nondeterministic automata, minimization is PSPACE-complete [16], whereas for deterministic automata, a minimization algorithm, based on the Myhill-Nerode right congruence [28, 29], generates in polynomial time a canonical minimal deterministic automaton [14]. Essentially, the canonical automaton, a.k.a. the *quotient automaton*, is obtained by merging equivalent states.

A prime application of automata theory is specification, verification, and synthesis of reactive systems [36, 8]. The automata-theoretic approach considers relationships between systems and their specifications as relationships between languages. Since we care about the on-going behavior of nonterminating systems, the automata run on infinite words. Acceptance in such automata is determined according to the set of states that are visited infinitely often along the run. In Büchi automata [5] (NBW and DBW, for nondeterministic and deterministic Büchi word automata, respectively), the acceptance condition is a subset α of states, and a run is accepting iff it visits α infinitely often. Dually, in co-Büchi automata (NCW and DCW), a run is accepting iff it visits α only finitely often. In spite of the extensive use of automata on infinite words in verification and synthesis algorithms and tools, some fundamental problems around their minimization are still open. For nondeterministic automata, minimization is PSPACE-complete, as it is for automata on finite words. Before

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 100; pp. 100:1–100:16


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

we describe the situation for deterministic automata, let us elaborate some more on the power of nondeterminism in the context of automata on infinite words, as this would be relevant to our contribution.

For automata on finite words, nondeterminism does not increase the expressive power, yet it leads to an exponential succinctness [31]. For automata on infinite words, nondeterminism may increase the expressive power and also leads to an exponential succinctness. For example, NBWs are strictly more expressive than DBWs [21]. In some applications of automata on infinite words, such as model checking, algorithms can proceed with nondeterministic automata, whereas in other applications, such as synthesis and control, they cannot. There, the advantages of nondeterminism are lost, and the algorithms involve complicated determinization constructions [32] or acrobatics for circumventing determinization [20]. Essentially, the inherent difficulty of using nondeterminism in synthesis lies in the fact that each guess of the nondeterministic automaton should accommodate all possible futures.

The study of nondeterministic automata that can resolve their nondeterministic choices in a way that only depends on the past and still accept all words in the language started already in 1996 [19], where the setting is modeled by means of tree automata for derived languages. It then continued by means of *good for games* (GFG) automata, introduced in [13].[1] Formally, a nondeterministic automaton $\mathcal{A}$ over an alphabet Σ is GFG if there is a strategy g that maps each finite word $u \in \Sigma^*$ to the transition to be taken after u is read; and following g results in accepting all the words in the language of $\mathcal{A}$. Note that a state q of $\mathcal{A}$ may be reachable via different words, and g may suggest different transitions from q after different words are read. Still, g depends only on the past, namely on the word read so far. Obviously, there exist GFG automata: deterministic ones, or nondeterministic ones that are *determinizable by pruning* (DBP); that is, ones that just add transitions on top of a deterministic automaton. In fact, the GFG automata constructed in [13] are DBP.[2]

In terms of expressive power, it is shown in [19, 30] that GFG automata with an acceptance condition γ (e.g., Büchi) are as expressive as deterministic γ automata. The picture in terms of succinctness is diverse. For automata on finite words, GFG automata are always DBP [19, 26]. For automata on infinite words, in particular NBWs and NCWs, GFG automata need not be DBP [3]. Moreover, the best known determinization construction for GFG-NBWs is quadratic, whereas determinization of GFG-NCWs has an exponential blow-up lower bound [17]. Thus, in terms of succinctness, GFG automata on infinite words are more succinct (possibly even exponentially) than deterministic ones. Further research studies characterization, typeness, complementation, and further constructions and decision procedures for GFG automata [17, 4, 2].

Back to the minimization problem. Recall that for finite words, an equivalent minimal deterministic automaton can be obtained by merging equivalent states. A similar algorithm is valid for determinisitic *weak* automata on infinite words: DBWs in which each strongly connected component is either contained in α or is disjoint form α [27, 23]. For general DBWs (and hence, also DCWs, as the two dualize each other), merging of equivalent states fails, and minimization is NP-complete [33].

The intractability of the minimization problem has led to a development of numerous heuristics. The heuristics either relax the minimality requirement, for example algorithms based on *fair bisimulation* [10], which reduce the state space but need not return a minimal

[1] GFGness is also used in [6] in the framework of cost functions under the name "history-determinism".

[2] As explained in [13], the fact that the GFG automata constructed there are DBP does not contradict their usefulness in practice, as their transition relation is simpler than the one of the embodied deterministic automaton and it can be defined symbolically.

automaton, or relax the equivalence requirement, for example algorithms based on *hyper-minimization* [1, 15] or *almost-equivalence* [33], which come with a guarantee about the difference between the language of the original automaton and the ones generated by the algorithm. In some cases, these algorithms do generate of a minimal equivalent automaton (in particular, applying relative minimization based on almost equivalence on a deterministic weak automaton results in an equivalent minimal weak automaton [33]), but in general, they are only heuristics. In an orthogonal line of work, researchers have studied minimization in richer settings of automata on finite words. One direction is to allow some nondeterminism. As it turns out, however, even the slightest extension of the deterministic model towards a nondeterministic one, for example by allowing at most one nondeterministic choice in every accepting computation or allowing just two initial states instead of one, results in NP-complete minimization problems [24]. Another direction is a study of quantitative settings. Here, the picture is diverse. For example, minimization of deterministic lattice automata [18] is polynomial for automata over linear lattices and is NP-complete for general lattices [11], and minimization of deterministic weighted automata over the tropical semiring is polynomial [25], yet the problem is open for general semirings.

Proving NP-hardness for DBW minimization, Schewe used a reduction from the vertex-cover problem [33]. Essentially[3], given a graph $G = \langle V, E\rangle$, we seek a minimal DBW for the language L_G of words of the form $v_{i_1}^+ \cdot v_{i_2}^+ \cdot v_{i_3}^+ \cdots \in V^\omega$, where for all $j \geq 1$, we have that $\langle v_{i_j}, v_{i_{j+1}}\rangle \in E$. We can recognize L_G by an automaton obtained from G by adding self loops to all vertices, labelling each edge by its destination, and requiring a run to traverse infinitely many original edges of G. Indeed, such runs correspond to words that traverse an infinite path in G, possibly looping at vertices, but not getting trapped in a self loop, as required by L_G. When, however, the acceptance condition is defined by a set of vertices, rather than edges, we need to duplicate some states, and a minimal duplication corresponds to a minimal vertex cover. Thus, a natural question arises: Is there a polynomial minimization algorithms for DBWs and DCWs whose acceptance condition is *transition based*? Beyond the theoretical interest, there is recently growing use of transition-based automata in practical applications, with evidences they offer a simpler translation of LTL formulas to automata and enable simpler constructions and decision procedures [9, 7, 34, 22].

In this paper we present a significant step towards a positive answer to this question and describe a polynomial-time algorithm for the minimization of GFG transition-based NCWs. Consider a GFG-NCW $\mathcal{A}$. Our algorithm is based on a chain of transformations we apply to $\mathcal{A}$. Some of the transformations are introduced in [17], in algorithms for deciding GFGness. We add two more transformations and prove that they guarantee minimality. Our reasoning is based on a careful analysis of the *safe components* of $\mathcal{A}$, namely the components obtained by removing transitions in α. We show that a minimal GFG-NCW equivalent to $\mathcal{A}$ can be obtained by defining an order on the safe components, and applying the quotient construction on a GFG-NCW obtained by restricting attention to states that belong to components that form a frontier in this order.

The paper is organized as follows. In Section 2, we define GFG-NCWs and some properties of GFG-NCWs that can be attained in polynomial time using existing results. In Section 3, we describe two additional properties and prove that they guarantee minimality. Then, in Sections 4 – 5, we show how the two properties can be attained in polynomial time, thus concluding our minimization procedure. In Section 6, we discuss how our results contribute to the quest for efficient DBW and DCW minimization.

[3] The exact reduction is more complicated and involves an additional letter that is required for cases in which vertices in the graph have similar neighbours.

2 Preliminaries

For a finite nonempty alphabet Σ, an infinite *word* $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$ is an infinite sequence of letters from Σ. A *language* $L \subseteq \Sigma^\omega$ is a set of words. We denote the empty word by ϵ, the set of finite words over Σ by Σ^*. For $i \geq 0$, we use $w[1, i]$ to denote the (possibly empty) prefix $\sigma_1 \cdot \sigma_2 \cdots \sigma_i$ of w and use $w[i+1, \infty]$ to denote its suffix $\sigma_{i+1} \cdot \sigma_{i+2} \cdots$.

A *nondeterministic automaton* over infinite words is $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, where Σ is an alphabet, Q is a finite set of *states*, $q_0 \in Q$ is an *initial state*, $\delta : Q \times \Sigma \to 2^Q \setminus \emptyset$ is a *transition function*, and α is an *acceptance condition*, to be defined below. For states q and s and a letter $\sigma \in \Sigma$, we say that s is a σ-successor of q if $s \in \delta(q, \sigma)$. The *size* of $\mathcal{A}$, denoted $|\mathcal{A}|$, is defined as its number of states, thus, $|\mathcal{A}| = |Q|$. Note that $\mathcal{A}$ is *total*, in the sense that it has at least one successor for each state and letter, and that $\mathcal{A}$ may be *nondeterministic*, as the transition function may specify several successors for each state and letter. If $|\delta(q, \sigma)| = 1$ for every state $q \in Q$ and letter $\sigma \in \Sigma$, then $\mathcal{A}$ is *deterministic*.

When $\mathcal{A}$ runs on an input word, it starts in the initial state and proceeds according to the transition function. Formally, a *run* of $\mathcal{A}$ on $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$ is an infinite sequence of states $r = r_0, r_1, r_2, \ldots \in Q^\omega$, such that $r_0 = q_0$, and for all $i \geq 0$, we have that $r_{i+1} \in \delta(r_i, \sigma_{i+1})$. We sometimes extend δ to sets of states and finite words. Then, $\delta : 2^Q \times \Sigma^* \to 2^Q$ is such that for every $S \in 2^Q$, finite word $u \in \Sigma^*$, and letter $\sigma \in \Sigma$, we have that $\delta(S, \epsilon) = S$, $\delta(S, \sigma) = \bigcup_{s \in S} \delta(s, \sigma)$, and $\delta(S, u \cdot \sigma) = \delta(\delta(S, u), \sigma)$. Thus, $\delta(S, u)$ is the set of states that $\mathcal{A}$ may reach when it reads u from some state in S.

The transition function δ induces a transition relation $\Delta \subseteq Q \times \Sigma \times Q$, where for every two states $q, s \in Q$ and letter $\sigma \in \Sigma$, we have that $\langle q, \sigma, s \rangle \in \Delta$ iff $s \in \delta(q, \sigma)$. We sometimes view the run $r = r_0, r_1, r_2, \ldots$ on $w = \sigma_1 \cdot \sigma_2 \cdots$ as an infinite sequence of successive transitions $\langle r_0, \sigma_1, r_1 \rangle, \langle r_1, \sigma_2, r_2 \rangle, \ldots \in \Delta^\omega$. The acceptance condition α determines which runs are "good". We consider here *transition-based* automata, in which α refers to the set of transitions that are traversed infinitely often during the run; specifically, $\alpha \subseteq \Delta$. We use the terms *α-transitions* and *$\bar{\alpha}$-transitions* to refer to transitions in α and in $\Delta \setminus \alpha$, respectively. We also refer to restrictions δ^α and $\delta^{\bar{\alpha}}$ of δ, where for all $q, s \in Q$ and $\sigma \in \Sigma$, we have that $s \in \delta^\alpha(q, \sigma)$ iff $\langle q, \sigma, s \rangle \in \alpha$, and $s \in \delta^{\bar{\alpha}}(q, \sigma)$ iff $\langle q, \sigma, s \rangle \in \Delta \setminus \alpha$. For a run $r \in \Delta^\omega$, let $inf(r) \subseteq \Delta$ be the set of transitions that r traverses infinitely often. Thus, $inf(r) = \{\langle q, \sigma, s \rangle \in \Delta : q = r_i, \sigma = \sigma_{i+1}$ and $s = r_{i+1}$ for infinitely many i's$\}$. In *co-Büchi* automata, a run r is *accepting* iff $inf(r) \cap \alpha = \emptyset$, thus if r traverses transitions in α only finitely often. A run that is not accepting is *rejecting*. A word w is accepted by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on w. The language of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts. Two automata are *equivalent* if their languages are equivalent. We use tNCW and tDCW to abbreviate nondeterministic and deterministic transition-based co-Büchi automata over infinite words, respectively.

For a state $q \in Q$ of an automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, we define $\mathcal{A}^q$ to be the automaton obtained from $\mathcal{A}$ by setting the initial state to be q. Thus, $\mathcal{A}^q = \langle \Sigma, Q, q, \delta, \alpha \rangle$. We say that two states $q, s \in Q$ are *equivalent*, denoted $q \sim_{\mathcal{A}} s$, if $L(\mathcal{A}^q) = L(\mathcal{A}^s)$. The automaton $\mathcal{A}$ is *semantically deterministic* if different nondeterministic choices lead to equivalent states. Thus, for every state $q \in Q$ and letter $\sigma \in \Sigma$, all the σ-successors of q are equivalent: for every two states $s, s' \in Q$ such that $\langle q, \sigma, s \rangle$ and $\langle q, \sigma, s' \rangle$ are in Δ, we have that $s \sim_{\mathcal{A}} s'$. The following proposition follows immediately from the definitions.

▶ **Proposition 1.** *Consider a semantically deterministic automaton $\mathcal{A}$, states $q, s \in Q$, letter $\sigma \in \Sigma$, and transitions $\langle q, \sigma, q' \rangle, \langle s, \sigma, s' \rangle \in \Delta$. If $q \sim_{\mathcal{A}} s$, then $q' \sim_{\mathcal{A}} s'$.*

A tNCW $\mathcal{A}$ is *safe deterministic* if by removing its α-transitions, we get a (possibly not total) deterministic automaton. Thus, $\mathcal{A}$ is *safe deterministic* if for every state $q \in Q$ and letter $\sigma \in \Sigma$, it holds that $|\delta^{\bar{\alpha}}(q, \sigma)| \leq 1$. We refer to the components we get by removing $\mathcal{A}$'s α-transitions as the *safe components* of $\mathcal{A}$, and we denote the set of safe components of $\mathcal{A}$ by $S(\mathcal{A})$. For a safe component $S \in S(\mathcal{A})$, the *size* of S, denoted $|S|$, is the number of states in S. Note that an accepting run of $\mathcal{A}$ eventually gets trapped in one of $\mathcal{A}$'s safe components.

An automaton $\mathcal{A}$ is *good for games* (*GFG*, for short) if its nondeterminism can be resolved based on the past, thus on the prefix of the input word read so far. Formally, $\mathcal{A}$ is *GFG* if there exists a *strategy* $f : \Sigma^* \to Q$ such that the following holds:

1. The strategy f is consistent with the transition function. That is, for every finite word $u \in \Sigma^*$ and letter $\sigma \in \Sigma$, we have that $\langle f(u), \sigma, f(u \cdot \sigma) \rangle \in \Delta$.
2. Following f causes $\mathcal{A}$ to accept all the words in its language. That is, for every infinite word $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$, if $w \in L(\mathcal{A})$, then the run $f(w[1,0]), f(w[1,1]), f(w[1,2]), \ldots$, which we denote by $f(w)$, is accepting.

We say that the strategy f *witnesses* $\mathcal{A}$'s GFGness. For an automaton $\mathcal{A}$, we say that a state q of $\mathcal{A}$ is *GFG* if $\mathcal{A}^q$ is GFG. Finally, we say that a GFG-tNCW $\mathcal{A}$ is *minimal* if for every equivalent GFG-tNCW $\mathcal{B}$, it holds that $|\mathcal{A}| \leq |\mathcal{B}|$.

Consider a directed graph $G = \langle V, E \rangle$. A *strongly connected set* in G (SCS, for short) is a set $C \subseteq V$ such that for every two vertices $v, v' \in C$, there is a path from v to v'. A SCS is *maximal* if it is maximal w.r.t containment, that is, for every non-empty set $C' \subseteq V \setminus C$, it holds that $C \cup C'$ is not a SCS. The *maximal strongly connected sets* are also termed *strongly connected components* (SCCs, for short). The *SCC graph of* G is the graph defined over the SCCs of G, where there is an edge from a SCC C to another SCC C' iff there are two vertices $v \in C$ and $v' \in C'$ with $\langle v, v' \rangle \in E$. A SCC is *ergodic* iff it has no outgoing edges in the SCC graph. The SCC graph of G can be computed in linear time by standard SCC algorithms [35]. An automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$ induces a directed graph $G_\mathcal{A} = \langle Q, E \rangle$, where $\langle q, q' \rangle \in E$ iff there is a letter $\sigma \in \Sigma$ such that $\langle q, \sigma, q' \rangle \in \Delta$. The SCSs and SCCs of $\mathcal{A}$ are those of $G_\mathcal{A}$. We say that a tNCW $\mathcal{A}$ is *normal* if all the safe components of $\mathcal{A}$ are SCSs. That is, for all states q and s of $\mathcal{A}$, if there is a path of $\bar{\alpha}$-transition from q to s, then there is also a path of $\bar{\alpha}$-transition from s to q.

We now combine several properties defined above and say that a GFG-tNCW $\mathcal{A}$ is *nice* if all the states in $\mathcal{A}$ are reachable and GFG, and $\mathcal{A}$ is normal, safe deterministic, and semantically deterministic. In the theorem below we combine arguments from [17] showing that each of these properties can be obtained in at most polynomial time, and without the properties being conflicting. For some properties, we give an alternative and simpler proof.

▶ **Theorem 2.** [17] *Every GFG-tNCW $\mathcal{A}$ can be turned, in polynomial time, into an equivalent nice GFG-tNCW $\mathcal{B}$ such that $|\mathcal{B}| \leq |\mathcal{A}|$.*

Proof. It is shown in [17] that one can decide the GFGness of a tNCW $\mathcal{A}$ in polynomial time. The proof goes through an intermediate step where the authors construct a two-players game such that if the first player does not win the game, then $\mathcal{A}$ is not GFG, and otherwise a winning strategy for him induces a safe-deterministic GFG-tNCW $\mathcal{B}$ equivalent to $\mathcal{A}$. As we start with a GFG-tNCW $\mathcal{A}$, such a winning strategy is guaranteed to exist, and we obtain an equivalent safe-deterministic GFG-tNCW $\mathcal{B}$ in polynomial time. In fact, it can be shown that $\mathcal{B}$ is also semantically deterministic. Yet, for completeness we give below a general procedure for semantic determinization.

For a tNCW $\mathcal{A}$, we say that a transition $\langle q, \sigma, s \rangle \in \Delta$ is *covering* if for every transition $\langle q, \sigma, s' \rangle$, it holds that $L(\mathcal{A}^{s'}) \subseteq L(\mathcal{A}^s)$. If $\mathcal{A}$ is GFG and f is a strategy witnessing its GFGness, we say that a state q of $\mathcal{A}$ is *used by* f if there is a finite word u with $f(u) = q$, and

we say that a transition $\langle q, \sigma, q' \rangle$ of $\mathcal{A}$ is *used by* f if there is a finite word u with $f(u) = q$ and $f(u\sigma) = q'$. Since states that are not GFG can be detected in polynomial time, and as all states that are used by a strategy that witnesses $\mathcal{B}$'s GFGness are GFG, the removal of non-GFG states does not affect $\mathcal{B}$'s language. Note that removing the non-GFG states may result in a non-total automaton, in which case we add a rejecting sink. Now, using the fact that language containment of GFG-tNCWs can be checked in polynomial time [12, 17], and transitions that are used by strategies are covering [17], one can semantically determinize $\mathcal{B}$ by removing non-covering transitions.

States that are not reachable are easy to detect, and their removal does not affect $\mathcal{B}$'s language. Normalization is also easy to obtain and involves adding some existing transitions to α [17]. Indeed, if the safe components of $\mathcal{B}$ are not SCSs, then every $\bar{\alpha}$-transition connecting different SCCs of $\mathcal{B}$'s safe components can be added to α without affecting the acceptance of runs in $\mathcal{B}$, as every accepting run traverses such transitions only finitely often. Thus, the language and GFGness of all states are not affected. Finally, it is not hard to verify that the properties, in the order we obtain them in the proof, are not conflicting, and thus the described sequence of transformations results in a nice GFG-tNCW. ◀

3 A Sufficient Condition for GFG-tNCW Minimality

In this section, we define two additional properties for nice GFG-tNCWs, namely *safe-centralized* and *safe-minimal*, and we prove that nice GFG-tNCWs that attain these properties are minimal. In Sections 4 – 5, we are going to show that the two properties can be attained in polynomial time. Before we start, let us note that a GFG-tNCW may be nice and still not be minimal. A simple example is a GFG-tNCW $\mathcal{A}_{\mathsf{fm}}$ for the language $(a+b)^* \cdot a^\omega$ that has two states, both with a $\bar{\alpha}$-self-loop labeled a and an α-transition labeled b to the other state. It is easy to see that $\mathcal{A}_{\mathsf{fm}}$ is nice but not minimal.

Consider a tNCW $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$. A run r of $\mathcal{A}$ is *safe* if it does not traverse α-transitions. The *safe language* of $\mathcal{A}$, denoted $L_{safe}(\mathcal{A})$, is the set of infinite words w, such that there is a safe run of $\mathcal{A}$ on w. Recall that two states $q, s \in Q$ are equivalent ($q \sim_{\mathcal{A}} s$) if $L(\mathcal{A}^q) = L(\mathcal{A}^s)$. Then, q and s are *strongly-equivalent*, denoted $q \approx_{\mathcal{A}} s$, if $q \sim_{\mathcal{A}} s$ and $L_{safe}(\mathcal{A}^q) = L_{safe}(\mathcal{A}^s)$. Finally, q is *subsafe-equivalent to* s, denoted $q \precsim_{\mathcal{A}} s$, if $q \sim_{\mathcal{A}} s$ and $L_{safe}(\mathcal{A}^q) \subseteq L_{safe}(\mathcal{A}^s)$. Note that the three relations are transitive. When $\mathcal{A}$ is clear from the context, we omit it from the notations, thus write $L_{safe}(q)$, $q \precsim s$, etc. The tNCW $\mathcal{A}$ is *safe-minimal* if it has no strongly-equivalent states. Then, $\mathcal{A}$ is *safe-centralized* if for every two states $q, s \in Q$, if $q \precsim s$, then q and s are in the same safe component of $\mathcal{A}$.

▶ **Example 3.** The nice GFG-tNCW $\mathcal{A}_{\mathsf{fm}}$ described above is neither safe-minimal (its two states are strongly-equivalent) nor safe-centralized (its two states are in different safe components). As another example, consider the tDCW $\mathcal{A}$ appearing in Figure 1. The dashed transitions are α-transitions. All the states of $\mathcal{A}$ are equivalent, yet they all differ in their safe language. Accordingly, $\mathcal{A}$ is safe-minimal. Since $a^\omega = L_{safe}(\mathcal{A}^{q_2}) \subseteq L_{safe}(\mathcal{A}^{q_0})$, we have that $q_2 \precsim q_0$. Hence, as q_0 and q_2 are in different safe components, the tDCW $\mathcal{A}$ is not safe-centralized.

▶ **Proposition 4.** *Consider a nice GFG-tNCW $\mathcal{A}$ and states q and s of $\mathcal{A}$ such that $q \approx s$ ($q \precsim s$). For every letter $\sigma \in \Sigma$ and $\bar{\alpha}$-transition $\langle q, \sigma, q' \rangle$, there is an $\bar{\alpha}$-transition $\langle s, \sigma, s' \rangle$ such that $q' \approx s'$ ($q' \precsim s'$, respectively).*

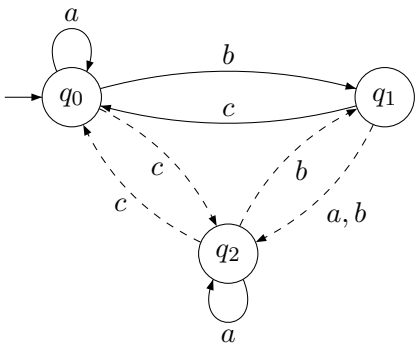


Figure 1 The tDCW $\mathcal{A}$.

Proof. We prove the proposition for the case $q \approx s$. The case $q \precsim s$ is similar. Since $\mathcal{A}$ is normal, the existence of the $\bar{\alpha}$-transition $\langle q, \sigma, q' \rangle$ implies that there is a safe run from q' back to q. Hence, there is a word $z \in L_{safe}(\mathcal{A}^{q'})$. Clearly, $\sigma \cdot z$ is in $L_{safe}(\mathcal{A}^q)$. Now, since $q \approx s$, we have that $L_{safe}(\mathcal{A}^q) = L_{safe}(\mathcal{A}^s)$. In particular, $\sigma \cdot z \in L_{safe}(\mathcal{A}^s)$, and thus there is a $\bar{\alpha}$-transition $\langle s, \sigma, s' \rangle$. We prove that $q' \approx s'$. Since $L(\mathcal{A}^q) = L(\mathcal{A}^s)$ and $\mathcal{A}$ is semantically deterministic, then, by Proposition 1, we have that $L(\mathcal{A}^{q'}) = L(\mathcal{A}^{s'})$. It is left to prove that $L_{safe}(\mathcal{A}^{q'}) = L_{safe}(\mathcal{A}^{s'})$. We prove that $L_{safe}(\mathcal{A}^{q'}) \subseteq L_{safe}(\mathcal{A}^{s'})$. The second direction is similar. Since $\mathcal{A}$ is safe deterministic, the transition $\langle s, \sigma, s' \rangle$ is the only σ-labeled $\bar{\alpha}$-transition from s. Hence, if by contradiction there is a word $z \in L_{safe}(\mathcal{A}^{q'}) \setminus L_{safe}(\mathcal{A}^{s'})$, we get that $\sigma \cdot z \in L_{safe}(\mathcal{A}^q) \setminus L_{safe}(\mathcal{A}^s)$, contradicting the fact that $L_{safe}(\mathcal{A}^q) = L_{safe}(\mathcal{A}^s)$. ◀

We continue with propositions that relate two automata, $\mathcal{A} = \langle \Sigma, Q_\mathcal{A}, q^0_\mathcal{A}, \delta_\mathcal{A}, \alpha_\mathcal{A} \rangle$ and $\mathcal{B} = \langle \Sigma, Q_\mathcal{B}, q^0_\mathcal{B}, \delta_\mathcal{B}, \alpha_\mathcal{B} \rangle$. We assume that $Q_\mathcal{A}$ and $Q_\mathcal{B}$ are disjoint, and extend the $\sim$, $\approx$, and $\precsim$ relations to states in $Q_\mathcal{A} \cup Q_\mathcal{B}$ in the expected way. For example, for $q \in Q_\mathcal{A}$ and $s \in Q_\mathcal{B}$, we use $q \sim s$ to indicate that $L(\mathcal{A}^q) = L(\mathcal{B}^s)$.

▶ **Proposition 5.** *Let $\mathcal{A}$ and $\mathcal{B}$ be equivalent nice GFG-tNCWs. For every state $q \in Q_\mathcal{A}$, there is a state $s \in Q_\mathcal{B}$ such that $q \precsim s$.*

Proof. Let g be a strategy witnessing $\mathcal{B}$'s GFGness. Consider a state $q \in Q_\mathcal{A}$. Let $u \in \Sigma^*$ be such that $q \in \delta_\mathcal{A}(q^0_\mathcal{A}, u)$. Since $\mathcal{A}$ and $\mathcal{B}$ are equivalent and semantically deterministic, an iterative application of Proposition 1 implies that for every state $q' \in \delta_\mathcal{B}(q^0_\mathcal{B}, u)$, we have $q \sim q'$. In particular, $q \sim g(u)$. If $L_{safe}(\mathcal{A}^q) = \emptyset$, then we are done, as $L_{safe}(\mathcal{A}^q) \subseteq L_{safe}(\mathcal{B}^{g(u)})$. If $L_{safe}(\mathcal{A}^q) \neq \emptyset$, then the proof proceeds as follows. Assume by way of contradiction that for every state $s \in Q_\mathcal{B}$ that is equivalent to q, it holds that $L_{safe}(\mathcal{A}^q) \not\subseteq L_{safe}(\mathcal{B}^s)$. We define an infinite word z such that $\mathcal{A}$ accepts $u \cdot z$, yet $g(u \cdot z)$ is a rejecting run of $\mathcal{B}$. Since $\mathcal{A}$ and $\mathcal{B}$ are equivalent, this contradicts the fact that g witnesses $\mathcal{B}$'s GFGness.

We define z as follows. Let $s_0 = g(u)$. Since $L_{safe}(\mathcal{A}^q) \not\subseteq L_{safe}(\mathcal{B}^{s_0})$, there is a finite nonempty word z_1 such that there is a safe run of $\mathcal{A}^q$ on z_1, but every run of $\mathcal{B}^{s_0}$ on z_1 is not safe. In particular, the run of $\mathcal{B}^{s_0}$ that is induced by g, namely $g(u), g(u \cdot z_1[1,1]), g(u \cdot z_1[1,2]), \ldots, g(u \cdot z_1)$, traverses an α-transition. Since $\mathcal{A}$ is normal, we can define z_1 so the safe run of $\mathcal{A}^q$ on z_1 ends in q. Let $s_1 = g(u \cdot z_1)$. We have so far two finite runs: $q \xrightarrow{z_1} q$ and $s_0 \xrightarrow{z_1} s_1$, where the first run is safe, and the second is not. Now, since $q \sim s_0$, then again by Proposition 1 we have that $q \sim s_1$, and by applying the same considerations, we can define a finite nonempty word z_2 and $s_2 = g(u \cdot z_1 \cdot z_2)$ such that $q \xrightarrow{z_2} q$ and $s_1 \xrightarrow{z_2} s_2$, where the first run is safe, and the second is not. After at most $|Q_\mathcal{B}|$ iterations, we get that there are

$0 \leq j_1 < j_2 \leq |Q_\mathcal{B}|$ such that $s_{j_1} = s_{j_2}$, and define $z = z_1 \cdots z_2 \cdots z_{j_1} \cdot (z_{j_1+1} \cdots z_{j_2})^\omega$. Since $j_1 < j_2$, the extension $z_{j_1+1} \cdots z_{j_2}$ is nonempty and thus z is infinite. On the one hand, since $q \in \delta_\mathcal{A}(q^0_\mathcal{A}, u)$ and there is a safe run of $\mathcal{A}^q$ on z, we have that $u \cdot z \in L(\mathcal{A})$. On the other hand, the run $g(u \cdot z)$ traverses an α-transitions infinitely often, and is thus rejecting. ◀

▶ **Proposition 6.** *Let $\mathcal{A}$ and $\mathcal{B}$ be equivalent nice GFG-tNCWs. For every state $p \in Q_\mathcal{A}$, there are states $q \in Q_\mathcal{A}$ and $s \in Q_\mathcal{B}$ such that $p \precsim q$ and $q \approx s$.*

Proof. The proposition follows from the combination of Proposition 5 with the transitivity of $\precsim$ and the fact $Q_\mathcal{A}$ and $Q_\mathcal{B}$ are finite. Formally, consider the directed bipartite graph $G = \langle Q_\mathcal{A} \cup Q_\mathcal{B}, E\rangle$, where $E \subseteq (Q_\mathcal{A} \times Q_\mathcal{B}) \cup (Q_\mathcal{B} \times Q_\mathcal{A})$ is such that $\langle p_1, p_2\rangle \in E$ iff $p_1 \precsim p_2$. Proposition 5 implies that E is total. That is, from every state in $Q_\mathcal{A}$ there is an edge to some state in $Q_\mathcal{B}$, and from every state in $Q_\mathcal{B}$ there is an edge to some state in $Q_\mathcal{A}$. Since $Q_\mathcal{A}$ and $Q_\mathcal{B}$ are finite, this implies that for every $p \in Q_\mathcal{A}$, there is a path in G that starts in p and reaches a state $q \in Q_\mathcal{A}$ (possibly $q = p$) that belongs to a nonempty cycle. We take s to be some state in $Q_\mathcal{B}$ in this cycle. By the transitivity of $\precsim$, we have that $p \precsim q$, $q \precsim s$, and $s \precsim q$. The last two imply that $q \approx s$, and we are done. ◀

▶ **Lemma 7.** *Consider a nice GFG-tNCW $\mathcal{A}$. If $\mathcal{A}$ is safe-centralized and safe-minimal, then for every nice GFG-tNCW $\mathcal{B}$ equivalent to $\mathcal{A}$, there is an injection $\eta : S(\mathcal{A}) \to S(\mathcal{B})$ such that for every safe component $T \in S(\mathcal{A})$, it holds that $|T| \leq |\eta(T)|$.*

Proof. We define η as follows. Consider a safe component $T \in S(\mathcal{A})$. Let p_T be some state in T. By Proposition 6, there are states $q_T \in Q_\mathcal{A}$ and $s_T \in Q_\mathcal{B}$ such that $p_T \precsim q_T$ and $q_T \approx s_T$. Since $\mathcal{A}$ is safe-centralized, the states p_T and q_T are in the same safe component, thus $q_T \in T$. We define $\eta(T)$ to be the safe component of s_T in $\mathcal{B}$. We show that η is an injection; that is, for every two safe components T_1 and T_2 in $S(\mathcal{A})$, it holds that $\eta(T_1) \neq \eta(T_2)$. Assume by way of contradiction that T_1 and T_2 are such that s_{T_1} and s_{T_2}, chosen as described above, are in the same safe component of $\mathcal{B}$. Then, there is a safe run from s_{T_1} to s_{T_2}. Since $s_{T_1} \approx q_{T_1}$, an iterative application of Proposition 4 implies that there is a safe run from q_{T_1} to some state q such that $q \approx s_{T_2}$. Since the run from q_{T_1} to q is safe, the states q_{T_1} and q are in the same safe component, and so $q \in T_1$. Since $q_{T_2} \approx s_{T_2}$, then $q \approx q_{T_2}$. Since $\mathcal{A}$ is safe-centralized, the latter implies that q and q_{T_2} are in the same safe component, and so $q \in T_2$, and we have reached a contradiction.

It is left to prove that for every safe component $T \in S(\mathcal{A})$, it holds that $|T| \leq |\eta(T)|$. Let $T \in S(\mathcal{A})$ be a safe component of $\mathcal{A}$. By the definition of η, there are $q_T \in T$ and $s_T \in \eta(T)$ such that $q_T \approx s_T$. Since $\mathcal{A}$ is normal, there is a safe run $q_0, q_1, \ldots q_m$ of $\mathcal{A}$ that starts in q_T and traverses all the states in T. Since $\mathcal{A}$ is safe-minimal, no two states in T are strongly equivalent. Therefore, there is a subset $I \subseteq \{0, 1, \ldots, m\}$ of indices, with $|I| = |T|$, such that for every two different indices $i_1, i_2 \in I$, it holds that $q_{i_1} \not\approx q_{i_2}$. By applying Proposition 4 iteratively, there is a safe run $s_0, s_1, \ldots s_m$ of $\mathcal{B}$ that starts in s_T and such that for every $0 \leq i \leq m$, it holds that $q_i \approx s_i$. Since the run is safe, it stays in $\eta(T)$. Then, however, for every two different indices $i_1, i_2 \in I$, we have that $s_{i_1} \not\approx s_{i_2}$, and so $s_{i_1} \neq s_{i_2}$. Hence, $|\eta(T)| \geq |I| = |T|$. ◀

We can now prove that the additional two properties imply the minimality of nice GFG-tNCWs.

▶ **Theorem 8.** *Consider a nice GFG-tNCW $\mathcal{A}$. If $\mathcal{A}$ is safe-centralized and safe-minimal, then $\mathcal{A}$ is a minimal GFG-tNCW for $L(\mathcal{A})$.*

Proof. Let $\mathcal{B}$ be a GFG-tNCW equivalent to $\mathcal{A}$. By Theorem 2, we can assume that $\mathcal{B}$ is nice. Indeed, otherwise we can make it nice without increasing its state space. Then, by Lemma 7, there is an injection $\eta : S(\mathcal{A}) \to S(\mathcal{B})$ such that for every safe component $T \in S(\mathcal{A})$, it holds that $|T| \leq |\eta(T)|$. Hence,

$$|\mathcal{A}| = \sum_{T \in S(\mathcal{A})} |T| \leq \sum_{T \in S(\mathcal{A})} |\eta(T)| \leq \sum_{T' \in S(\mathcal{B})} |T'| = |\mathcal{B}|.$$

Indeed, the first inequality follows from the fact $|T| \leq |\eta(T)|$, and the second inequality follows from the fact that η is injective. ◀

▶ Remark 9. Recall that we assume that the transition function of GFG-tNCWs is total. Clearly, a non-total GFG-tNCW can be made total by adding a rejecting sink. One may wonder whether the additional state that this process involves interferes with our minimality proof. The answer is negative: if $\mathcal{B}$ in Theorem 8 is not total, then, by Proposition 5, $\mathcal{A}$ has a state s such that $q_{rej} \precsim s$, where q_{rej} is a rejecting sink we need to add to $\mathcal{B}$ if we want to make it total. Thus, $L(\mathcal{A}^s) = \emptyset$, and we may not count it if we allow GFG-tNCWs without a total transition function.

4 Safe Centralization

Consider a nice GFG-tNCW $\mathcal{A} = \langle \Sigma, Q_{\mathcal{A}}, q^0_{\mathcal{A}}, \delta_{\mathcal{A}}, \alpha_{\mathcal{A}} \rangle$. Recall that $\mathcal{A}$ is safe-centralized if for every two states $q, s \in Q_{\mathcal{A}}$, if $q \precsim s$, then q and s are in the same safe component. In this section we describe how to turn a given nice GFG-tNCW into a nice safe-centralized GFG-tNCW. The resulted tNCW is also going to be *α-homogenous*: for every state $q \in Q_{\mathcal{A}}$ and letter $\sigma \in \Sigma$, either $\delta^{\alpha}_{\mathcal{A}}(q, \sigma) = \emptyset$ or $\delta^{\bar{\alpha}}_{\mathcal{A}}(q, \sigma) = \emptyset$.

Let $H \subseteq S(\mathcal{A}) \times S(\mathcal{A})$ be such that for all safe components $S, S' \in S(\mathcal{A})$, we have that $H(S, S')$ iff there exist states $q \in S$ and $q' \in S'$ such that $q \precsim q'$. That is, when $S \neq S'$, then the states q and q' witness that $\mathcal{A}$ is not safe-centralized. Recall that $q \precsim q'$ iff $L(\mathcal{A}^q) = L(\mathcal{A}^{q'})$ and $L_{safe}(\mathcal{A}^q) \subseteq L_{safe}(\mathcal{A}^{q'})$. Since language containment for GFG-tNCWs can be checked in polynomial time [12, 17], the first condition can be checked in polynomial time. Since $\mathcal{A}$ is safe deterministic, the second condition reduces to language containment between deterministic automata and can also be checked in polynomial time. Hence, the relation H can be computed in polynomial time.

▶ **Lemma 10.** *Consider safe components $S, S' \in S(\mathcal{A})$ such that $H(S, S')$. Then, for every $p \in S$ there is $p' \in S'$ such that $p \precsim p'$.*

Proof. Since $H(S, S')$, then, by definition, there are states $q \in S$ and $q' \in S'$ such that $q \precsim q'$. Let p be a state in S. Since $\mathcal{A}$ is normal, there is a safe run from q to p in S. Since $q \precsim q'$, an iterative application of Proposition 4 implies that there is a safe run from q' to some state p' in S' for which $p \precsim p'$, and we are done. ◀

▶ **Lemma 11.** *The relation H is transitive: for every safe components $S, S', S'' \in S(\mathcal{A})$, if $H(S, S')$ and $H(S', S'')$, then $H(S, S'')$.*

Proof. Let $S, S', S'' \in S(\mathcal{A})$ be safe components of $\mathcal{A}$ such that $H(S, S')$ and $H(S', S'')$. Since, $H(S, S')$, there are states $q \in S$ and $q' \in S'$ such that $q \precsim q'$. Now, since $H(S', S'')$, we get by Lemma 10 that that for all states in S', in particular for q', there is a state $q'' \in S''$ such that $q' \precsim q''$. The transitivity of $\precsim$ then implies that $q \precsim q''$, and so $H(S, S'')$. ◀

We say that a set $\mathcal{S} \subseteq S(\mathcal{A})$ is a *frontier of* $\mathcal{A}$ if for every safe component $S \in S(\mathcal{A})$, there is a safe component $S' \in \mathcal{S}$ with $H(S, S')$, and for all safe components $S, S' \in \mathcal{S}$ such that $S \neq S'$, we have that $\neg H(S, S')$ and $\neg H(S', S)$. Once H is calculated, a frontier of $\mathcal{A}$ can be found in linear time. For example, as H is transitive, we can take one vertex from each ergodic SCC in the graph $\langle S(\mathcal{A}), H\rangle$. Note that all frontiers of $\mathcal{A}$ are of the same size, namely the number of ergodic SCCs in this graph.

Given a frontier $\mathcal{S}$ of $\mathcal{A}$, we define the automaton $\mathcal{B}_\mathcal{S} = \langle \Sigma, Q_\mathcal{S}, q^0_\mathcal{S}, \delta_\mathcal{S}, \alpha_\mathcal{S}\rangle$, where $Q_\mathcal{S} = \{q \in Q_\mathcal{A} : q \in S \text{ for some } S \in \mathcal{S}\}$, and the other components are defined as follows. The initial state $q^0_\mathcal{S}$ is chosen such that $q^0_\mathcal{S} \sim_\mathcal{A} q^0_\mathcal{A}$. Specifically, if $q^0_\mathcal{A} \in Q_\mathcal{S}$, we take $q^0_\mathcal{S} = q^0_\mathcal{A}$. Otherwise, by Lemma 10 and the definition of $\mathcal{S}$, there is a state $q' \in Q_\mathcal{S}$ such that $q^0_\mathcal{A} \precsim q'$, and we take $q^0_\mathcal{S} = q'$. The transitions in $\mathcal{B}_\mathcal{S}$ are either $\bar{\alpha}$-transitions of $\mathcal{A}$, or α-transitions that we add among the safe components in $\mathcal{S}$ in a way that preserves language equivalence. Formally, consider a state $q \in Q_\mathcal{S}$ and a letter $\sigma \in \Sigma$. If $\delta^{\bar{\alpha}}_\mathcal{A}(q, \sigma) \neq \emptyset$, then $\delta^{\bar{\alpha}}_\mathcal{S}(q, \sigma) = \delta^{\bar{\alpha}}_\mathcal{A}(q, \sigma)$ and $\delta^{\alpha}_\mathcal{S}(q, \sigma) = \emptyset$. If $\delta^{\bar{\alpha}}_\mathcal{A}(q, \sigma) = \emptyset$, then $\delta^{\bar{\alpha}}_\mathcal{S}(q, \sigma) = \emptyset$ and $\delta^{\alpha}_\mathcal{S}(q, \sigma) = \{q' \in Q_\mathcal{S} :$ there is $q'' \in \delta^{\alpha}_\mathcal{A}(q, \sigma)$ such that $q' \sim_\mathcal{A} q''\}$. Note that $\mathcal{B}_\mathcal{S}$ is α-homogenous.

▶ **Example 12.** Consider the tDCW $\mathcal{A}$ appearing in Figure 1. Recall that the dashed transitions are α-transitions. Since $\mathcal{A}$ is normal and deterministic, it is nice. By removing the α-transitions of $\mathcal{A}$, we get the safe components described in in Figure 2. Since $q_2 \precsim q_0$, we have that $\mathcal{A}$ has a single frontier $\mathcal{S} = \{\{q_0, q_1\}\}$. The automaton $\mathcal{B}_\mathcal{S}$ appears in Figure 3. As all the states of $\mathcal{A}$ are equivalent, we direct a σ-labeled α-transition to q_0 and to q_1, for every state with no σ-labeled transition in $\mathcal{S}$.

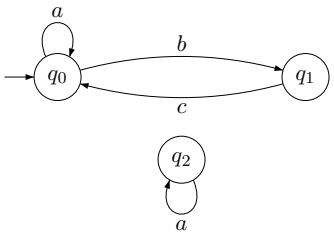


Figure 2 The safe components of $\mathcal{A}$.

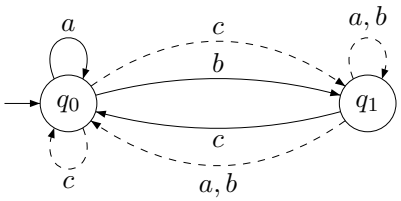


Figure 3 The tNCW $\mathcal{B}_{\{\{q_0,q_1\}\}}$.

We extend Proposition 1 to the setting of $\mathcal{A}$ and $\mathcal{B}_\mathcal{S}$:

▶ **Proposition 13.** *Consider states q and s of $\mathcal{A}$ and $\mathcal{B}_\mathcal{S}$, respectively, a letter $\sigma \in \Sigma$, and transitions $\langle q, \sigma, q'\rangle$ and $\langle s, \sigma, s'\rangle$ of $\mathcal{A}$ and $\mathcal{B}_\mathcal{S}$, respectively. If $q \sim_\mathcal{A} s$, then $q' \sim_\mathcal{A} s'$.*

Proof. If $\langle s, \sigma, s'\rangle$ is an $\bar{\alpha}$-transition of $\mathcal{B}_\mathcal{S}$, then, by the definition of $\Delta_\mathcal{S}$, it is also an $\bar{\alpha}$-transition of $\mathcal{A}$. Hence, since $q \sim_\mathcal{A} s$ and $\mathcal{A}$ is nice, in particular, semantically deterministic, we get by Proposition 1 that $q' \sim_\mathcal{A} s'$. If $\langle s, \sigma, s'\rangle$ is an α-transition of $\mathcal{B}_\mathcal{S}$, then, by the definition of $\Delta_\mathcal{S}$, there is some $s'' \in \delta_\mathcal{A}(s, \sigma)$ with $s' \sim_\mathcal{A} s''$. Again, since $q \sim_\mathcal{A} s$ and $\mathcal{A}$ is semantically deterministic, we have by Proposition 1 that $s'' \sim_\mathcal{A} q'$, and thus $s' \sim_\mathcal{A} q'$. ◀

▶ **Proposition 14.** *Let q and s be states of $\mathcal{A}$ and $\mathcal{B}_\mathcal{S}$, respectively, with $q \sim_\mathcal{A} s$. It holds that $\mathcal{B}^s_\mathcal{S}$ is a GFG-tNCW equivalent to $\mathcal{A}^q$.*

Proof. We first prove that $L(\mathcal{B}^s_\mathcal{S}) \subseteq L(\mathcal{A}^q)$. Consider a word $w = \sigma_1\sigma_2 \ldots \in L(\mathcal{B}^s_\mathcal{S})$. Let $s_0, s_1, s_2, \ldots$ be an accepting run of $\mathcal{B}^s_\mathcal{S}$ on w. Then, there is $i \geq 0$ such that $s_i, s_{i+1}, \ldots$ is a safe run of $\mathcal{B}^{s_i}_\mathcal{S}$ on the suffix $w[i+1, \infty]$. Let $q_0, q_1, \ldots q_i$ be a run of $\mathcal{A}^q$ on the prefix $w[1, i]$.

Since $q_0 \sim_{\mathcal{A}} s_0$, we get, by an iterative application of Proposition 13, that $q_i \sim_{\mathcal{A}} s_i$. In addition, as the run of $\mathcal{B}_{\mathcal{S}}^{s_i}$ on the suffix $w[i+1,\infty]$ is safe, it is also a safe run of $\mathcal{A}^{s_i}$. Hence, $w[i+1,\infty] \in L(\mathcal{A}^{q_i})$, and thus $q_0, q_1, \ldots, q_i$ can be extended to an accepting run of $\mathcal{A}^q$ on w.

Next, we prove that $L(\mathcal{A}^q) \subseteq L(\mathcal{B}_{\mathcal{S}}^s)$ and that $\mathcal{B}_{\mathcal{S}}^s$ is a GFG-tNCW. We do this by defining a strategy $g : \Sigma^* \to Q_{\mathcal{S}}$ such that for all words $w \in L(\mathcal{A}^q)$, we have that $g(w)$ is an accepting run of $\mathcal{B}_{\mathcal{S}}^s$ on w. First, $g(\epsilon) = s$. Then, for $u \in \Sigma^*$ and $\sigma \in \Sigma$, we define $g(u \cdot \sigma)$ as follows. Recall that $\mathcal{A}$ is nice. So, in particular, $\mathcal{A}^q$ is GFG. Let f be a strategy witnessing $\mathcal{A}^q$'s GFGness. If $\delta_{\mathcal{S}}^{\bar{\alpha}}(g(u),\sigma) \neq \emptyset$, then $g(u \cdot \sigma) = q'$ for some $q' \in \delta_{\mathcal{S}}^{\bar{\alpha}}(g(u),\sigma)$. If $\delta_{\mathcal{S}}^{\bar{\alpha}}(g(u),\sigma) = \emptyset$, then $g(u \cdot \sigma) = q'$ for some state $q' \in Q_{\mathcal{S}}$ such that $f(u \cdot \sigma) \precsim_{\mathcal{A}} q'$. Note that since $\mathcal{S}$ is a frontier, such a state q' exists. We prove that g is consistent with $\Delta_{\mathcal{S}}$. In fact, we prove a stronger claim, namely for all $u \in \Sigma^*$ and $\sigma \in \Sigma$, we have that $f(u) \sim_{\mathcal{A}} g(u)$ and $\langle g(u), \sigma, g(u \cdot \sigma)\rangle \in \Delta_{\mathcal{S}}$.

The proof proceeds by an induction on $|u|$. For this induction base, as $f(\epsilon) = q$, $g(\epsilon) = s$, and $q \sim_{\mathcal{A}} s$, we are done. Given u and σ, consider a transition $\langle g(u), \sigma, s'\rangle \in \Delta_{\mathcal{S}}$. Since $\mathcal{B}_{\mathcal{S}}$ is total, such a transition exists. We distinguish between two cases. If $\delta_{\mathcal{S}}^{\bar{\alpha}}(g(u),\sigma) \neq \emptyset$, then, as $\mathcal{B}_{\mathcal{S}}$ is α-homogenous and safe deterministic, the state s' is the only state in $\delta_{\mathcal{S}}^{\bar{\alpha}}(g(u),\sigma)$. Hence, by the definition of g, we have that $g(u \cdot \sigma) = s'$ and so $\langle g(u), \sigma, g(u \cdot \sigma)\rangle \in \Delta_{\mathcal{S}}$. If $\delta_{\mathcal{S}}^{\bar{\alpha}}(g(u),\sigma) = \emptyset$, we claim that $g(u \cdot \sigma) \sim_{\mathcal{A}} s'$ Then, as $s' \in \delta_{\mathcal{S}}^{\alpha}(g(u),\sigma)$, the definition of $\Delta_{\mathcal{S}}$ for the case $\delta_{\mathcal{S}}^{\bar{\alpha}}(g(u),\sigma) = \emptyset$ implies that $\langle g(u), \sigma, g(u \cdot \sigma)\rangle \in \Delta_{\mathcal{S}}$. By the induction hypothesis, we have that $f(u) \sim_{\mathcal{A}} g(u)$. Hence, as $\langle f(u), \sigma, f(u \cdot \sigma))\rangle \in \delta_{\mathcal{A}}$ and $\langle g(u), \sigma, s'\rangle \in \Delta_{\mathcal{S}}$, we have, by Proposition 13, that $f(u \cdot \sigma) \sim_{\mathcal{A}} s'$. Recall that g is defined so that $f(u \cdot \sigma) \precsim_{\mathcal{A}} g(u \cdot \sigma)$. In particular, $f(u \cdot \sigma) \sim_{\mathcal{A}} g(u \cdot \sigma)$. Hence, by transitivity of $\sim_{\mathcal{A}}$, we have that $g(u \cdot \sigma) \sim_{\mathcal{A}} s'$. In addition, by the induction hypothesis, we have that $f(u) \sim_{\mathcal{A}} g(u)$, and so, in both cases, Proposition 13 implies that $f(u \cdot \sigma) \sim_{\mathcal{A}} g(u \cdot \sigma)$.

It is left to prove that for every infinite word $w = \sigma_1\sigma_2 \ldots \in \Sigma^\omega$, if $w \in L(\mathcal{A}^q)$, then $g(w)$ is accepting. Assume that $w \in L(\mathcal{A}^q)$ and consider the run $f(w)$ of $\mathcal{A}^q$ on w. Since $f(w)$ is accepting, there is $i \geq 0$ such that $f(w[1,i]), f(w[1,i+1]) \ldots$ is a safe run of $\mathcal{A}^{f(w[1,i])}$ on the suffix $w[i+1,\infty]$. We prove that $g(w)$ may traverse at most one α-transition when it reads the suffix $w[i+1,\infty]$. Assume that there is some $j \geq i$ such that $\langle g(w[1,j]), \sigma_{j+1}, g(w[1,j+1])\rangle \in \alpha_{\mathcal{S}}$. Then, by g's definition, we have that $f(w[1,j+1]) \precsim_{\mathcal{A}} g(w[1,j+1])$. Therefore, as $\mathcal{B}_{\mathcal{S}}$ follows the safe components in $\mathcal{S}$, we have that $L_{safe}(\mathcal{A}^{f(w[1,j+1])}) \subseteq L_{safe}(\mathcal{A}^{g(w[1,j+1])}) = L_{safe}(\mathcal{B}_{\mathcal{S}}^{g(w[1,j+1])})$, and thus $w[j+2,\infty] \in L_{safe}(\mathcal{B}_{\mathcal{S}}^{g(w[1,j+1])})$. Since $\mathcal{B}_{\mathcal{S}}$ is α-homogenous and safe-deterministic, there is a single run of $\mathcal{B}_{\mathcal{S}}^{g(w[1,j+1])}$ on $w[j+2,\infty]$, and this is the run that g follows. Therefore, $g(w[1,j+1]), g(w[1,j+2]), \ldots$ is a safe run, and we are done. ◀

▶ **Proposition 15.** *For every frontier $\mathcal{S}$, the GFG-tNCW $\mathcal{B}_{\mathcal{S}}$ is nice, safe-centralized, and α-homogenous.*

Proof. It is easy to see that the fact $\mathcal{A}$ is nice implies that $\mathcal{B}_{\mathcal{S}}$ is normal and safe deterministic. It can be shown that all the states in $\mathcal{B}_{\mathcal{S}}$ are reachable, yet anyway states that are nonreachable are easy to detect and their removal affects neither $\mathcal{B}_{\mathcal{S}}$'s language nor its other properties. Finally, Proposition 14 implies that all its states are GFG. To conclude that $\mathcal{B}_{\mathcal{S}}$ is nice, we prove below that it is semantically deterministic. Consider transitions $\langle q, \sigma, s_1\rangle$ and $\langle q, \sigma, s_2\rangle$ in $\Delta_{\mathcal{S}}$. We need to show that $s_1 \sim_{\mathcal{B}_{\mathcal{S}}} s_2$. By the definition of $\Delta_{\mathcal{S}}$, there are transitions $\langle q, \sigma, s_1'\rangle$ and $\langle q, \sigma, s_2'\rangle$ in $\Delta_{\mathcal{A}}$ for states s_1' and s_2' such that $s_1 \sim_{\mathcal{A}} s_1'$ and $s_2 \sim_{\mathcal{A}} s_2'$. As $\mathcal{A}$ is semantically deterministic, we have that $s_1' \sim_{\mathcal{A}} s_2'$, thus by transitivity of $\sim_{\mathcal{A}}$, we get that $s_1 \sim_{\mathcal{A}} s_2$. Then, Proposition 14 implies that $L(\mathcal{A}^{s_1}) = L(\mathcal{B}_{\mathcal{S}}^{s_1})$ and $L(\mathcal{A}^{s_2}) = L(\mathcal{B}_{\mathcal{S}}^{s_2})$, and so we get that $s_1 \sim_{\mathcal{B}_{\mathcal{S}}} s_2$. Thus, $\mathcal{B}_{\mathcal{S}}$ is semantically deterministic.

As we noted in the definition of its transitions, $\mathcal{B}_\mathcal{S}$ is α-homogenous. It is thus left to prove that $\mathcal{B}_\mathcal{S}$ is safe-centralized. Let q and s be states of $\mathcal{B}_\mathcal{S}$ such that $q \preceq_{\mathcal{B}_\mathcal{S}} s$; that is, $L(\mathcal{B}^q_\mathcal{S}) = L(\mathcal{B}^s_\mathcal{S})$ and $L_{safe}(\mathcal{B}^q_\mathcal{S}) \subseteq L_{safe}(\mathcal{B}^s_\mathcal{S})$. Let $S, T \in \mathcal{S}$ be the safe components of q and s, respectively. We need to show that $S = T$. By Proposition 14, we have that $L(\mathcal{A}^q) = L(\mathcal{B}^q_\mathcal{S})$ and $L(\mathcal{A}^s) = L(\mathcal{B}^s_\mathcal{S})$. As $\mathcal{B}_\mathcal{S}$ follows the safe components in $\mathcal{S}$, we have that $L_{safe}(\mathcal{A}^q) = L_{safe}(\mathcal{B}^q_\mathcal{S})$ and $L_{safe}(\mathcal{A}^s) = L_{safe}(\mathcal{B}^s_\mathcal{S})$. Hence, $q \preceq_\mathcal{A} s$, implying $H(S, T)$. Since $\mathcal{S}$ is a frontier, this is possible only when $S = T$. ◀

▶ **Theorem 16.** *Every nice GFG-tNCW can be turned in polynomial time into an equivalent nice, safe-centralized, and α-homogenous GFG-tNCW.*

5 Safe Minimization

In the setting of finite words, a *quotient automaton* is obtained by merging equivalent states, and is guaranteed to be minimal. In the setting of co-Büchi automata, it may not be possible to define an equivalent language on top of the quotient automaton. For example, all the states in the GFG-tNCW $\mathcal{A}$ in Figure 1 are equivalent, and still it is impossible to define its language on top of a single-state tNCW. In this section we show that when we start with a nice, safe-centralized, and α-homogenous GFG-tNCW $\mathcal{B}$, the transition to a quotient automaton, namely merging of strongly-equivalent states, is well defined and results in a GFG-tNCW equivalent to $\mathcal{B}$ that attains all the helpful properties of $\mathcal{B}$, and is also safe minimal[4]. By Theorem 8, it is also minimal.

Consider a nice, safe-centralized, and α-homogenous GFG-tNCW $\mathcal{B} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$. For a state $q \in Q$, define $[q] = \{q' \in Q : q \approx_\mathcal{B} q'\}$. We define the tNCW $\mathcal{C} = \langle \Sigma, Q_\mathcal{C}, [q_0], \delta_\mathcal{C}, \alpha_\mathcal{C} \rangle$, where $Q_\mathcal{C} = \{[q] : q \in Q\}$, the transition function is such that $\langle [q], \sigma, [p] \rangle \in \Delta_\mathcal{C}$ iff there are $q' \in [q]$ and $p' \in [p]$ such that $\langle q', \sigma, p' \rangle \in \Delta$, and $\langle [q], \sigma, [p] \rangle \in \alpha_\mathcal{C}$ iff $\langle q', \sigma, p' \rangle \in \alpha$. Note that $\mathcal{B}$ being α-homogenous implies that $\alpha_\mathcal{C}$ is well defined; that is, independent of the choice of q' and p'. To see why, assume that $\langle q', \sigma, p' \rangle \in \bar{\alpha}$ and let q'' be a state in $[q]$. As $q' \approx_\mathcal{B} q''$, we have by Proposition 4 that there is $p'' \in [p]$ such that $\langle q'', \sigma, p'' \rangle \in \bar{\alpha}$. Thus, as $\mathcal{B}$ is α-homogenous, there is no σ-labeled α-transition from q'' to a state in $[p]$. Note that we have proved that if $\langle [q], \sigma, [p] \rangle$ is an $\bar{\alpha}$-transition of $\mathcal{C}$, then for every $q' \in [q]$, there is $p' \in [p]$ such that $\langle q', \sigma, p' \rangle$ is an $\bar{\alpha}$-transition of $\mathcal{B}$, and thus the $\supseteq$-direction of the following proposition, suggesting that a safe run in $\mathcal{C}$ induces a safe run in $\mathcal{B}$, follows by a simple induction. The $\subseteq$-direction follows immediately from the definition of $\mathcal{C}$.

▶ **Proposition 17.** *For every $[p] \in Q_\mathcal{C}$ and every $s \in [p]$, it holds that $L_{safe}(\mathcal{B}^s) = L_{safe}(\mathcal{C}^{[p]})$.*

We extend Propositions 1 and 13 to the setting of $\mathcal{B}$ and $\mathcal{C}$:

▶ **Proposition 18.** *Consider states $s \in Q$ and $[p] \in Q_\mathcal{C}$, a letter $\sigma \in \Sigma$, and transitions $\langle s, \sigma, s' \rangle$ and $\langle [p], \sigma, [p'] \rangle$ of $\mathcal{B}$ and $\mathcal{C}$, respectively. If $s \sim p$, then $s' \sim p'$.*

Proof. As $\langle [p], \sigma, [p'] \rangle$ is a transition of $\mathcal{C}$, there are states $t \in [p]$ and $t' \in [p']$, such that $\langle t, \sigma, t' \rangle \in \Delta$. If $s \sim p$, then $s \sim t$. Since $\mathcal{B}$ is nice, in particular, semantically deterministic, and $\langle s, \sigma, s' \rangle \in \Delta$, we get by Proposition 1 that $s' \sim t'$. Thus, as $t' \sim p'$, we are done. ◀

▶ **Proposition 19.** *For every $[p] \in Q_\mathcal{C}$ and $s \in [p]$, we have that $\mathcal{C}^{[p]}$ is a GFG-tNCW equivalent to $\mathcal{B}^s$.*

[4] In fact, α-homogeneity is not required, but as the GFG-tNCW $\mathcal{B}_\mathcal{S}$ obtained in Section 4 is α-homogenous, which simplifies the proof, we are going to rely on it.

Proof. We first prove that $L(\mathcal{C}^{[p]}) \subseteq L(\mathcal{B}^s)$. Consider a word $w = \sigma_1\sigma_2\ldots \in L(\mathcal{C}^{[p]})$. Let $[p_0], [p_1], [p_2], \ldots$ be an accepting run of $\mathcal{C}^{[p]}$ on w. Then, there is $i \geq 0$ such that $[p_i], [p_{i+1}], \ldots$ is a safe run of $\mathcal{C}^{[p_i]}$ on the suffix $w[i+1, \infty]$. Let $s_0, s_1, \ldots s_i$ be a run of $\mathcal{B}^s$ on the prefix $w[1, i]$. Note that $s_0 = s$. Since $s_0 \in [p_0]$, we have that $s_0 \sim p_0$, and thus an iterative application of Proposition 18 implies that $s_i \sim p_i$. In addition, as $w[i+1, \infty]$ is in $L_{safe}(\mathcal{C}^{[p_i]})$, we get, by Proposition 17, that $w[i+1, \infty] \in L_{safe}(\mathcal{B}^{p_i})$. Since $L_{safe}(\mathcal{B}^{p_i}) \subseteq L(\mathcal{B}^{p_i})$ and $s_i \sim p_i$, we have that $w[i+1, \infty] \in L(\mathcal{B}^{s_i})$. Hence, $s_0, s_1, \ldots s_i$ can be extended to an accepting run of $\mathcal{B}^s$ on w.

Next, we prove that $L(\mathcal{B}^s) \subseteq L(\mathcal{C}^{[p]})$ and that $\mathcal{C}^{[p]}$ is a GFG-tNCW. We do this by defining a strategy $h : \Sigma^* \to Q_{\mathcal{C}}$ such that for all words $w \in L(\mathcal{B}^s)$, we have that $h(w)$ is an accepting run of $\mathcal{C}^{[p]}$ on w. We define h as follows. Recall that $\mathcal{B}$ is nice. So, in particular, $\mathcal{B}^s$ is GFG. Let g be a strategy witnessing $\mathcal{B}^s$'s GFGness. We define $h(u) = [g(u)]$, for every finite word $u \in \Sigma^*$. Consider a word $w \in L(\mathcal{B}^s)$, and consider the accepting run $g(w) = g(w[1,0]), g(w[1,1]), g(w[1,2]), \ldots$ of $\mathcal{B}^s$ on w. Note that by the definition of $\mathcal{C}$, we have that $h(w) = [g(w[1,0])], [g(w[1,1])], [g(w[1,2])], \ldots$ is an accepting run of $\mathcal{C}^{[p]}$ on w, and so we are done. ◀

▶ **Proposition 20.** *The GFG-tNCW $\mathcal{C}$ is nice, safe-centralized, and safe-minimal.*

The proof of the proposition is in the full version. The considerations are similar to those in the proof of Proposition 15. In particular, for safe minimality, note that for states q and s of $\mathcal{B}$, we have that $[q] \approx [s]$ iff $[q] \precsim [s]$ and $[s] \precsim [q]$. Thus, it is sufficient to prove that if $[q] \precsim [s]$ then $q \precsim s$. Thus, we can now conclude the following:

▶ **Theorem 21.** *Every nice, safe-centralized, and α-homogenous GFG-tNCW can be turned in polynomial time into an equivalent nice, safe-centralized, and safe-minimal GFG-tNCW.*

6 Discussion

We presented a polynomial minimization algorithm for GFG-tNCWs. In contrast, minimization of DCWs is NP-complete [33]. This raises a natural question, as to whether both relaxations of the problem, namely the consideration of GFG automata, rather than deterministic ones, and the consideration of transition-based acceptance, rather than state-based one, are crucial for efficiency. Our conjecture is that minimization of transition-based DCWs (and hence, also transition-based DBWs) can be solved in polynomial time. Thus, the relaxation to GFG is not needed. Our conjecture is based on the understanding that the quotient construction fails for automata on infinite words as it does not capture traversal of transitions. Moreover, the study of GFG automata so far shows that their behavior is similar to that of deterministic automata. In particular, it is not hard to see that the NP-hardness proof of Schewe for DBWs minimization applies also to GFG-NBWs. The use of transition-based acceptance is related to another open problem in the context of DBW minimization: is there a 2-approximation polynomial algorithm for it, that is one that generates a DBW that is at most twice as big as a minimal one. Note that a tight minimization for the transition-based case would imply a positive answer here. Note also that the vertex-cover problem, used in Schewe's reduction has a polynomial 2-approximation. As described in Section 1, there is recently growing use of automata with transition-based acceptance. Our work here is another evidence to their usefulness.

We find the study of minimization of GFG automata of interest also beyond being an intermediate result in the quest for efficient transition-based DBW minimization. Indeed, GFG automata are important in practice, as they are used in synthesis and control, and in

the case of the co-Büchi acceptance condition, they may be exponentially more succinct than their deterministic equivalences. Another open problem, which is interesting from both the theoretical and practical points of view, is minimization of GFG-tNBW. Note that unlike the deterministic case, GFG-tNBW and GFG-tNCW are not dual. Also, experience shows that algorithms for GFG-tNBW and GFG-tNCW are quite different [3, 17, 4, 2].

Finally, recall that there may be different minimal tDCWs for a given language of infinite words. Our results show that the picture for minimal GFG-tNCWs is cleaner: Consider a language $L \subseteq \Sigma^\omega$, and let $\mathcal{A}$ be a minimal GFG-tNCW for L obtained by safe-centralizing and safe-minimizing a nice GFG-tNCW for it. Consider a nice minimal GFG-tNCW $\mathcal{B}$ for L. Then, the injection $\eta : S(\mathcal{A}) \to S(\mathcal{B})$ from Lemma 7 is actually a *bijection*; that is, η is one-to-one and onto. Indeed, for every safe component $T \in S(\mathcal{A})$ it holds that $|T| = |\eta(T)|$. Moreover, as both $\mathcal{A}$ and $\mathcal{B}$ are nice, related safe components are *isomorphic*, thus there is an bijection $\kappa : Q_\mathcal{A} \to Q_\mathcal{B}$ such that for every $q \in Q_\mathcal{A}$, we have that $q \approx \kappa(q)$, and for every $\bar{\alpha}$-transition $\langle q, \sigma, s\rangle$ of $\mathcal{A}$, we have that $\langle \kappa(q), \sigma, \kappa(s)\rangle$ is an $\bar{\alpha}$-transition of $\mathcal{B}$. Thus, all nice minimal GFG-tNCWs for L have the same set of safe components, and they differ only in α-transitions among these safe components. An interesting research direction is a study of these safe components and in particular a characterization of L by a congruence-based relation on finite words that is induced by them.

References

1 A. Badr, V. Geffert, and I. Shipman. Hyper-minimizing minimized deterministic finite state automata. *ITA*, 43(1):69–94, 2009.

2 M. Bagnol and D. Kuperberg. Büchi Good-for-Games Automata Are Efficiently Recognizable. In *Proc. 38th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 122 of *LIPIcs*, pages 16:1–16:14. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2018.

3 U. Boker, D. Kuperberg, O. Kupferman, and M. Skrzypczak. Nondeterminism in the Presence of a Diverse or Unknown Future. In *Proc. 40th Int. Colloq. on Automata, Languages, and Programming*, volume 7966 of *Lecture Notes in Computer Science*, pages 89–100, 2013.

4 U. Boker, O. Kupferman, and M. Skrzypczak. How Deterministic are Good-For-Games Automata? In *Proc. 37th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 93 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:14, 2017.

5 J.R. Büchi. On a Decision Method in Restricted Second Order Arithmetic. In *Proc. Int. Congress on Logic, Method, and Philosophy of Science. 1960*, pages 1–12. Stanford University Press, 1962.

6 Th. Colcombet. The theory of stabilisation monoids and regular cost functions. In *Proc. 36th Int. Colloq. on Automata, Languages, and Programming*, volume 5556 of *Lecture Notes in Computer Science*, pages 139–150. Springer, 2009.

7 A. Duret-Lutz, A. Lewkowicz, A. Fauchille, Th. Michaud, E. Renault, and L. Xu. Spot 2.0 – a framework for LTL and ω-automata manipulation. In *14th Int. Symp. on Automated Technology for Verification and Analysis*, volume 9938 of *Lecture Notes in Computer Science*, pages 122–129. Springer, 2016.

8 J. Esparza, O. Kupferman, and M.Y. Vardi. Verification. In *Handbook AutoMathA*, pages 549–588. European Mathematical Society, 2018.

9 D. Giannakopoulou and F. Lerda. From States to Transitions: Improving Translation of LTL Formulae to Büchi Automata. In *Proc. 22nd International Conference on Formal Techniques for Networked and Distributed Systems*, volume 2529 of *Lecture Notes in Computer Science*, pages 308–326. Springer, 2002.

10 S. Gurumurthy, R. Bloem, and F. Somenzi. Fair simulation minimization. In *Proc. 14th Int. Conf. on Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pages 610–623. Springer, 2002.

11 S. Halamish and O. Kupferman. Minimizing Deterministic Lattice Automata. *ACM Transactions on Computational Logic*, 16(1):1–21, 2015.

12 T.A. Henzinger, O. Kupferman, and S. Rajamani. Fair simulation. *Information and Computation*, 173(1):64–81, 2002.

13 T.A. Henzinger and N. Piterman. Solving Games without Determinization. In *Proc. 15th Annual Conf. of the European Association for Computer Science Logic*, volume 4207 of *Lecture Notes in Computer Science*, pages 394–410. Springer, 2006.

14 J.E. Hopcroft. An $n \log n$ algorithm for minimizing the states in a finite automaton. In Z. Kohavi, editor, *The Theory of Machines and Computations*, pages 189–196. Academic Press, 1971.

15 A. Jez and A. Maletti. Hyper-Minimization for Deterministic Tree Automata. *Int. J. Found. Comput. Sci.*, 24(6):815–830, 2013.

16 T. Jiang and B. Ravikumar. Minimal NFA problems are hard. *SIAM Journal on Computing*, 22(6):1117–1141, 1993.

17 D. Kuperberg and M. Skrzypczak. On Determinisation of Good-for-Games Automata. In *Proc. 42nd Int. Colloq. on Automata, Languages, and Programming*, pages 299–310, 2015.

18 O. Kupferman and Y. Lustig. Lattice Automata. In *Proc. 8th Int. Conf. on Verification, Model Checking, and Abstract Interpretation*, volume 4349 of *Lecture Notes in Computer Science*, pages 199–213. Springer, 2007.

19 O. Kupferman, S. Safra, and M.Y. Vardi. Relating word and tree automata. *Ann. Pure Appl. Logic*, 138(1-3):126–146, 2006.

20 O. Kupferman and M.Y. Vardi. Safraless Decision Procedures. In *Proc. 46th IEEE Symp. on Foundations of Computer Science*, pages 531–540, 2005.

21 L.H. Landweber. Decision Problems for ω–Automata. *Mathematical Systems Theory*, 3:376–384, 1969.

22 W. Li, Sh. Kan, and Z. Huang. A Better Translation From LTL to Transition-Based Generalized Büchi Automata. *IEEE Access*, 5:27081–27090, 2017.

23 C. Löding. Efficient Minimization of Deterministic Weak Omega-Automata. *Information Processing Letters*, 79(3):105–109, 2001.

24 A. Malcher. Minimizing finite automata is computationally hard. *Theoretical Computer Science*, 327(3):375–390, 2004.

25 M. Mohri. Finite-State Transducers in Language and Speech Processing. *Computational Linguistics*, 23(2):269–311, 1997.

26 G. Morgenstern. Expressiveness results at the bottom of the ω-regular hierarchy. M.Sc. Thesis, The Hebrew University, 2003.

27 D.E. Muller, A. Saoudi, and P. E. Schupp. Weak Alternating Automata Give a Simple Explanation of Why Most Temporal and Dynamic Logics are Decidable in Exponential Time. In *Proc. 3rd IEEE Symp. on Logic in Computer Science*, pages 422–427, 1988.

28 J. Myhill. Finite automata and the representation of events. Technical Report WADD TR-57-624, pages 112–137, Wright Patterson AFB, Ohio, 1957.

29 A. Nerode. Linear Automaton Transformations. *Proceedings of the American Mathematical Society*, 9(4):541–544, 1958.

30 D. Niwinski and I. Walukiewicz. Relating hierarchies of word and tree automata. In *Proc. 15th Symp. on Theoretical Aspects of Computer Science*, volume 1373 of *Lecture Notes in Computer Science*. Springer, 1998.

31 M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:115–125, 1959.

32 S. Safra. On the Complexity of ω-Automata. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 319–327, 1988.

33 S. Schewe. Beyond Hyper-Minimisation—Minimising DBAs and DPAs is NP-Complete. In *Proc. 30th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 8 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 400–411, 2010.

34 S. Sickert, J. Esparza, S. Jaax, and J. Křetínský. Limit-Deterministic Büchi Automata for Linear Temporal Logic. In *Proc. 28th Int. Conf. on Computer Aided Verification*, volume 9780 of *Lecture Notes in Computer Science*, pages 312–332. Springer, 2016.

35 R.E. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal of Computing*, 1(2):146–160, 1972.

36 M.Y. Vardi and P. Wolper. Reasoning about Infinite Computations. *Information and Computation*, 115(1):1–37, 1994.

A Type System for Interactive JSON Schema Inference (Extended Abstract)

Mohamed-Amine Baazizi
Sorbonne Université, CNRS, LIP6 UMR 7606, Paris, France
baazizi@ia.lip6.fr

Dario Colazzo
Université Paris-Dauphine, PSL, LAMSADE, France
dario.colazzo@dauphine.fr

Giorgio Ghelli
Dipartimento di Informatica, Università di Pisa, Italy
ghelli@di.unipi.it

Carlo Sartiani
DIMIE, Università della Basilicata – Potenza, Italy
sartiani@gmail.com

Abstract

In this paper we present the first JSON type system that provides the possibility of inferring a schema by adopting different levels of precision/succinctness for different parts of the dataset, under user control. This feature gives the data analyst the possibility to have detailed schemas for parts of the data of greater interest, while more succinct schema is provided for other parts, and the decision can be changed as many times as needed, in order to explore the schema in a gradual fashion, moving the focus to different parts of the collection, without the need of reprocessing data and by only performing type rewriting operations on the most precise schema.

2012 ACM Subject Classification Theory of computation → Type theory; Information systems → Semi-structured data

Keywords and phrases JSON, type systems, interactive inference

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.101

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version A full version of the paper is available at [3], `https://hal.archives-ouvertes.fr/hal-02112560/file/icalp2019-full.pdf`.

1 Introduction

When a data analyst, or a programmer, accesses a JSON data collection for the first time, it is usually necessary to first have a look at its schema, as it is often the case that the data collection is badly documented and that it requires some visual data navigation in order to be fully understood. To this aim, some automated support can be quite useful, especially when the collection is massive and one is interested in a complete description of the structure.

The problem is that the structure of a data collection can be described at many different levels of abstraction: for example, when a semistructured collection of records is examined, one may either just list which fields are present in at least one element, or one may list every possible field combination. The choice of the abstraction level is extremely important, since an abstract description may hide too much information while, in other cases, a detailed description may be so big as to make the description unreadable. Unfortunately, there is

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 101; pp. 101:1–101:13

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

no objective and absolute definition of an "optimal" abstraction level, since it depends on the specific needs of the analyst, who may even need different abstraction levels in different phases of her/his work.

In [4] we proposed a first approach to this problem: a schema inference technique that is parametrized by an Equivalence Relation (ER); through this parameter, the user chooses one specific trade-off between succinctness and precision. This is much better than having no choice, but has important drawbacks. First of all, the choice of the value of the parameter is difficult. Secondly, and more importantly, one is typically very interested in some parts of the collection only, hence there is not a globally optimum trade-off: any intermediate choice between maximal detail and maximal compactness will typically be too compact for the interesting parts of the collection, but too detailed with respect to anywhere else. Hence, we built an interactive workbench, where the analyst can start from an abstract description of the collection, and then ask the system to selectively expand the structural description of some specific parts. Of course, the process can be iterated back and forth across the structural description of the data.

The analyst starts from an initial schema and interactively "expands" and "contracts" different parts of it. The main problem in the design of our workbench was that of understanding the exact meaning of these local "expansion" and "contraction" steps, as well as the lack of formal foundations for the optimization of these operations. In this paper we fill these gaps, by giving the formal foundations of interactive schema inference for massive JSON data. Our first contribution is a non-deterministic type system providing a formal characterisation of schemas featuring different precision levels for different parts of the typed dataset. We then define a deterministic version of this system, that is, a system where the user can choose a precision level through the choice of a *split criterion* parameter, that we will define. We then introduce an explicit representation of proof manipulations, in order to formalize the re-typing process, that is, the process of expanding/contracting a specific point of the schema. In principle, re-typing always involves to re-infer a schema from a portion of the data, which is not feasible in a big data setting. In our implementation, that we describe in [3], we exploit some properties of our *split criteria* in order to perform the re-typing without accessing data for a second time. Hence, we conclude our work by proving the soundness of this implementation technique, and studying the properties of our *split criteria* that allow this optimization.

2 Overview

Consider a massive collection of records, not perfectly homogeneous and sampled by the tiny collection below (left-hand side). The schema inference techniques described in [4] allows one to infer a type for each of the records, so as to obtain the collection of types below (right-hand side).

Data				*Types*			
$\{$	$a : \{j : 0, k : 0\},$	$b : \{bb : 0\}$	$\}$	$\{a : \{j : \texttt{Num}, k : \texttt{Num}\},$	$b : \{bb : \texttt{Num}\}$	$\}$
$\{$	$a : \{j : 0\},$	$c : \{cc : 0\}$	$\}$	$\{a : \{j : \texttt{Num}\},$	$c : \{cc : \texttt{Num}\}$	$\}$
$\{$	$a : \{y : 0, z : 0\}$	$c : \{cd : 0\}$	$\}$	$\{a : \{y : \texttt{Num}, z : \texttt{Num}\},$	$c : \{cd : \texttt{Num}\}$	$\}$
$\{$	$a : \{j : 0\},$	$b : 0$	$\}$	$\{a : \{j : \texttt{Num}\},$	$b : \texttt{Num}$	$\}$

The schema that is synthesized out of this collection of types depends on a parameter that is set by the analyst, which is an equivalence relation (ER) E: the inferred schema is obtained by merging any two types that are E-equivalent. For instance, we may use the

$\mathcal{K}$-equivalence, defined in [4] as the equivalence that equates all pairs of types of the same kind (record, array, or base type). In this way, all record types are merged, thus obtaining the following type, where a question mark indicates that the field is optional.

$$\{a : \{j : \texttt{Num}?, k : \texttt{Num}?, y : \texttt{Num}?, z : \texttt{Num}?\}, b : +_K(\{bb : \texttt{Num}\}, \texttt{Num})?, c : \{cc : \texttt{Num}?, cd : \texttt{Num}?\}?\}$$

Observe that all outer record types are merged, and, inside the record types, the types for the a and c keys are merged. The only two types that are kept distinct are those for the b key, since $\{bb : \texttt{Num}\}$ and $\texttt{Num}$ have two different kinds, hence b has a union type $+_K(\{bb : \texttt{Num}\}, \texttt{Num})$, where the $\mathcal{K}$ labels refers to the equivalence that guided the merging.

Before going on, we now rewrite this schema according to the syntax that we are using in this paper, which is more verbose but a bit more natural for our task. In this syntax, we have a union type in front of every record, array, or base type, so that the type above becomes:

$$\begin{array}{lll} +_K(& \{ & a : +_K(\{j : +_K(\texttt{Num})?, k : +_K(\texttt{Num})?, y : +_K(\texttt{Num})?, z : +_K(\texttt{Num})?\}), \\ & & b : +_K(\{bb : +_K(\texttt{Num})\}, \texttt{Num})?, \\ & & c : +_K(\{cc : +_K(\texttt{Num})?, cd : +_K(\texttt{Num})?\})?\}) \end{array}$$

The more types are merged, the smaller is the resulting schema, and the less precise it results. For example, the above schema is compact but is not very precise, since it does not specify how the different fields are mutually related: it only specifies that a and bb are mandatory, but, for the other fields, any combination is allowed. If the analyst wants more information about field correlation, she/he may move to the $\mathcal{L}$-equivalence, that specifies that two records are equivalent iff the respective sets of top-level field labels are the same, hence merging these equivalent types. In this way, we infer the following type (the $\mathcal{L}$-type):

$$\begin{array}{lll} +_L(& \{ & a : +_L(\quad \{j : +_L(\texttt{Num}), k : +_L(\texttt{Num})\}, \{j : +_L(\texttt{Num})\} \quad), \\ & & b : +_L(\quad \{bb : +_L(\texttt{Num})\}, \texttt{Num}) \\ & \}, & \\ & \{ & a : +_L(\quad \{j : +_L(\texttt{Num})\}, \{y : +_L(\texttt{Num}), z : +_L(\texttt{Num})\} \quad), \\ & & c : +_L(\quad \{cc : +_L(\texttt{Num})\}, \{cd : +_L(\texttt{Num})\} \quad) \\ & \}) & \end{array}$$

This type is much bigger, and it gives a lot of information about label correlation: it shows that b and c are mutually exclusive and one must always be present, and the same for cc and cd. It also shows that j and k are exclusive with y, z, that the presence of y, z inside a implies the presence of c, and so on.

The ability to start from a maximally compact type to get an overview of the schema, and then to move to a schema that is extremely detailed is already interesting but, in practice, the expanded schema may be overwhelming and may produce a lot of repetition. The analyst is often interested in just a subset of the input data and would like to be able to study the structure of that subset. For example, in this case the analyst may be mostly interested in the a field, and would like to be able to drill down on its structure only. Hence, she/he would like to start from the $\mathcal{K}$-type and to expand the a field only:

$$\begin{array}{lll} +_K(\{ & a : +_L(& \{j : +_L(\texttt{Num}), k : +_L(\texttt{Num})\}, \{j : +_L(\texttt{Num})\}, \{y : +_L(\texttt{Num}), z : +_L(\texttt{Num})\} \quad), \\ & b : +_K(& \{bb : +_K(\texttt{Num})\}, +_K(\texttt{Num}) \quad)?, \\ & c : +_K(& \{cc : +_K(\texttt{Num})?, cd : +_K(\texttt{Num})?\} \quad)? \ \} \quad) \end{array}$$

In this schema, we have full information about the type of the a field, while the outer structure and the c field are still represented in the compact style of the $\mathcal{K}$-types. Of course the analyst should be free now to also expand the c field, obtaining the following type.

$$+_K(\ \{\ a : +_L(\quad \{j : +_L(\texttt{Num}), k : +_L(\texttt{Num})\}, \{j : +_L(\texttt{Num})\}, \{y : +_L(\texttt{Num}), z : +_L(\texttt{Num})\}\quad),$$
$$\qquad b : +_K(\quad \{bb : +_K(\texttt{Num})\}, +_K(\texttt{Num})\quad)?,$$
$$\qquad c : +_L(\quad \{cc : +_L(\texttt{Num})\}, \{cd : +_L(\texttt{Num})\}\quad)?\ \}\quad)$$

Finally, we may collapse the a field, getting the following type.

$$+_K(\ \{\ a : +_K(\quad \{j : +_K(\texttt{Num})?, k : +_K(\texttt{Num})?, y : +_K(\texttt{Num})?, z : +_K(\texttt{Num})?\}\quad),$$
$$\qquad b : +_K(\quad \{bb : +_K(\texttt{Num})\}, +_K(\texttt{Num})\quad)?,$$
$$\qquad c : +_L(\quad \{cc : +_L(\texttt{Num})\}, \{cd : +_L(\texttt{Num})\}\quad)?\ \}\quad)$$

The formal definition of this operation of *local-equivalence-switch* is less obvious than it may appear. Essentially, the basic idea is that we apply different type-inference techniques to different parts of the data, but these parts are scattered in the dataset: in our example, the a fields are interleaved with the b and c fields. We would like to describe each step in the expand/collapse structure as a rewrite operation in the typing proof-tree, but this rewrite is non-local on the data, which is not trivial to formalize. Moreover, we would like that the relationship between a schema and the described data were fully described by the type, with its $\mathcal{K}$ and $\mathcal{L}$ annotations, and did not depend on the sequence of steps that has been used to arrive to that type. This property is very important for the analyst, who will usually not even remember the sequence of steps that she/he has used in order to arrive at the current schema, but still would like to have an interpretation of the meaning of that schema. The formalization of these properties is the theme of the paper.

3 Syntax and Semantics

We represent JSON values through the following grammar, that corresponds to actual JSON syntax, with the only exception that we do not put quotes around the keys (the labels) of the records. Following [6], we assume key uniqueness in records.

Syntax

$J ::=$	$B \mid R \mid A$		**JSON expressions**
$B ::=$	null $\mid$ true $\mid$ false $\mid n \mid s$	$n \in$ **Number**, $s \in$ **String**	**Basic values**
$R ::=$	$\{l_1 : J_1, \ldots, l_n : J_n\}$	$n \geq 0,\quad i \neq j \Rightarrow l_i \neq l_j$	**Records**
$A ::=$	$[J_1, \ldots, J_n]\quad n \geq 0$		**Arrays**

Basic values B include the null value, booleans, numbers n, and strings s. Records represent sets of fields, each field being a key-value pair (l, J), and arrays represent sequences of values. We will use J to range over JSON expressions and $\mathbb{J}$ to range over lists (or *collections*) of JSON expressions.

▶ **Notation 1.** *We use $L_1 \oplus L_2$ for list concatenation and $e.L$ to add e at the beginning of a list L. We use* collection *as a synonym for* list*, in situations where the order is there for technical reasons but is otherwise irrelevant.*

In the full paper [3] we define a semantic function $[\![\, J \,]\!]$ that maps each JSON term J to the corresponding mathematical entity, e.g. a number term into a number, a record into a set of pairs, an array into a list. That semantics is standard. As an example, the semantics of records is defined as $[\![\, \{l_1 : J_1, \ldots, l_n : J_n\} \,]\!] \triangleq \{\ (l_1, [\![\, J_1 \,]\!]), \ldots, (l_n, [\![\, J_n \,]\!])\ \}$

Our JSON types obey the following grammar. Every *union type* $\mathcal{T}$ is a union $+_{\mathcal{C}}(\mathcal{S}_1, \ldots, \mathcal{S}_n)$ of *structural types* $\mathcal{S}_1, \ldots, \mathcal{S}_n$, where the optional $\mathcal{C}$ subscript on non zero-ary union types can be ignored for the moment, and later will be used to indicate the "split criterion" that has been used in order to infer that part of the type. Every structural type

is either a base type $\mathcal{B}$, a record type $\mathcal{R}$ or an array type $\mathcal{A}$; record types and array types are defined in terms of union types. Each record field in a record type is labeled with a quantifier indicating whether the field is optional, noted as ?, or mandatory, noted as !. In our examples we will often omit the "!" symbol and only keep "?" for optional fields.

Syntax

$\mathcal{U}$	::=	$+(\mathcal{S}_1,\ldots,\mathcal{S}_n) \mid +_{\mathcal{C}}(\mathcal{S}_1,\ldots,\mathcal{S}_{n+1})$	$n \geq 0$	**Union Types**
$\mathcal{S}$	::=	$\mathcal{B} \mid \mathcal{R} \mid \mathcal{A}$		**Struct. types**
$\mathcal{B}$	::=	$\mathtt{Null} \mid \mathtt{Bool} \mid \mathtt{Num} \mid \mathtt{Str}$		**Basic types**
$\mathcal{R}$	::=	$\{l_1 : \mathcal{U}_1\mathtt{q}_1,\ldots,l_n : \mathcal{U}_n\mathtt{q}_n\}$	$n \geq 0\ , \mathtt{q}_i \in \{\ ?,!\ \}$	**Record types**
$\mathcal{A}$	::=	$[\mathcal{U}]$		**Array types**

The formal semantics of types is standard, and needs some explanation only in the records case. A record is a set of pairs, hence the empty record type { } denotes a singleton that only contains the empty record, that is, the empty set of pairs. A one-field record type is a set of singletons when the field is mandatory, and it also contains the empty record when the field is optional. Finally, a record that contains n fields is just a set that contains n pairs, hence a record that belongs to $\{l_1 : \mathcal{U}_1\mathtt{q}_1,\ldots,l_n : \mathcal{U}_n\mathtt{q}_n\}$ is just the union of one record for each field in the type. When a field $\mathcal{U}_i\mathtt{q}_i$ is optional, then the corresponding field in the value may be missing, which is captured by the presence of the empty set in the semantics of that field. In the semantics of $+(\mathcal{S}_1,\ldots,\mathcal{S}_n)$ we adopt the usual convention that $\cup_{i\in 1..0}\mathcal{S}_i$ is the empty set, hence $[\![+(\ )]\!] = \emptyset$. In this phase we just ignore the $\mathcal{C}$ annotation. The semantics of base types is standard, and we omit it in this extended abstract.

Semantics

$$
\begin{array}{lcl}
[\![\{\ \}]\!] & \triangleq & \{\ \emptyset\ \} \\
[\![\{l : \mathcal{U}!\}]\!] & \triangleq & \{\ \{\ (l, V)\ \} \mid V \in [\![\mathcal{U}]\!]\ \} \\
[\![\{l : \mathcal{U}?\}]\!] & \triangleq & [\![\{l : \mathcal{U}!\}]\!] \cup [\![\{\ \}]\!] \\
[\![\{l_1 : \mathcal{U}_1\mathtt{q}_1,\ldots,l_n : \mathcal{U}_n\mathtt{q}_n\}]\!] & \triangleq & \{\ R_1 \cup \ldots \cup R_n \mid R_i \in [\![\{l_i : \mathcal{U}_i\mathtt{q}_i\}]\!],\ i = 1,\ldots,n\ \}
\end{array}
$$

$$[\![[\mathcal{U}]]\!] \triangleq \{\ (V_1,\ldots,V_n) \mid n \geq 0,\ V_i \in [\![\mathcal{U}]\!]\ \}$$

$$[\![+_{[\mathcal{C}]}(\mathcal{S}_1,\ldots,\mathcal{S}_n)]\!] \triangleq \bigcup_{i\in 1..n} [\![\mathcal{S}_i]\!]$$

We partition the structural types into six kinds $\{\ \mathtt{Null}, \mathtt{Bool}, \mathtt{Num}, \mathtt{Str}, \{\ \}, [\ ]\ \}$ by using the following *kind*() function, that returns the *kind* of each structural type. $kind(K) = K$ for $K \in \{\ \mathtt{Null}, \mathtt{Bool}, \mathtt{Num}, \mathtt{Str}\ \}$ while $kind(\mathcal{A}) = [\ ]$ and $kind(\mathcal{R}) = \{\ \}$. We also define a *kind*() function that maps every JSON term to its kind – we omit the obvious definition. We say that a collection $\mathbb{J}$ of JSON terms is *kind-homogeneous* if it is not empty and all its elements have the same kind, in which case $kind(\mathbb{J})$ denotes that kind; $kind(\mathbb{J})$ is undefined when $\mathbb{J}$ is empty or when $\mathbb{J}$ is not kind-homogeneous.

4 Type System

4.1 The non-deterministic type system

In [4] we introduced a parametric deterministic type system, that is, a function that, given a JSON collection $\mathbb{J}$ and a parameter E, returns a type $\mathcal{U}$ for $\mathbb{J}$. The user could influence the type by choosing a specific equivalence relation for E: a finer equivalence results into a type that is more informative but bigger, while a coarser equivalence yields a type that is more compact but less informative. We focused our attention on two equivalence relations only, which we called $\mathcal{L}$ and $\mathcal{K}$, hence the analyst could choose, for each data collection, between two types, the $\mathcal{L}$-type, big and detailed, or the $\mathcal{K}$-type, smaller but less informative.

That approach is not flexible enough. The analyst may prefer to use a different size-precision trade-off for different parts of the data, depending on her/his interests. Hence, we need to formalize the notion that one term collection may have many different types, hence we need a type system that is much more flexible than the one in [4].

This inference system is based on two judgments that are mutually recursive: $\vdash \mathbb{J} :_u \mathcal{U}$ and $\vdash \mathbb{J} :_s \mathcal{S}$. Type inference rules are shown in Figure 1. The judgment $\vdash \mathbb{J} :_u \mathcal{U}$ specifies that the union type $\mathcal{U}$ describes the collection $\mathbb{J}$. Rule (EMPTY) specifies that the empty collection has the empty union type, while rule (+) splits $\mathbb{J}$, non-deterministically, into n kind-homogeneous subcollections $(\mathbb{J}_1, \ldots, \mathbb{J}_n)$. The *split criterion* $\mathcal{C}$ describes how $\mathbb{J}$ has been split. It is a partial function that describes how to split a collection of JSON terms. In this type system it is chosen non-deterministically, it may be different in any instance of the rule, can be optionally reported in the type, and it is only used to record how one specific collection has been split. In the deterministic variant of the system, presented in the next subsection, it will be chosen in a systematic way. The judgment $\vdash \mathbb{J} :_s \mathcal{S}$ assigns a structural type $\mathcal{S}$ to the kind-homogeneous collection $\mathbb{J}$, so that $kind(\mathbb{J}) = kind(\mathcal{S})$.

▶ **Definition 2** (*split*($\mathbb{J}$), split criterion $\mathcal{C}$). *For any non-empty collection of JSON terms, split*($\mathbb{J}$) *is the set of all partitionings of* $\mathbb{J}$ *into kind-homogeneous subcollections (abbreviated k-hom). A* split criterion $\mathcal{C}$ *is a partial function such that, for any non-empty collection of JSON terms* $\mathbb{J}$ *that belongs to its domain,* $\mathcal{C}(\mathbb{J}) \in split(\mathbb{J})$.

$$split(\mathbb{J}) = \{ (\mathbb{J}_1, \ldots, \mathbb{J}_n) \mid \mathbb{J}_1 \oplus \ldots \oplus \mathbb{J}_n = \mathbb{J}, \ \forall i, j.\ i \neq j \Rightarrow \mathbb{J}_i \cap \mathbb{J}_j = \emptyset, \forall i \in 1..n.\ \mathbb{J}_i \text{ is k-hom} \}$$

The three rules of Figure 1 for structural types can only be applied to kind-homogeneous collections. The notations used in the structural rules are introduced below.

$$\text{(EMPTY)}\ \frac{}{\vdash \emptyset :_u +()} \qquad \text{(+)}\ \frac{\mathcal{C} \text{ is a split criterion} \quad (\mathbb{J}_1, \ldots, \mathbb{J}_n) = \mathcal{C}(\mathbb{J}) \quad \forall i = 1, \ldots, n.\ \vdash \mathbb{J}_i :_s \mathcal{S}_i}{\vdash \mathbb{J} :_u +(\mathcal{S}_1, \ldots, \mathcal{S}_n) \text{ or } +_{\mathcal{C}}(\mathcal{S}_1, \ldots, \mathcal{S}_n)}$$

$$\text{(BASE)}\ \frac{kind(\mathbb{J}) = \mathcal{B}}{\vdash \mathbb{J} :_s \mathcal{B}} \qquad \text{(REC)}\ \frac{kind(\mathbb{J}) = \{\ \} \quad \text{Let } (a_1, \ldots, a_n) = keys(\mathbb{J}) \quad \forall i = 1, \ldots, n.\ \vdash \mathbb{J}/a_i :_u \mathcal{U}_i \quad q_i = Q(\frac{|\mathbb{J}/a_i|}{|\mathbb{J}|})}{\vdash \mathbb{J} :_s \{a_1 : \mathcal{U}_1 q_1, \ldots, a_n : \mathcal{U}_n q_n\}}$$

$$\text{(ARRAY)}\ \frac{kind(\mathbb{J}) = [\,] \quad \vdash \mathbb{J}/[*] :_u \mathcal{U}}{\vdash \mathbb{J} :_s [\mathcal{U}]}$$

Figure 1 Type inference rules.

Rule (+) is the key rule. Consider again the collection of Section 2.

$$\begin{array}{llllll} (& \{ & a : \{j : 0, k : 0\}, & b : \{bb : 0\} & \}, & \{ \quad a : \{j : 0\}, \quad c : \{cc : 0\} \quad \} \\ & \{ & a : \{y : 0, z : 0\} & c : \{cd : 0\} & \}, & \{ \quad a : \{j : 0\}, \quad b : 0 \qquad\quad \} \quad) \end{array}$$

If we split it into two different subsets, one for each different choice of the top-level labels, we will get a union type with two addends, similar to the $\mathcal{L}$-type presented in Section 2. However, if we consider the trivial split where all records are put together, we will have a type with only one top level addend, such as the three types whose root is labeled with $+_K$ in Section 2. We could also decide to split the collection in three subsets, in which case the resulting type will be the union of three types, one for each subset. A finer split will result into a union type that is bigger but where each addend is more precise. A coarser split will

yield a union type with less addends, and where each addend will be less precise. Every application of a union rule can use a different approach to splitting, and every application of a union rule corresponds to a different + in the resulting type. Hence, this rule gives us the flexibility that we need in order to model the process of interactive type modification.

For the rule (REC), we first introduce some notations.

▶ **Notation 3** ($J.a$, $\mathbb{J}/a, Q(x)$, $keys(\mathbb{J})$).

$$\begin{array}{llll} \{\ldots, a : J, \ldots\}.a = (\,J\,) & & Q(x) = \,! \;\; \textit{if } x = 1 \\ \{a_1 : J_1, \ldots, a_n : J_n\}.a = (\,) & \textit{if } \forall i \in 1..n.\ a_i \neq a & Q(x) = \,? \;\; \textit{if } 0 < x < 1 \\ \mathbb{J}/a = \bigoplus_{J \in \mathbb{J}} J.a & & keys(\mathbb{J}) = \bigcup_{J \in \mathbb{J}} (keys(J)) \end{array}$$

The rule (REC) specifies that, in order to analyze a kind-homogeneous collection of records, which may have different structures, we extract the collection of all keys that appear in any of them. For each key a_i, we collect the content of that key in all records and we infer a union type $\mathcal{U}_i$ for the collection $\mathbb{J}/a_i$. $\mathcal{U}_i$ will be the type for a_i in the result. Its qualifier will depend on the fraction of records where the a_i field is present, and will be ! if and only if the fraction is 1. In $Q(\,|\mathbb{J}/a_i| \,/\, |\mathbb{J}|\,)$, $|C|$ denotes the cardinality of a collection C.

The rule (ARRAY) is based on the $\mathbb{J}/[*]$ operator, that returns the content of all arrays in $\mathbb{J}$ as defined below; observe that both $(\,)/[*]$ and $(\,[\,], \ldots, [\,]\,)/[*]$ return the empty collection.

▶ **Notation 4** ($\mathbb{J}/[*]$). *For any collection of arrays, the operator $\mathbb{J}/[*]$ is defined as follows.*

$$(\,[J_1^1, \ldots, J_{n_1}^1], \ldots, [J_1^m, \ldots, J_{n_m}^m]\,)/[*] \triangleq (\,J_1^1, \ldots, J_{n_1}^1, \ldots, J_1^m, \ldots, J_{n_m}^m\,)$$

The use of the $\mathbb{J}/a$ and $\mathbb{J}/[*]$ operators in the typing rules is a technical contribution of this work and, in combination with the approach of typing a whole collection at the same time, is the fundamental tool that allows us to express the fact that the same criterion will be used for data that is not consecutive in the dataset but is related by the fact that it is reached through the same path.

This type system enjoys soundness, in the following sense.

▶ **Theorem 5** (Soundness). *For any JSON collection $\mathbb{J}$, union type $\mathcal{U}$, structural type $\mathcal{S}$:*

$$\vdash \mathbb{J} :_u \mathcal{U} \;\Rightarrow\; [\![\mathbb{J}]\!] \subseteq [\![\mathcal{U}]\!] \qquad \vdash \mathbb{J} :_s \mathcal{S} \;\Rightarrow\; [\![\mathbb{J}]\!] \subseteq [\![\mathcal{S}]\!]$$

▶ **Example 6.** Consider a collection $\mathbb{J} = (\,20, [1,3,5], [\,], [1, \text{true}], [2,4]\,)$. We want to find a union type $\mathcal{U}$ such that $\vdash \mathbb{J} :_u \mathcal{U}$. We start with the (+) rule, which divides $\mathbb{J}$ into kind-homogeneous subsets to be separately analyzed. For example, it may be divided into four homogeneous collections $(\,20\,), (\,[1,3,5], [2,4]\,), (\,[\,]\,), (\,[1, \text{true}]\,)$. The (BASE) rule assigns `Num` to $(\,20\,)$. The (ARRAY) rule reduces the problem $\vdash (\,[1,3,5], [2,4]\,) :_s \mathcal{S}_1$ to $\vdash (\,1,3,5,2,4\,) :_u \mathcal{U}_1$. If the split criterion keeps all the integers together, the (BASE) rule assigns `Num` to the integers, hence we have $\vdash (\,1,3,5,2,4\,) :_u +(\texttt{Num})$, hence $\vdash (\,[1,3,5], [2,4]\,) :_s [+(\texttt{Num})]$.

In this way, starting from a four-way split of the original collection, we prove the following judgment: $\vdash \mathbb{J} :_u +(\texttt{Num}, [+(\texttt{Num})], [+()], [+(\texttt{Num}, \texttt{Bool})])$.

By splitting $\mathbb{J}$ in different ways, we may prove the following judgments (among others), all of them correct, each exhibiting a different trade-off of size and precision. For example, the first type is the only one that fully describes all the possible different shapes of the arrays in the data. The second type indicates the existence of some "pure integers" array, while the third one indicates the presence of empty arrays. The fourth one is the less informative but is still sound, since every array in the collection meets that description.

i) $\vdash \mathbb{J} :_u +\,(\texttt{Num},\ [+(\texttt{Num})],\ [+()],\ [+(\texttt{Num}, \texttt{Bool})]\,)$
ii) $\vdash \mathbb{J} :_u +(\,\texttt{Num},\ [+(\texttt{Num})],\ [+(\texttt{Num}, \texttt{Bool})]\,)$
iii) $\vdash \mathbb{J} :_u +(\,\texttt{Num},\ [+()],\ [+(\texttt{Num}, \texttt{Bool})]\,)$
iv) $\vdash \mathbb{J} :_u +(\,\texttt{Num},\ [+(\texttt{Num}, \texttt{Bool})]\,)$

4.2 The deterministic type system

The non-deterministic type system is a general framework that will be used to prove that every judgment that is generated by the process of interactive typing is sound. We formalize now a deterministic subsystem, that models how the user guides the behaviour of the system.

While the non-deterministic type system takes a non-deterministic choice of a split criterion for each instance of the (+) rule, our type checker splits the $\mathbb{J}$ collection according to a criterion that is defined by the user, and this is modeled by the deterministic system.

Our split criteria will be based on equivalence relations defined on JSON terms, as follows.

▶ **Definition 7** ($\mathcal{C}(E)$)**.** *For a given equivalence relation E defined on JSON terms, the split criterion $\mathcal{C}(E)$ maps a collection $\mathbb{J}$ to a partition of $\mathbb{J}$ according to E.*

We can now define two important equivalence relations which will be used to define two split criteria. They are *kind equivalence* $\mathcal{K}(J_1, J_2)$ and *label equivalence* $\mathcal{L}(J_1, J_2)$, and correspond to the equivalences $\mathcal{K}(\mathcal{S}_1, \mathcal{S}_2)$ and $\mathcal{L}(\mathcal{S}_1, \mathcal{S}_2)$ defined in [4] for structural types. Kind equivalence is the coarsest equivalence that can be used to define a split criterion, since it keeps all kind-homogeneous terms together. Label equivalence is the same as kind equivalence for base values and for arrays but, when records are compared, it only groups those records that have exactly the same keys, hence it is much finer. The formal definition of the two equivalences is obvious, and reported in the full version [3].

The simplest way to "determinize" the non-deterministic type system would be to choose a split criterion to be adopted in every instance of the (+) rule. We need, however, some more freedom than this, hence we adopt the notion of *determinizer*, which is a function from integer sequences to split criteria (Definition 8), that we use to associate a split criterion to each instance of the (+) rule in a proof.

Formally, we define the deterministic type system by adding a $\mathcal{D}$ parameter (a determinizer) to the rules, and by specializing the (+) rule to the following (+D) rule, that specifies that $\mathcal{D}(\epsilon)$, where ϵ is the empty sequence, is used to split $\mathbb{J}$, while the determinizers $next(\mathcal{D}, i)$, as defined in Definition 8, will be used in the subproofs.

$$\frac{(\mathbb{J}_1, \ldots, \mathbb{J}_n) = \mathcal{D}(\epsilon)(\mathbb{J}) \qquad \vdash^{next(\mathcal{D},i)} \mathbb{J}_i :_s \mathcal{S}_i \quad i = 1, \ldots, n}{\vdash^{\mathcal{D}} \mathbb{J} :_u +_{\mathcal{D}(\epsilon)}(\mathcal{S}_1, \ldots, \mathcal{S}_n)} \text{(+D)}$$

▶ **Definition 8** (Determinizer, $next(\mathcal{D}, i)$)**.** *A* determinizer $\mathcal{D}$ *is a function that maps any sequence, possibly empty, of positive integers $\omega = (i_1, \ldots, i_n)$ to a split criterion $\mathcal{D}(\omega)$. For any determinizer $\mathcal{D}$, $next(\mathcal{D}, i)$ is the determinizer that maps ω to $\mathcal{D}(i.\omega)$.*

A determinizer is hence a function that determines the split criterion used at each node of the type proof. By construction, every proof in the deterministic type system is also a proof in the non-deterministic one. In the full paper we show that the converse also holds: for every proof in the non-deterministic system a determinizer exists that generates that proof.

A determinizer may map any sequence to any split criterion, but, in practice, we will focus on very simple split criteria and very simple determinizers. Our system currently supports only three determinizers, L, K, LK, all of them employing equivalence-based criteria and a constant or almost-constant function.

▶ **Definition 9** (L, K, LK)**.** *L is the constant determinizer that maps every sequence ω to the split criterion $\mathcal{C}(\mathcal{L})$. K maps every sequence to $\mathcal{C}(\mathcal{K})$. LK maps the empty sequence to $\mathcal{C}(\mathcal{L})$ and any other sequence to $\mathcal{C}(\mathcal{K})$.*

Hereafter, we use the terms $\mathcal{L}$-type and $\mathcal{K}$-type, for a collection $\mathbb{J}$, to refer to the type that is inferred using, respectively, the L or the K determinizer. By Definition 9, L and K use, respectively, $\mathcal{C}(\mathcal{L})$ and $\mathcal{C}(\mathcal{K})$ as split criteria, everywhere in the proof. LK uses $\mathcal{C}(\mathcal{L})$ at the top level but $\mathcal{C}(\mathcal{K})$ everywhere else, that is, it expands like the $\mathcal{L}$-type at the top level, but computes a $\mathcal{K}$-type everywhere else.

4.3 The proof validity system

The aim of this paper is to provide a formal specification of a system that dynamically recomputes a type for a subset of the input data. We believe that the most direct approach is to describe it as a system that manipulates typing proofs and, to this aim, we need a notation to explicitly describe proofs and their manipulations. To this aim, we extend our judgments with a proof term that describes the proof itself, so that, rather than judgments $\vdash \mathbb{J} : \mathcal{T}$, that specify that the collection $\mathbb{J}$ has type $\mathcal{T}$, we will have ternary judgments with the form $\pi \vdash \mathbb{J} : \mathcal{T}$, that specify that π is a proof tree that proves that $\mathbb{J}$ has type $\mathcal{T}$.

More precisely, we first define a language of proof terms as follows.

$$\begin{array}{lcll} \pi^u & ::= & \langle +(\pi^s_1, \ldots, \pi^s_n)\rangle & n \geq 0 \\ \pi^s & ::= & \langle \mathbb{J}, \mathcal{B}\rangle \mid \langle \mathbb{J}, \{l_1 : \pi^u_1 \mathtt{q}_1, \ldots, l_n : \pi^u_n \mathtt{q}_n\}\rangle \mid \langle \mathbb{J}, [\pi^u]\rangle & \end{array}$$

Then, we define an inference system, based on two judgments that are mutually recursive $\pi^u \vdash \mathbb{J} :_u \mathcal{U}$ and $\pi^s \vdash \mathbb{J} :_s \mathcal{S}$ that specify when a proof term π is a proof (and not just a proof term), and, in that case, the fact that π proves that $\mathbb{J}$ has type $\mathcal{U}$. In the sequel we will use the metavariable π to range over both structural type proofs π^s and union type proofs π^u. The type inference rules are shown in Figure 2.

$$\text{(EMPTY)}\ \frac{}{\langle +(\,)\rangle \vdash \emptyset :_u +(\,)} \qquad \text{(+)}\ \frac{\mathcal{C} \text{ is a split criterion} \quad (\mathbb{J}_1, \ldots, \mathbb{J}_n) = \mathcal{C}(\mathbb{J}) \quad \pi_i \vdash \mathbb{J}_i :_s \mathcal{S}_i \quad i = 1, \ldots, n}{\langle +(\pi_1, \ldots, \pi_n)\rangle \vdash \mathbb{J} :_u +_{[\mathcal{C}]}(\mathcal{S}_1, \ldots, \mathcal{S}_n)} \qquad \text{(BASE)}\ \frac{kind(\mathbb{J}) = \mathcal{B}}{\langle \mathbb{J}, \mathcal{B}\rangle \vdash \mathbb{J} :_s \mathcal{B}}$$

$$\text{(REC)}\ \frac{kind(\mathbb{J}) = \{\,\} \quad \text{Let } (a_1, \ldots, a_n) = keys(\mathbb{J}) \quad \forall i = 1, \ldots, n.\ \pi_i \vdash \mathbb{J}/a_i :_u \mathcal{U}_i \quad \forall i = 1, \ldots, n.\ q_i = Q(\frac{|\mathbb{J}/a_i|}{|\mathbb{J}|})}{\langle \mathbb{J}, \{a_1 : \pi_1 q_1, \ldots, a_n : \pi_n q_n\}\rangle \vdash \mathbb{J} :_s \{a_1 : \mathcal{U}_1 q_1, \ldots, a_n : \mathcal{U}_n q_n\}} \qquad \text{(ARRAY)}\ \frac{kind(\mathbb{J}) = [\,] \quad \pi \vdash \mathbb{J}/[*] :_u \mathcal{U}}{\langle \mathbb{J}, [\pi]\rangle \vdash \mathbb{J} :_s [\mathcal{U}]}$$

Figure 2 Proof validity rules.

Given a proof π, in the full paper we define $\mathtt{JJ}(\pi)$ and $\mathtt{U}(\pi)$ as the unique collection $\mathtt{JJ}(\pi)$ and type $\mathtt{U}(\pi)$ such that $\pi \vdash \mathtt{JJ}(\pi) : \mathtt{U}(\pi)$. We then define a notion of path inside a proof or a type, where the steps (i), $[*]$ and a traverse, respectively, the union, array, and record constructors:

$$p \quad ::= \quad \epsilon \mid (i)/p \mid [*]/p \mid a/p$$

We then define a notion of subproof along a path $\pi \downarrow p$, and a notion of substitution $\pi[p \leftarrow \pi_1]$, that is only well defined when $\mathtt{JJ}(\pi_1) = \mathtt{JJ}(\pi \downarrow p)$, that is, a substitution will change the type but not the term. In the full paper we prove the fundamental property of this operation: if π and π_1 are valid proofs, then $\pi[p \leftarrow \pi_1]$ is a valid proof as well, where *valid* is defined by the rules of Figure 2. This property is the foundation of the formalization that we are going to present.

5 The Local-Retype Operation

We are now ready to define the local-retype operation. In our tool, the analyst has a collection $\mathbb{J}$ and a type $\mathcal{T}$ for $\mathbb{J}$, she/he clicks on a union type $\mathcal{U}$ inside $\mathcal{T}$ and selects the new determinizer $\mathcal{D}$ to use in order to re-type the subforest of $\mathbb{J}$ that corresponds to the type $\mathcal{U}$.

In general, this subforest does not correspond to a subsequence of the JSON collection: when the analyst selects the type of the *authors* field of a record type, the objects that correspond to the *authors* key are scattered through all those records that actually contain an *authors* field. However, if we consider a proof tree π such that $\pi \vdash \mathbb{J} : \mathcal{T}$, then these objects are all collected in the only premise of the (REC) rule that infers a type for $\mathbb{J}/authors$. Hence, we may formalize this action as follows: the analyst chooses a path p inside the type $\mathcal{T}$ and a new determinizer $\mathcal{D}$, and the system computes a new proof π' for the collection that corresponds to $\pi \downarrow p$, substitutes π' to $\pi \downarrow p$, and visualizes the type that corresponds to the new proof. The analysts operate on types, but the system is manipulating the corresponding proofs. We formalize this through the *retype* operation, as follows.

▶ **Definition 10** (*retype*$(\pi, p, \mathcal{D})$)**.** *For any proof π such that $\pi \vdash \mathbb{J} : \mathcal{T}$, for any p such that $\mathcal{T} \downarrow p$ is defined, for any determinizer $\mathcal{D}$, retype$(\pi, p, \mathcal{D})$ is defined as follows , where $\pi_{\mathcal{D}}$ is uniquely determined by $\mathcal{D}$, π and p :*

$$(\exists \mathcal{T}'.\ \pi_{\mathcal{D}} \vdash^{\mathcal{D}} \mathit{JJ}(\pi \downarrow p) : \mathcal{T}') \ \Rightarrow\ \mathit{retype}(\pi, p, \mathcal{D}) \triangleq \pi[p \leftarrow \pi_{\mathcal{D}}]$$

In other terms, the analyst indicates a path p, the system retrieves the corresponding collection $\mathtt{JJ}(\pi \downarrow p)$ inside the current proof π, computes a $\mathcal{D}$-proof $\pi_{\mathcal{D}}$ for that collection, and substitutes $\pi \downarrow p$ with $\pi^{\mathcal{D}}$. From the analyst's viewpoint, the type at $\mathcal{T} \downarrow p$ has been substituted with a $\mathcal{D}$-type for the corresponding collection.

In the full paper, we extend this definition to a *retype*$^*(\pi, \sigma)$ operation that executes a sequence $\sigma = (\,(p_1, \mathcal{D}_1), \ldots, (p_n, \mathcal{D}_n)\,)$ of retyping steps, and we prove that, for any such sequence, the resulting type is always a correct description of the input collection and a tight description, where every path in the type actually corresponds to some piece of data. Moreover, we prove that the split criteria that are present in the type actually correspond to how the collections have been split in the proof. In other terms, we prove that the specific sequence of steps is irrelevant, hence the final type that is displayed to the analyst only depends on the terms in the collection. For space reason, we only present here the most important results, but leave the discussion and the proofs to the full paper.

▶ **Theorem 11** (Type soundness of *retype*$^*()$)**.** *For any proof π such that $\pi \vdash \mathit{JJ}(\pi) : \mathtt{U}(\pi)$, for any sequence σ of retyping steps, the proof retype$^*(\pi, \sigma)$ is still a proof for $\mathit{JJ}(\pi)$:*

$$\mathit{retype}^*(\pi, \sigma) = \pi_n \ \Rightarrow\ \pi_n \vdash \mathit{JJ}(\pi) : \mathtt{U}(\pi_n)$$

▶ **Corollary 12.** *For any π such that $\pi \vdash \mathit{JJ}(\pi) : \mathtt{U}(\pi)$, for any sequence σ of retyping steps such that retype$^*(\pi, \sigma) = \pi_r$, for any path p:*

1. $[\![\, \mathit{JJ}(\pi_r \downarrow p) \,]\!] \subseteq [\![\, \mathtt{U}(\pi_r \downarrow p) \,]\!]$ — *Soundness of retype$^*()$*
2. *if $\pi \downarrow p$ is not $+()$:* $[\![\, \mathit{JJ}(\pi_r \downarrow p) \,]\!] \neq \emptyset$ — *Tightness of retype$^*()$*
3. *if $\pi_r \downarrow p = +_{\mathcal{C}}(\pi_1, \ldots, \pi_m)$:* $(\,\mathit{JJ}(\pi_1), \ldots, \mathit{JJ}(\pi_m)\,) = \mathcal{C}(\mathit{JJ}(\pi_r \downarrow p))$ — *Soundness of the split criterion*

6 Implementation

According to our definition, any time the analyst expands or contracts a node in the type, all data associated to that node are type-checked again. Since we are dealing with massive data collections, and we aim for a very fast response, this approach is not acceptable.

In our implementation, we do a different thing. We currently support the three determinizers K, L, and LK of Definition 9, and the L-type contains enough information to rebuild the K-type and the LK-type. Hence, we first compute and store the L-type $\mathcal{U}_L$ of the dataset $\mathbb{J}$, and we compute the K-type $\mathcal{U}_K$, which is the first one to be visualized, starting from $\mathcal{U}_L$. Then, rather than keeping track of a proof $\pi \vdash \mathbb{J} : \mathcal{U}_K$, we store a type-level abstract proof $\pi_{LK} \vdash \mathcal{U}_L : \mathcal{U}_K$, formalized in a type-to-type inference system specified below. When we need to recompute the type of a subcollection $\texttt{JJ}(\pi \downarrow p)$, rather than type-checking the subcollection, which may be extremely big, we compute its type starting from the collection of types $\texttt{U}(\pi_{LK} \downarrow p)$, which is a subforest of $\mathcal{U}_L$, and is an abstract description of $\texttt{JJ}(\pi \downarrow p)$.

In the full paper, in this section we present this approach, which is crucial for the applicability of our tool, and prove its correctness. We recap here the contents of the section.

We want to specify a set of conditions that make it possible to compute an abstract type from a detailed type without accessing the data. We focus our attention to a specific class of determinizers, those whose split condition is defined in terms of an equivalence, and, for the detailed type, we also ask that the split condition is constant, as it happens for the L determinizer, which constantly uses the $\mathcal{C}(\mathcal{L})$ split criterion. We then define a judgment $\vdash^{\mathcal{D}} \mathbb{J} :: \mathbb{S}$ that specifies that a collection of terms $\mathbb{J}$ is "described" by a collection of structural types $\mathbb{S}$. We can now define a notion of *refinement*, that specifies a condition that is sufficient in order to compute an abstract type starting from the detailed type.

The notion is based on the existence of a *Type-level split criterion for $\mathcal{D}$ and $\mathcal{C}$* $tsplit^{\mathcal{C}}_{\mathcal{D}}(\mathbb{S})$, that is, a function that splits the collection $\mathbb{S}$ describing $\mathbb{J}$ in the "same way" as $\mathcal{C}$ would have split $\mathbb{J}$. If, for a determinizer $\mathcal{E}$, for any ω, we have a type-level split criterion $tsplit^{\mathcal{E}(\omega)}_{\mathcal{D}}(\mathbb{S})$, then $\mathcal{D}$ *refines* $\mathcal{E}$, and we prove that this is sufficient to compute the $\mathcal{E}$-type from the $\mathcal{D}$-type.

To this aim, we present a set of rules to compute an $\mathcal{E}$-type from a $\mathcal{D}$-type if $\mathcal{D}$ refines $\mathcal{E}$, and the corresponding proof of correctness. This set of rules, reported in Figure 3, defines a *type-to-type* system, where the type collections at the left hand side substitute the terms that they represent, and the operators $\mathbb{S}/a$, $qual(\mathbb{S}, a)$, $keys(\mathbb{S})$, $\mathbb{S}/[*]$ represent the type-level lifting of the corresponding term operators.

$$\frac{}{\vdash^{\mathcal{E}}_{\mathcal{D}} +_C(\,) :_u +_{\mathcal{E}(\epsilon)}(\,)} \text{(EMPTY)} \qquad \frac{kind(\mathbb{S}) = \mathcal{B}}{\vdash^{\mathcal{E}}_{\mathcal{D}} \mathbb{S} :_s \mathcal{B}} \text{(BASE)} \qquad \frac{kind(\mathbb{S}) = [\,] \quad \vdash^{\mathcal{E}}_{\mathcal{D}} \mathbb{S}/[*] :_u \mathcal{U}}{\vdash^{\mathcal{E}}_{\mathcal{D}} \mathbb{S} :_s [\mathcal{U}]} \text{(ARRAY)}$$

$$\frac{(\mathbb{S}^1, \ldots, \mathbb{S}^n) = tsplit^{\mathcal{E}(\epsilon)}_{\mathcal{D}}(\mathbb{S}) \quad \vdash^{next(\mathcal{E},i)}_{\mathcal{D}} \mathbb{S}^i :_s \mathcal{S}'_i \quad i = 1, \ldots, n}{\vdash^{\mathcal{E}}_{\mathcal{D}} \mathbb{S} :_u +_{\mathcal{E}(\epsilon)}(\mathcal{S}'_1, \ldots, \mathcal{S}'_n)} \text{(+D)}$$

$$\frac{kind(\mathbb{S}) = \{\,\} \quad \text{Let } (a_1, \ldots, a_n) = keys(\mathbb{S}) \quad \forall i = 1, \ldots, n.\ \vdash^{\mathcal{E}}_{\mathcal{D}} \mathbb{S}/a_i :_u \mathcal{U}_i \quad q_i = qual(\mathbb{S}, a_i)}{\vdash^{\mathcal{E}}_{\mathcal{D}} \mathbb{S} :_s \{a_1 : \mathcal{U}_1 q_1, \ldots, a_n : \mathcal{U}_n q_n\}} \text{(REC)}$$

Figure 3 The t2t system: rules to infer an $\mathcal{E}$-type from a $\mathcal{D}$-type.

We then prove that the type-to-type system can be used to correctly compute the $\mathcal{E}$-type of any collection starting from its $\mathcal{D}$-type.

▶ **Property 13** (Soundness and completeness of $\vdash^{\mathcal{E}}_{\mathcal{D}} \mathcal{U}_D :_u \mathcal{U}_E$)**.** *For any $\mathbb{J}$, $\mathbb{S}$, $\mathcal{D}$, $\mathcal{E}$, $\mathbb{J}_c$, $\mathcal{U}_E$, $\mathcal{S}_E$, where $\mathbb{J}_c$ is kind-homogeneous, if $\mathcal{D}$ refines $\mathcal{E}$, then:*

$$\vdash^{\mathcal{D}} \mathbb{J} :: \mathbb{S} \quad \Rightarrow \quad (\ \vdash^{\mathcal{E}}_{\mathcal{D}} \mathbb{S} :_u \mathcal{U}_E \quad \Leftrightarrow \quad \vdash^{\mathcal{E}} \mathbb{J} :_u \mathcal{U}_E\ )$$
$$\vdash^{\mathcal{D}} \mathbb{J}_c :: \mathbb{S} \quad \Rightarrow \quad (\ \vdash^{\mathcal{E}}_{\mathcal{D}} \mathbb{S} :_s \mathcal{S}_E \quad \Leftrightarrow \quad \vdash^{\mathcal{E}} \mathbb{J}_c :_s \mathcal{S}_E\ )$$

7 Related Work

While the problem of inferring structural information from JSON collections has recently gained a momentum [8, 1, 2, 13, 9, 14], we are not aware of any approach that is interactive. In the XML realm, among the plethora of schema inference approaches [5, 10, 7, 11, 15, 12], only the one presented [15] is interactive since it relies on user intervention for recognizing regular expressions that are similar enough to be merged and for guiding the system to derive sophisticated schema expressing inheritance and restrictions, two constructs that are difficult to infer in a purely automatic fashion. While this approach bears some resemblance with our work in that both give the user some form of freedom to decide when two types are similar, the interaction presented in [15] is only meant to guide the system during the schema inference and does not allow for exploring different parts of an already existing schema with different precision lenses. Another notable difference with our approach is the lack of a mechanism for reasoning about type "similarity" like an equivalence class.

8 Conclusions

When semistructured JSON data are retrieved from the web, the presence of an automatic tool that can infer a schema is very useful, but the same data can be described with schemas with different size-precision trade offs, and no trade-off is optimal in general. Moreover, it is often the case that one is mostly interested in a specific part of the data, hence we designed and implemented a system where the analyst can interactively decide which parts of the data should be described with a greater or lower level of detail.

In this paper we have studied the formal foundations of this approach. The main novelty of this formal study is the use of specific precision criteria on specific parts of the data, that are identified by acting on a node of the syntax tree of a type. This aspect has been formalized through the use of some technical tools such as the use of selectors such as $\mathbb{J}/a$ and $\mathbb{J}/[*]$ inside the type proofs, and the notion of proof terms. With these tools we have proved the soundness of the local retype operation that we implemented.

The basic ideas that we presented here admit many extensions. First of all, while we designed the foundations of an interactive tool for an elementary type system, all the extensions of that elementary type system that we briefly presented in [4] – such as type constraints, enumeration types, variant types, tuple types, keyset equivalence – can be easily added. In the same way, the approach that we defined here can be easily applied to the counting type system that we defined in [2], hence combining the interactivity that we presented here with the quantitative information that was presented in that paper. We are currently exploring these possibilities.

Finally, while in [4] and [2] we studied the use of equivalence relations in order to split collection, here we allow any criterion to be used. This opens the way to some new possibilities, such as a top-n split, where one separates and groups the n shapes that are most important and collapses the others, or even forms of value-based grouping, where different records are grouped in a way that depends on the values of the fields and not just on the types.

References

1 Mohamed Amine Baazizi, Houssem Ben Lahmar, Dario Colazzo, Giorgio Ghelli, and Carlo Sartiani. Schema Inference for Massive JSON Datasets. In *EDBT '17*, 2017.

2 Mohamed-Amine Baazizi, Dario Colazzo, Giorgio Ghelli, and Carlo Sartiani. Counting Types for Massive JSON Datasets. In *DBPL '17*, 2017.

3 Mohamed-Amine Baazizi, Dario Colazzo, Giorgio Ghelli, and Carlo Sartiani. A Type System for Interactive JSON Schema Inference, April 2019. URL: `https://hal.archives-ouvertes.fr/hal-02112560`.

4 Mohamed-Amine Baazizi, Dario Colazzo, Giorgio Ghelli, and Carlo Sartiani. Parametric schema inference for massive JSON datasets. *The VLDB Journal*, pages 1–25, 2019.

5 Geert Jan Bex, Frank Neven, Thomas Schwentick, and Karl Tuyls. Inference of Concise DTDs from XML Data. In *VLDB '06*, pages 115–126, 2006. URL: `http://dl.acm.org/citation.cfm?id=1164139`.

6 Tim Bray. The JavaScript Object Notation (JSON) Data Interchange Format. Technical report, Internet Engineering Task Force (IETF), December 20017. Standards Track.

7 Radu Ciucanu and Slawek Staworko. Learning Schemas for Unordered XML. In *Proceedings of the 14th International Symposium on Database Programming Languages (DBPL 2013), August 30, 2013, Riva del Garda, Trento, Italy.*, 2013. `arXiv:1307.6348`.

8 Dario Colazzo, Giorgio Ghelli, and Carlo Sartiani. Typing Massive JSON Datasets. In *XLDI '12, Affiliated with ICFP*, 2012.

9 Michael DiScala and Daniel J. Abadi. Automatic Generation of Normalized Relational Schemas from Nested Key-Value Data. In Fatma Özcan, Georgia Koutrika, and Sam Madden, editors, *SIGMOD '16*, pages 295–310. ACM, 2016. `doi:10.1145/2882903.2882924`.

10 Dominik D. Freydenberger and Timo Kötzing. Fast Learning of Restricted Regular Expressions and DTDs. *Theory Comput. Syst.*, 57(4):1114–1158, 2015.

11 Markus Lohrey, Sebastian Maneth, and Carl Philipp Reh. Compression of Unordered XML Trees. In *ICDT'07*, pages 18:1–18:17, 2017. `doi:10.4230/LIPIcs.ICDT.2017.18`.

12 Irena Mlỳnková and Martin Nečaskỳ. Heuristic methods for inference of XML schemas: Lessons learned and open issues. *Informatica*, 24(4):577–602, 2013.

13 Felipe Pezoa, Juan L. Reutter, Fernando Suarez, Martín Ugarte, and Domagoj Vrgoč. Foundations of JSON Schema. In *WWW '16*, pages 263–273, 2016. `doi:10.1145/2872427.2883029`.

14 Stefanie Scherzinger, Eduardo Cunha de Almeida, Thomas Cerqueus, Leandro Batista de Almeida, and Pedro Holanda. Finding and Fixing Type Mismatches in the Evolution of Object-NoSQL Mappings. In *Proceedings of the Workshops of the EDBT/ICDT 2016*, 2016. URL: `http://ceur-ws.org/Vol-1558/paper10.pdf`.

15 Julie Vyhnanovska and Irena Mlynkova. Interactive Inference of XML Schemas. In *Proceedings of the Fourth IEEE International Conference on Research Challenges in Information Science, RCIS 2010, Nice, France, May 19-21, 2010*, pages 191–202, 2010.

On the Complexity of Value Iteration

Nikhil Balaji
University of Oxford, UK
nikhil.balaji@cs.ox.ac.uk

Stefan Kiefer
University of Oxford, UK
stekie@cs.ox.ac.uk

Petr Novotný
Masaryk University, Brno, Czech Republic
petr.novotny@fi.muni.cz

Guillermo A. Pérez
University of Antwerp, Belgium
guillermoalberto.perez@uantwerpen.be

Mahsa Shirmohammadi
CNRS, Paris, France
IRIF, Paris, France
mahsa.shirmohammadi@irif.fr

Abstract

Value iteration is a fundamental algorithm for solving Markov Decision Processes (MDPs). It computes the maximal n-step payoff by iterating n times a recurrence equation which is naturally associated to the MDP. At the same time, value iteration provides a policy for the MDP that is optimal on a given finite horizon n. In this paper, we settle the computational complexity of value iteration. We show that, given a horizon n in binary and an MDP, computing an optimal policy is **EXPTIME**-complete, thus resolving an open problem that goes back to the seminal 1987 paper on the complexity of MDPs by Papadimitriou and Tsitsiklis. To obtain this main result, we develop several stepping stones that yield results of an independent interest. For instance, we show that it is **EXPTIME**-complete to compute the n-fold iteration (with n in binary) of a function given by a straight-line program over the integers with max and + as operators. We also provide new complexity results for the bounded halting problem in linear-update counter machines.

2012 ACM Subject Classification Theory of computation → Probabilistic computation; Theory of computation → Logic and verification; Theory of computation → Markov decision processes

Keywords and phrases Markov decision processes, Value iteration, Formal verification

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.102

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version A full version of the paper is available at `https://arxiv.org/abs/1807.04920`.

Funding *Nikhil Balaji*: Part of this work was done was when the author was a Postdoctoral fellow at Ulm university supported by DFG grant TH 472/4.
Stefan Kiefer: is supported by a Royal Society University Research Fellowship.
Petr Novotný: is supported by the Czech Science Foundation, grant no. GJ19-15134Y "Verification and Analysis of Probabilistic Programs."
Mahsa Shirmohammadi: is supported by the "AAPS" PEPS JCJC grant.

Acknowledgements We thank James Worrell for helpful comments on early version of this work.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 102; pp. 102:1–102:15

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

1 Introduction

Markov decision processes (MDP) are a fundamental formalism of decision making under probabilistic uncertainty [29, 9]. As such, they play a prominent role in numerous domains, including artificial intelligence and machine learning [34, 33], control theory [10, 1], operations research and finance [11, 31], as well as formal verification [12, 5], to name a few. Informally, an MDP represents a system which is, at every time step, in one of the *states* from a finite set S. The system evolves in steps: in each step, we can perform an *action* (or *decision*) from a finite set A. When using an action $a \in A$ in state $s \in S$, we collect an immediate *reward* $R(s, a)$ and then transition stochastically to a successor state according to a rational-valued distribution $P(s, a)$, which is given as a part of the MDP. This interaction with an MDP proceeds over either a *finite* or *infinite* horizon. In the finite-horizon case, we are given a bound $H \in \mathbb{N}$ (a *horizon*) such that the interaction stops after H steps; in the infinite horizon case the process goes on forever. To *solve* an MDP means to find an optimal *policy*; that is, a blueprint for selecting actions that maximizes the expected reward accumulated over a finite or infinite horizon. The accumulated rewards are typically *discounted* by some factor $0 < \gamma \leq 1$; for infinite horizon, we need $\gamma < 1$ to ensure that the infinite sum is well defined.

Value iteration. Given the importance of MDPs, it is hardly surprising that they have attracted significant interest in the theory community. Past research on MDPs included the study of complexity issues [27] as well as the design and analysis of algorithms for solving MDPs [22, 24, 38, 39]. In this paper, we provide a fresh look on one of the most familiar algorithms for MDPs: *value iteration (VI)*. Introduced by Bellman in the 1950s [6], VI makes use of the optimality principle: the maximal n-step reward achievable from a state s, which we denote by $\vec{v}_n(s)$, satisfies the recurrence

$$\vec{v}_n(s) = \max_{a \in A} \left\{ R(s, a) + \gamma \cdot \sum_{s' \in S} P(s, a)(s') \cdot \vec{v}_{n-1}(s') \right\} , \tag{1}$$

with $\vec{v}_0(s) = 0$. Consequently, a finite-horizon policy is optimal if and only if it chooses, in a situation when the current state is s and n steps are remaining, an action maximizing the right-hand side (RHS) of (1). Thus, to solve an MDP with a finite horizon H, the VI algorithm computes the values $\vec{v}_n(s)$ for all $0 \leq n \leq H$ and all states s, by iterating the recurrence (1). Using these values, VI then outputs (using some tie-breaking rule) some policy satisfying the aforementioned optimality characterization. VI can be deployed also for infinite-horizon MDPs: one can effectively compute a horizon H such that action a is optimal in state s for an infinite horizon[1] if it maximizes the RHS of (1) for $n = H$ [8]. This H has a bit-size which is polynomial in the size of the original MDP, but the magnitude of H can be exponential in the size of the MDP if the discount factor is given in binary [22].

VI is one of the most popular MDP-solving algorithms due to its versatility (as shown above, it can be used for several MDP-related problems) and conceptual simplicity, which makes it easy to implement within different programming paradigms [30, 37], including implementation via neural nets [35]. Several variants of VI with improved performance were developed [36, 14]. For instance, the recent paper by Sidford et al. [32] presented a new class of randomized VI techniques with the best theoretical runtime bounds (for certain values

[1] In infinite-horizon MDPs, there is always an optimal *stationary* policy, which makes decisions based only on the current state. [29]

of parameters) among all known MDP solvers. The paper also expresses hope that their techniques "*will be useful in the development of even faster MDP algorithms.*" To get insight into the underlying structure of VI, which might enable or limit further such accelerations, we take a complexity-theoretic vantage point and study the theoretical complexity of computing an *outcome* of a VI execution. That is, we consider the following decision problem ValIt: given an MDP with a finite horizon H (encoded as a binary number), does a given action a maximize the RHS of (1) for $n = H$? This problem is inspired by the paper of Fearnley and Savani [16], where they show **PSPACE**-hardness (and thus also completeness) for the problem of determining an outcome of *policy iteration*, another well-known algorithm for MDP solving. To the best of our knowledge, VI has not yet been explicitly subjected to this type of analysis. However, questions about the complexity of ValIt were implicitly raised by previous work on the complexity of finite-horizon MDPs, as discussed in the next paragraph.

Finite-horizon MDPs. The complexity of finite-horizon MDPs is a long-standing open problem. Since "finding an optimal policy" is a function problem, we can instead consider the decision variant: "In a given finite-horizon MDP, is it optimal to use a given action in the first step?" As discussed above, this is exactly the ValIt problem in disguise.

In the seminal 1987 paper on the complexity of MDPs [27], Papadimitriou and Tsitsiklis showed **P**-completeness of a special case of finite-horizon optimization where the horizon H has magnitude polynomial in the size of the MDP. At the same time, they noted that in the general case of binary-encoded H, VI can be executed on an **EXPTIME**-bounded Turing machine (since H is represented using $\log(H)$ bits, the number of iterations is exponential in the size of the input). Hence ValIt is in **EXPTIME**. However, the exact complexity of the general finite-horizon optimization remained open ever since, with the best lower bound being the **P**-hardness inherited from the "polynomial H" sub-problem. Tseng [36] presented a more efficient (though still exponential) algorithm for finite-horizon MDPs satisfying a certain stability condition; in the same paper, he comments that *"in view of the stability assumptions needed to obtain an exact solution and the absence of negative results, we are still far from a complete complexity theory for this problem."*

In this paper, we address this issue, provide the missing negative results, and *provide tight bounds on the computational complexity* of ValIt and finite-horizon MDP optimization.

Our Results

The main result of the paper is that ValIt is **EXPTIME**-complete (Theorem 1). In the rest of this section, we first explain some challenges we needed to overcome to obtain the result. Then we sketch our main techniques and conclude with discussing the significance of our results, which extends beyond MDPs to several areas of independent interest.

Challenges

Bitsize of numbers. One might be tempted to believe that ValIt is in **PSPACE**, since the algorithm needs to store only polynomially many values at a time. However, the bitsize of these values may become exponentially large during the computation (e.g., the quantity $\vec{v}_n(s)$ may halve in every step). Hence, the algorithm *cannot* be directly implemented by a polynomial-space Turing machine (TM). One could try to adapt the method of Allender et al. [20, 2] based on an intricate use of the Chinese remainder representation (CRR) of integers. However, there is no known way of computing the max operation directly and efficiently on numbers in CRR.

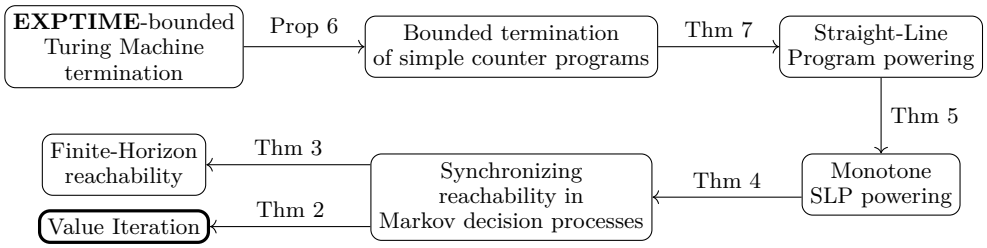


Figure 1 Chain of reductions.

Complex optimal policies. Another hope for **PSPACE** membership would be a possibly special structure of optimal policies. Fixing any concrete policy turns an MDP into a Markov chain, whose H-step behavior can be evaluated in polynomial space (using, e.g., the aforementioned CRR technique of Allender et al.). If we could prove that (A) an optimal policy can be represented in polynomial space and (B) that the Markov chain induced by such a policy is polynomially large in the size of the MDP, we would get the following **PSPACE** algorithm: cycle through all policies that satisfy (A) and (B), evaluate each of them, and keep track of the best one found so far. Tseng [36] commented that optimal policies in finite-horizon MDPs are "poorly understood". Hence, there was still hope that optimal Markovian deterministic policies may have a shape that satisfies both (A) and (B). Unless **PSPACE** = **EXPTIME**, our results put such hopes to rest.

No hardness by succinctness. One might try to prove **EXPTIME**-hardness using a *succinctness* argument. The results of [27] show that ValIt is **P**-hard when the horizon is written in unary, and many optimization problems over discrete structures incur an exponential blow-up in complexity when the discrete structure is encoded succinctly, e.g., by a circuit [28]. Giving a horizon H in binary amounts to a succinct encoding of an exponentially large MDP obtained by "unfolding" the original MDP into a DAG-like MDP of depth H. This unfolded MDP is "narrow" in the sense that it consists of many polynomial-sized layers, while standard **EXPTIME**-hardness-by-succinctness proofs, use succinct structures of an exponential "width" and "depth", accommodating the tape contents of an **EXPTIME**-bounded TM. Hence, straightforward succinctness proofs do not apply here; e.g., there does not seem to be a direct reduction from the succinct circuit value problem.

Our Techniques

To obtain **EXPTIME**-hardness of ValIt, we proceed by a sequence of non-trivial reductions. Below we outline these reductions in the order in which they appear in the sequence, see Figure 1. In the main text, we present the reductions in a different order (indicated by the numbering of propositions and theorems), so that we start with MDPs and gradually introduce more technical notions.

We start from a canonical **EXPTIME**-complete problem: the halting problem for an exponential-time TM. We then present a reduction to a halting problem for a class of counter programs (CPs; simple imperative programs with integer variables) that allow for linear variable updates. In this way, we encode the tape contents into numerical values (6). The crucial feature of this reduction is that the produced CP possesses a special *simplicity* property, which imposes certain restrictions on the use of tests during the computation.

Next, we introduce *straight-line programs (SLPs)* with max, +, and − operations. SLPs are a standard model of arithmetic computation [3] and they can be equivalently viewed

as arithmetic circuits consisting (in our case) of max, +, and − gates. We also consider a sub-class of SLPs with only max, + operations, so called *monotone SLPs*. We define the following *powering problem:* given a function $f\colon \mathbb{Q}^n \to \mathbb{Q}^n$ represented as an SLP, a horizon H, an initial argument $\vec{x} \in \{0,1\}^n$, and two indices $1 \leq i, j \leq n$, is it true that the i-component of $f^H(\vec{x})$, i.e. the image of $\vec{x}$ with respect to the H-fold composition of f, is greater than the j-component of $f^H(\vec{x})$? Although VI in MDPs does not necessarily involve integers, the powering problem for monotone SLPs captures the complexity inherent in iterating the recurrence (1). To obtain a reduction from CPs to SLP powering, we construct SLP gadgets with max, + and − (minus) operations to simulate the tests in CPs; the simplicity of the input CP is crucial for this reduction to work (Theorem 7). To get rid of the minus operation, we adapt a technique by Allender et al. [4], which introduces a new "offset" counter and models subtraction by increasing the value of the offset (Theorem 5).

A final step is to show a reduction from monotone SLP powering to ValIt. The reduction proceeds via an intermediate problem of *synchronizing reachability in MDPs* (maximize the probability of being in a target set of states T *after exactly* H steps [15]). This divides a rather technical reduction into more comprehensible parts. We present novel reductions from monotone SLP powering to synchronizing reachability (Theorem 4), and from the latter problem to ValIt (Theorem 2). As a by-product, we present a reduction proving **EXPTIME**-hardness of finite-horizon reachability in MDPs, arguably the conceptually simplest objective in probabilistic decision-making (Theorem 3).

Significance

As our main result, we characterize the complexity of computing an outcome of VI, one of the fundamental algorithms for solving both finite- and infinite-horizon MDPs. As a consequence, we resolve a long-standing complexity issue [27] of solving finite-horizon MDPs.

On our way to proving this result, we encounter non-trivial stepping stones which are of an independent interest. First, we shed light on the complexity of succinctly represented arithmetic circuits, showing that comparing two output wires of a given (max, +)-circuit incurs an exponential blow-up in complexity already when employing a very rudimental form of succinctness: composing a single (max, +)-circuit with H copies of itself, yielding a circuit of exponential "height" but only polynomial "width." Second, we obtain new hardness results for the bounded reachability problem in linear-update counter programs. CPs are related to several classical abstractions of computational machines, such as Minsky machines and Petri nets [25], see [13] for a recent breakthrough in this area. Our work establishes a novel connection between counter programs and MDPs.

Further Related Work

Our work is also related to a series of papers on finite-horizon planning [21, 17, 18, 23]. The survey paper [26] provides a comprehensive overview of these results. These papers consider either MDPs with a polynomially large horizon, or *succinctly* represented MDPs of possibly exponential "width" (the succinctness was achieved by circuit-encoding). The aforementioned hardness-by-succinctness proofs are often used here. The arbitrary horizon problem for standard MDPs, which we study, is left open in these papers, and our work employs substantially different techniques. The complexity of finite-horizon *decentralized* MDPs was studied in [7].

2 Markov Decision Processes and Finite-Horizon Problems

We start with some preliminaries. A *probability distribution* $d : S \to [0, 1]$ over a finite set S is a function such that $\sum_{s \in S} d(s) = 1$. We denote by $\mathcal{D}(S)$ the set of all (rational) probability distributions over S. The *Dirac* distribution on $s \in S$ assigns probability 1 to s.

A Markov decision process (MDP) $\mathcal{M} = (S, A, P, R, \gamma)$ consists of a finite set S of *states*, a finite set A of *actions*, a *transition function* $P \colon S \times A \to \mathcal{D}(S)$, a *reward function* $R : S \times A \to \mathbb{Q}$, and a *discount factor* $\gamma \in (0, 1]$. The transition function P assigns to each state s and action a a distribution over the successor states, while the reward function assigns to s and a a rational reward.

A *path* ϱ is an alternating sequence $s_0 a_1 s_1 \cdots a_n s_n$ of visited states and played actions in $\mathcal{M}$ (that starts and ends in a state); write $|\varrho| = n$ for the length of ϱ. We may use $s_0 \xrightarrow{\varrho} s_n$ to denote that path ϱ goes from s_0 to s_n. We extend the reward function R from single state-action pairs to paths by $R(\varrho) = \sum_{1 \leq i \leq n} R(s_{i-1}, a_i)\gamma^{i-1}$.

A *policy* for the controller is a function σ that assigns to each path a distribution over actions. Let $\mathbb{P}_{\mathcal{M},s,\sigma}(\varrho)$ denote the probability of a path ϱ starting in s when the controller follows the policy σ. This probability is defined inductively by setting $\mathbb{P}_{\mathcal{M},s,\sigma}(s_0) = 1$ if $s = s_0$, and $\mathbb{P}_{\mathcal{M},s,\sigma}(s_0) = 0$ otherwise. For a path $\varrho = s_0 a_1 s_1 \cdots s_{n-1} a_n s_n$, we set

$$\mathbb{P}_{\mathcal{M},s,\sigma}(\varrho) = \mathbb{P}_{\mathcal{M},s,\sigma}(s_0 \cdots s_{n-1}) \cdot \sigma(s_0 \cdots s_{n-1})(a_n) \cdot P(s_{n-1}, a_n)(s_n) \ .$$

We omit the subscripts from $\mathbb{P}_{\mathcal{M},s,\sigma}(\cdot)$ if they are clear from the context. Additionally, we extend $\mathbb{P}_{\mathcal{M},s,\sigma}(\cdot)$ to sets of paths of the same length by summing the probabilities of all the paths in the set.

In this paper, we focus on a special class of policies: A *(deterministic) Markov* policy is a function $\sigma : \mathbb{N} \times S \to A$. Intuitively, a controller following a Markov policy plays $\sigma(n, s)$ from s if it is the n-th visited state, irrespective of the other states in the path. Markov policies suffice for the problems we consider.

2.1 Finite-Horizon Problems

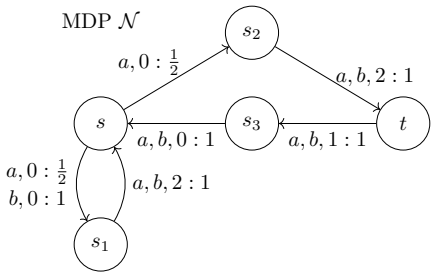
Given an MDP $\mathcal{M}$, the core problem of MDPs is computing the values of states with respect to the *maximum expected reward*. Let $\vec{v}_n \in \mathbb{Q}^S$ denote the vector of n-step maximum expected rewards obtainable from each state of the MDP. That is, for all $s \in S$ we have that

$$\vec{v}_n(s) = \max_{\sigma} \left(\sum_{|\varrho|=n} \mathbb{P}_{s,\sigma}(\varrho) \cdot R(\varrho) \right) \ .$$

Note that $\vec{v}_0 = \vec{0}$ by this definition. The vector $\vec{v}_n$ can be computed by *value iteration*, i.e. by iterating the recurrence stated in Equation (1). From that recurrence, for each $n \in \mathbb{N}$ and state s_0, one can extract an (optimal) Markov policy σ that achieves the maximum value $\vec{v}_n(s_0)$ after n steps: for each $s \in S$ and for $1 \leq i \leq n$ we have

$$\sigma(i-1, s) = \operatorname*{argmax}_{a \in A} \left\{ R(s, a) + \gamma \cdot \sum_{s' \in S} P(s, a)(s') \cdot \vec{v}_{n-i}(s') \right\} \ .$$

Papadimitriou and Tsitsiklis posed the **finite-horizon reward problem** which asks to compute such an optimal policy for the controller [27]. Formally, given an MDP $\mathcal{M}$, an initial state $s_0 \in S$, a distinguished action $a \in A$, and a horizon $H \in \mathbb{N}$ encoded in binary, the finite-horizon reward problem asks whether there exists a policy achieving $\vec{v}_H(s_0)$ by



$$\vec{v}_n(s) = \max\left(\frac{1}{4}\vec{v}_{n-1}(s_1) + \frac{1}{4}\vec{v}_{n-1}(s_2), \frac{1}{2}\vec{v}_{n-1}(s_1)\right)$$
$$\vec{v}_n(s_1) = 2 + \frac{1}{2}\vec{v}_{n-1}(s)$$
$$\vec{v}_n(s_2) = 2 + \frac{1}{2}\vec{v}_{n-1}(t)$$
$$\vec{v}_n(t) = 1 + \frac{1}{2}\vec{v}_{n-1}(s_3)$$
$$\vec{v}_n(s_3) = \frac{1}{2}\vec{v}_{n-1}(s)$$

Figure 2 The transitions are labelled with actions, rewards and their probabilities. For example, the reward of the transition from s to s_1 on action a is 0, and its probability is $\frac{1}{2}$.

choosing a as the first action from s_0. Note that this problem is equivalent to the ValIt problem defined in the introduction.

Consider the MDP $\mathcal{N}$ depicted in Figure 2 with $\gamma = \frac{1}{2}$. By iterating the indicated recurrence, we have that $\vec{v}_5(s) = \max(\frac{1}{4}\vec{v}_4(s_1) + \frac{1}{4}\vec{v}_4(s_2), \frac{1}{2}\vec{v}_4(s_1)) = \frac{41}{32}$. The value of $\vec{v}_5(s)$ is due to the second argument of max (corresponding to action b), hence a policy to maximize $\vec{v}_5(s)$ starts with b in s.

The finite-horizon reward problem can be decided by value iteration in exponential time by unfolding recurrence (1) for H steps [29], while the best known lower bound is **P**-hardness [27]. Our main result closes this long-standing complexity gap:

▶ **Theorem 1.** *The finite-horizon reward problem (and thus also the* ValIt *problem) is* **EXPTIME***-complete.*

To prove **EXPTIME**-completeness of the finite-horizon reward problem, we introduce a variant of reachability, which we call *synchronized reachability* [15]. Let $t \in S$ be a target state. For reachability, the objective is to maximize the probability of taking a path from s to t, whereas in synchronized reachability only a subset of such paths with the same length are considered.

Let $\mathcal{M}$ be an MDP, s_0 an initial state, and a an action. Define $\vec{p}_{\leq n} \in \mathbb{Q}^S$ as the vector of maximum probabilities of taking a path to t within n steps. Similarly, define $\vec{p}_{=n} \in \mathbb{Q}^S$ to be the vector of maximum probabilities of taking such a path with length exactly n. Formally, for all $s \in S$ we have that

$$\vec{p}_{\leq n}(s) = \max_{\sigma}\left(\mathbb{P}_{s,\sigma}(\{s \xrightarrow{\varrho} t : |\varrho| \leq n\})\right) \text{ and } \vec{p}_{=n}(s) = \max_{\sigma}\left(\mathbb{P}_{s,\sigma}(\{s \xrightarrow{\varrho} t : |\varrho| = n\})\right) .$$

Given a horizon H, encoded in binary, the **finite-horizon reachability problem** asks whether an optimal policy achieving $\vec{p}_{\leq H}(s_0)$ chooses action a as the first action from s_0; the **finite-horizon synchronized-reachability problem** asks whether an optimal policy achieving $\vec{p}_{=H}(s_0)$ chooses action a as the first action from s_0.

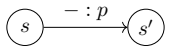
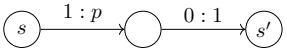
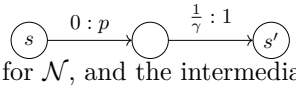
2.2 Connections Among Finite-Horizon Problems

We now prove the following theorem.

▶ **Theorem 2.** *The finite-horizon synchronized-reachability problem reduces, in polynomial time, to the finite-horizon reward problem.*

Consider an MDP $\mathcal{M}$, an initial state s_0, an action a and a target state t. The following recurrence can be used to compute $\vec{p}_{=n}(s)$:

$$\vec{p}_{=n}(s) = \max_{a \in A}\left\{\sum_{s' \in S} P(s,a)(s') \cdot \vec{p}_{=n-1}(s')\right\} , \tag{2}$$

where $\vec{p}_0(t) = 1$ and $\vec{p}_0(s) = 0$ for all $s \neq t$. We construct a new MDP $\mathcal{N}$ obtained from $\mathcal{M}$ by replacing all transitions by two consecutive transitions. The construction is such that the probability of going from s to t with a path of length n in $\mathcal{M}$ is equal to the probability of going from s to t with a path of length $2n$ in $\mathcal{N}$. More formally, for all s, s' and a with $P(s,a)(s') = p$, the transition  is replaced with  if $s = t$ and with  otherwise; where $0 < \gamma \leq 1$ is an arbitrary chosen discount factor for $\mathcal{N}$, and the intermediate state in both cases is a new state. The MDP $\mathcal{N}$ in Figure 2 is the result of applying the construction to $\mathcal{M}$ in Figure 3 with $\gamma = \frac{1}{2}$.

For the constructed MDP $\mathcal{N}$, one can show that for all states s, an action is optimal to maximize $\vec{p}_{=2H}(s)$ if and only if it is optimal to maximize $\vec{v}_{2H+1}(s)$. Consider the MDPs from Figure 2 as an example. We have previously argued that a policy maximizing $\vec{v}_5(s)$ in $\mathcal{N}$ starts with action b. Observe that the optimal first choice to maximize $\vec{p}_4(s)$ is also b. This implies that an optimal policy of $\mathcal{M}$ for synchronized-reachability with $H = 2$ starts with b, too. By the above argument, the finite-horizon synchronized-reachability problem reduces to the finite-horizon reward problem.

Hence, to obtain Theorem 1, it remains to determine the complexity of the finite-horizon synchronized-reachability problem. To this aim, we show a close connection between MDPs and a class of piecewise-affine functions represented by *straight line programs* (SLPs). Section 3 provides the details.

Finite-horizon reachability. We also show the finite-horizon synchronized-reachability problem reduces to the finite-horizon reachability problem. We remark that the natural probability-1 variants of these problems have different complexities: specifically, the problem of reaching t from s within H steps with probability 1 is in **P**; however, the analogous problem of reaching t from s in exactly H steps with probability 1 is **PSPACE**-complete [15].

▶ **Theorem 3.** *The finite-horizon synchronized reachability problem reduces, in polynomial time, to the finite-horizon reachability problem.*

3 Straight-Line Programs and The Powering Problem

We now establish the connection between MDPs and SLP powering. We start with preliminaries.

For all $n \in \mathbb{N}$, define the set $var_n := \{x_1, \ldots, x_n\}$ of variables and the collection of terms

$$\mathcal{T}_n := \{a_1 x_{j_1} + \cdots + a_n x_{j_n} + b \mid a_i, b \in \{-1, 0, 1\} \text{ and } 1 \leq j_i \leq n, \text{ for all } 1 \leq i \leq n\}.$$

A *straight-line program* (SLP) of *order* n is a sequence $c_1, \ldots, c_m$ of *commands* of the form $x \leftarrow \max(T)$, where $x \in var_n$ and $T \subseteq \mathcal{T}_n$ is non-empty. We refer to commands $x \leftarrow b$ as *initializations*. Recall that $\min(x, y) = -\max(-x, -y)$.

For complexity analyses we shall assume that T, for every command, is given explicitly as a list of terms. Each term is also assumed to be explicitly represented as a constant, a list of coefficients a_i, and a list of indices j_i, both lists having length n (i.e. the number of variables). The size of T, and also that of the command, corresponds to the length of its list of terms; the size of the SLP, the sum of the sizes of its commands.

A *valuation* ν is a vector in $\mathbb{Z}^n$, where the i-th coordinate gives the value of x_i. The semantics of a command c is a function $[\![c]\!] : \mathbb{Z}^n \to \mathbb{Z}^n$, transforming a valuation into another. An SLP $S = c_1, \ldots, c_m$ defines the function $[\![S]\!] : \mathbb{Z}^n \to \mathbb{Z}^n$ obtained by composing the

constituent commands: $[\![S]\!] = [\![c_m]\!] \circ \cdots \circ [\![c_1]\!]$. Clearly this is a piecewise-affine function. Given a function $f : \mathbb{Z}^n \to \mathbb{Z}^n$, we define its m-th power as $f^m : \mathbb{Z}^n \to \mathbb{Z}^n$ where

$$f^m = \underbrace{f \circ \cdots \circ f}_{m \text{ times}}$$

is the m-fold composition of f.

We denote by $\mathcal{T}_n^+$ the set of terms $a_1 x_{j_1} + \cdots + a_n x_{j_n} + b$ where the coefficients $a_1, \cdots, a_n, b$ are in $\{0, 1\}$. An SLP that only uses terms in $\mathcal{T}_n^+$ is called *monotone*. Note that monotone SLPs induce monotone functions from $\mathbb{Z}^n$ to $\mathbb{Z}^n$ (subtraction and min are not allowed).

3.1 The Powering Problem

For an SLP S of order n, a valuation $\nu \in \mathbb{N}^n$ and $m \in \mathbb{N}$ (encoded in binary), let $\nu' = [\![S]\!]^m(\nu)$. Given two variables $x, y \in var_n$ of the SLP, the **powering problem** asks whether $\nu'(x) \geq \nu'(y)$. Since the initial valuations ν are always non-negative, all valuations obtained by powering monotone SLPs are non-negative. The above problem is **P**-complete if the exponent m is written in unary [19].

Observe that all numbers generated by powering an SLP can be represented using exponentially-many bits in the bitsize of the exponent. It follows that the powered SLP can be explicitly evaluated in exponential time. We provide a matching lower bound in Section 4. Before that, we show the connection of SLP powering to MDPs.

3.2 Synchronized Reachability and SLP Powering

The connection is stated in the following Theorem.

▶ **Theorem 4.** *The powering problem for monotone SLPs reduces, in polynomial time, to the finite-horizon synchronized reachability problem in MDPs.*

To illustrate this reduction, let us consider the SLP S of order 2:

$$x_1 \leftarrow \max(x_1 + x_2, x_2 + x_2); \qquad x_2 \leftarrow \max(x_1 + x_1, x_1 + x_1).$$

This SLP is *normalized*, that is to say all its max commands have exactly two arguments $t_1, t_2 \in \mathcal{T}_n^+$ and furthermore t_1, t_2 have exactly two summands. (Note that focusing on normalized SLPs is no loss of generality.) We are interested in the 2-nd power of S with initial valuation $\nu(x_1) = 0$ and $\nu(x_2) = 1$. In Figure 3, two copies of S are shown on the right to visualize the concept of powering it. To obtain an MDP, we consider a set of actions $A = \{a, b\}$ and have each variable x_i become a state. In the example, s and t are the corresponding states for x_1 and x_2. The t_1, t_2 arguments of max commands determine the successors of actions a, b, respectively, where each successor has probability $\frac{1}{2}$. The command $x_1 \leftarrow \max(x_1 + x_2, x_2 + x_2)$ translates to $P(s, a)(s) = P(s, a)(t) = \frac{1}{2}$ and $P(s, b)(s) = 1$, as shown in the MDP in Figure 3. Since $\nu(x_2) = 1$, we make t a target state. Now the i-th iteration of value iteration of (2) (corresponding to the i-th step *before* the horizon) is tightly connected to the i-th power of the SLP. Indeed, letting $\nu_i = [\![S]\!]^i(\nu)$, one can prove that $\vec{p}_{=i}(s) = \frac{1}{2^i}\nu_i(s)$ and $\vec{p}_{=i}(t) = \frac{1}{2^i}\nu_i(t)$.

SLP vs. monotone SLP powering. It thus remains to provide a lower bound for the Monotone SLP powering problem. The crucial step, which we cover in Section 4, is providing lower bounds for the non-monotone variant. The remaining step from non-monotone to monotone powering can be made by adapting the techniques of Allender et al. [4].

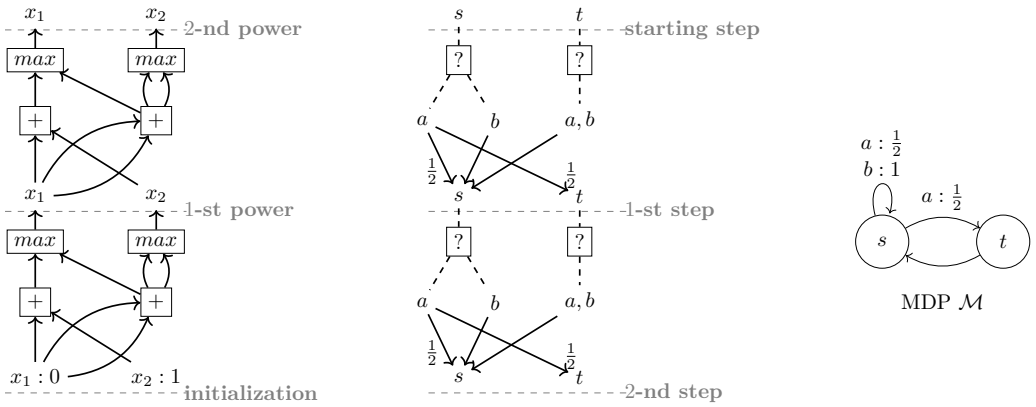


Figure 3 An example for the translation from SLPs to MDPs.

▶ **Theorem 5.** *The powering problem for arbitrary SLPs reduces, in polynomial time, to the powering problem for monotone SLPs.*

4 Main Reductions

To show **EXPTIME**-hardness of all the problems introduced so far, we introduce a class of counter programs that allow linear updates on counters and show that a (time-)bounded version of the termination problem for these programs is **EXPTIME**-complete. Finally, we reduce this bounded termination problem to the powering problem.

A *deterministic linear-update counter program* (CP) consists of n *counters* $\{c_i \mid 1 \leq i \leq n\}$, ranging over $\mathbb{Z}$, and a sequence of m instructions. We consider instructions of the form

$$p : c_1 \leftarrow c_2 + c_3 \qquad p : \texttt{if } c_1 \geq c_2 \texttt{ goto } t \qquad p : c_1 \leftarrow c_2 - c_3$$

where $1 \leq p < m$ and $1 \leq t \leq m$, and the final instruction is always $m : \texttt{halt}$. More precisely, the instructions allow

- **(i)** adding or subtracting two counters, assigning the result to a third one, and continuing to the next instruction;
- **(ii)** testing two counters against each other, and jumping to some given instruction if the result of the test is positive, continuing to the next instruction otherwise.

The `halt` instruction only loops to itself.

A *configuration* of a CP is a tuple $(p, v_1, \ldots, v_n) \in \{1, \ldots, m\} \times \mathbb{Z}^n$ consisting of an instruction p and values of the counters (e.g., v_1 is the value for the counter c_1). We equip CPs with a fixed initial configuration lying in $\{1\} \times \mathbb{N}^n$. Given a CP, the **termination problem** asks whether the `halt` instruction is reached. The **bounded termination problem** additionally takes as input an integer $N \in \mathbb{N}$, encoded in binary, and asks whether the `halt` instruction is reached within N steps.

The bounded termination problem is in **EXPTIME**: in a computation with N steps, the magnitude of the counters is bounded by 2^N, so each step can be simulated in time exponential in the bitsize of N. We will now show that the problem is **EXPTIME**-hard already for a certain subclass of CPs which facilitates the reductions to the powering problem.

Simple counter programs. A CP is *simple* if it satisfies the following conditions. First, all values in all reachable configurations $(p, v_1, \ldots, v_n)$ are non-negative: $v_i \in \mathbb{N}$ for all $1 \leq i \leq n$ (one may "guard" subtractions by test instructions to achieve this). Second, all test

instructions $q : \texttt{if } c_i \geq c_j \texttt{ goto } r$ use counters c_1 and c_2 exclusively. Moreover, for each such instruction q, there are counters $c_{\bar{q}_1}, c_{\bar{q}_2}$ such that in all reachable configurations $(q, v_1, \ldots, v_n)$ we have that

1. $v_1 = a_1 v_{\bar{q}_1}$ and $v_2 = a_2 v_{\bar{q}_2}$ with $a_1, a_2 \in \{64, 64 \cdot 10, 64 \cdot 12\}$. That is, the values of tested counters are "scaled-up" versions of the values of other counters.
2. Additionally, the absolute difference of the values of the tested counters is larger than the values of all other counters, in symbols $|v_1 - v_2| \geq \max\{v_k \mid 3 \leq k \leq n\}$.

Note that the class of simple CPs is a semantically defined subclass of all CPs. Further observe that for every test instruction we necessarily have that $\bar{q}_1, \bar{q}_2 \geq 3$.

The following proposition kick-starts our sequence of reductions.

▶ **Proposition 6.** *The bounded termination problem for simple CPs is* **EXPTIME***-complete.*

To prove the proposition, we follow the classical recipe of first simulating a Turing machine using a machine with two stacks, and then simulating the two-stack machine by a CP. We note two key differences between our construction and the classical reduction: (1) We use the expressiveness of linear updates in CPs to simulate pushing and popping on the stack in a linear number of steps of the CP. (2) We instrument the two-stack machine to ensure that the height of the two stacks differs by at most 1 along any computation. This is crucial to allow us to simulate the two-stack machine by a *simple* linear-update counter program.

4.1 From the Termination Problem to the Powering Problem

We now sketch the main ideas behind the last (and most technically involved) missing link in our sequence of reductions.

▶ **Theorem 7.** *The bounded termination problem for simple CPs reduces, in polynomial time, to the powering problem for SLPs.*

The encoding. Given a CP $\mathcal{C}$ we construct an SLP S of order $\geq 2n$ with variables including $\{x_1, \ldots, x_{2n}\}$. Let us denote x_{n+i} by Q_i for $1 \leq i \leq n$. The reduction is such that a configuration $(p, v_1, \ldots, v_n)$ of $\mathcal{C}$ is encoded as a valuation $\nu : var_{2n} \to \mathbb{Z}$ of the SLP with the property that $\nu(x_i) = v_i$ and $\nu(Q_i) = p\nu(x_i) = pv_i$ for all $1 \leq i \leq n$. In this way, the instruction p of the CP is encoded in the variables of the SLP (recall that SLPs are stateless).

Given this encoding, the main challenge is to realize the transition function of the CP as a function computed by an SLP. Once this is accomplished, for every $m \in \mathbb{N}$, the m-th power of the SLP S represents the m-step transition function of the CP.

Conditional commands. Intuitively, to encode the transition function we would like to equip the SLP with *conditional commands*, whose execution depends on a conditional. Specifically, we want to implement the following two kinds of conditional updates

$$(y \leftarrow y \pm x_k \text{ if } Q_k = px_k) \quad \text{and} \quad (Q_k \leftarrow p \cdot x_k \text{ if } x_i \geq x_j)$$

in terms of primitive commands of an SLP. In both commands, if the condition is not satisfied, the command is not executed, and the value of y or Q_k remains unchanged. For example, one can simulate the first type of conditional commands by executing $y \leftarrow y \pm \max(0, x_k + t)$, where t is an expression that is 0 if the test is passed and less than $-x_k$ otherwise. Intuitively, we think of t as "masking" the assignment if the test fails.

For the following result, which formalizes how we implement conditional commands, we call a valuation ν *valid* if there exists $q \in \{1, \ldots, m\}$ with $\nu(x_i) \geq 0$ and $\nu(Q_i) = q\nu(x_i)$ for all $1 \leq i \leq n$.

▶ **Lemma 8.** *Let $p \in \{1, \ldots, m\}$ and $i, j, k \in \{1, \ldots, n\}$ be distinct. The following equation holds for all valid valuations ν:*

$$\max(0, \nu(x_k) + \min(\nu(Q_k) - p\nu(x_k), p\nu(x_k) - \nu(Q_k))) = \begin{cases} \nu(x_k) & \text{if } \nu(Q_k) = p\nu(x_k) \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Moreover, if $|\nu(x_i) - \nu(x_j)| \geq \nu(x_k)$, then the following holds:

$$\max(0, \nu(x_k) + \min(0, \nu(x_i) - \nu(x_j))) = \begin{cases} \nu(x_k) & \text{if } \nu(x_i) \geq \nu(x_j) \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Proof. The equations follow directly from the assumption that ν is valid, since if $\nu(Q_j) \neq p\nu(x_j)$ then we also have $|\nu(Q_j) - p\nu(x_j)| \geq \nu(x_j)$. In addition, if $|\nu(x_i) - \nu(x_j)| \geq \nu(x_k)$ and $\nu(x_i) < \nu(x_j)$, we will have $\nu(x_k) + \nu(x_i) - \nu(x_j) \leq 0$. ◀

Using the property that the simulated program is simple, Equation (3) can be used to simulate the conditional update ($y \leftarrow y \pm x_k$ if $Q_k = px_k$) where $t = \min(Q_k - px_k, px_k - Q_k)$ masks the update. Likewise, Equation (4) can be used to simulate the second type of conditional update ($Q_k \leftarrow p \cdot x_k$ if $x_i \geq x_j$) where the masking expression is $t = \min(0, x_i - x_j)$. Finally, the multiplication-by-a-constant required for the second type of the conditional update is achieved via repeated addition.

Encoding the instructions. We recall that we encode being at the instruction p of the CP by a valuation ν such that $\nu(Q_i) = p\nu(x_i)$ for all $1 \leq i \leq n$.

Using the aforementioned conditional commands, we can construct the SLP S as the composition of m smaller SLPs. Each sub-SLP π_p simulates an instruction p from the given CP $\mathcal{C}$. Hence S, when applied upon a valid valuation ν (i.e., a properly-encoded configuration of $\mathcal{C}$), simulates all of its instructions at once. By using conditional commands, we make sure that only one sub-SLP results in a non-zero update: executing π_p has no effect on the valuation unless $p\nu(x_i) = \nu(Q_i)$ for all $1 \leq i \leq n$.

In this way, powering S allows us to simulate consecutive steps of $\mathcal{C}$. In particular, for all $N \in \mathbb{N}$ we have that $[\![S]\!]^N(\nu)(Q_1) \geq m \cdot [\![S]\!]^N(\nu)(x_1)$, where m is the *halt* instruction, holds if and only if $\mathcal{C}$ halts after at most N steps.

5 Conclusion

By the virtue of our chain of reductions (see Figure 1), we get the following theorem.

▶ **Theorem 9.** *All the following problems are* **EXPTIME***-complete:*

- *The finite-horizon reward problem for MDPs, and thus also the* ValIt *problem.*
- *The finite-horizon reachability and synchronized reachability problems for MDPs.*
- *The powering problem for SLPs and for monotone SLPs.*
- *The bounded termination problem for simple counter programs.*

The exact complexity of the following variant of the problem remains open: given an MDP and a horizon encoded in binary, determine whether there exists a policy achieving some given expected-reward threshold (with no restriction on the actions used to do so).

References

1 Pieter Abbeel and Andrew Y. Ng. Learning first-order Markov models for control. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1–8. MIT Press, 2005. URL: http://papers.nips.cc/paper/2569-learning-first-order-markov-models-for-control.pdf.

2 Eric Allender, Nikhil Balaji, and Samir Datta. Low-Depth Uniform Threshold Circuits and the Bit-Complexity of Straight Line Programs. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, volume 8635 of *Lecture Notes in Computer Science*, pages 13–24. Springer, 2014. doi:10.1007/978-3-662-44465-8_2.

3 Eric Allender, Peter Bürgisser, Johan Kjeldgaard-Pedersen, and Peter Bro Miltersen. On the complexity of numerical analysis. *SIAM Journal on Computing*, 38(5):1987–2006, 2009.

4 Eric Allender, Andreas Krebs, and Pierre McKenzie. Better complexity bounds for cost register automata. *Theory of Computing Systems*, pages 1–19, 2017.

5 Christel Baier and Katoen Joost-Pieter, editors. *Principles of Model Checking*. MIT Press, 2008.

6 Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

7 Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

8 Dimitri P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1987.

9 Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena scientific Belmont, MA, 2005.

10 Vincent D Blondel and John N Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274, 2000.

11 Nicole Bäuerle and Ulrich Rieder. *Markov Decision Processes with Applications to Finance*. Springer-Verlag Berlin Heidelberg, 2011.

12 Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Bloem Roderick, editors. *Handbook of Model Checking*. Springer International Publishing, 2018.

13 Wojciech Czerwinski, Slawomir Lasota, Ranko Lazic, Jérôme Leroux, and Filip Mazowiecki. The Reachability Problem for Petri Nets is Not Elementary (Extended Abstract). *CoRR*, abs/1809.07115, 2018. arXiv:1809.07115.

14 Peng Dai, Mausam, Daniel S. Weld, and Judy Goldsmith. Topological Value Iteration Algorithms. *J. Artif. Intell. Res.*, 42:181–209, 2011. URL: http://jair.org/papers/paper3390.html.

15 Laurent Doyen, Thierry Massart, and Mahsa Shirmohammadi. Limit Synchronization in Markov Decision Processes. In Anca Muscholl, editor, *Foundations of Software Science and Computation Structures - 17th International Conference, FOSSACS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings*, volume 8412 of *Lecture Notes in Computer Science*, pages 58–72. Springer, 2014. doi:10.1007/978-3-642-54830-7_4.

16 John Fearnley and Rahul Savani. The Complexity of the Simplex Method. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 201–208, New York, NY, USA, 2015. ACM. doi:10.1145/2746539.2746558.

17 Judy Goldsmith, Michael L Littman, and Martin Mundhenk. The complexity of plan existence and evaluation in probabilistic domains. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence (UAI'97)*, pages 182–189. Morgan Kaufmann Publishers Inc., 1997.

18 Judy Goldsmith and Martin Mundhenk. Complexity Issues in Markov Decision Processes. In *Proceedings of the 13th Annual IEEE Conference on Computational Complexity, Buffalo,*

New York, USA, June 15-18, 1998, pages 272–280. IEEE Computer Society, 1998. doi:10.1109/CCC.1998.694621.

19 R. Greenlaw, H.J. Hoover, and W.L. Ruzzo. *Limits to Parallel Computation: P-completeness Theory*. Oxford University Press, 1995. URL: https://books.google.fr/books?id=YZHnCwAAQBAJ.

20 William Hesse, Eric Allender, and David A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *J. Comput. Syst. Sci.*, 65(4):695–716, 2002. doi:10.1016/S0022-0000(02)00025-9.

21 Michael L Littman. Probabilistic propositional planning: Representations and complexity. In *AAAI'97*, pages 748–754, 1997.

22 Michael L. Littman, Thomas L. Dean, and Leslie Pack Kaelbling. On the Complexity of Solving Markov Decision Problems. In Philippe Besnard and Steve Hanks, editors, *UAI '95: Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence, Montreal, Quebec, Canada, August 18-20, 1995*, pages 394–402. Morgan Kaufmann, 1995. URL: https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=457&proceeding_id=11.

23 Michael L Littman, Judy Goldsmith, and Martin Mundhenk. The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research*, 9:1–36, 1998.

24 Yishay Mansour and Satinder P. Singh. On the Complexity of Policy Iteration. In Kathryn B. Laskey and Henri Prade, editors, *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30 - August 1, 1999*, pages 401–408. Morgan Kaufmann, 1999. URL: https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=192&proceeding_id=15.

25 Ernst W. Mayr. An algorithm for the general Petri net reachability problem. In *Proceedings of the thirteenth annual ACM Symposium on Theory of computing (STOC'81)*, pages 238–246. ACM, 1981.

26 Martin Mundhenk, Judy Goldsmith, Christopher Lusena, and Eric Allender. Complexity of Finite-horizon Markov Decision Process Problems. *J. ACM*, 47(4):681–720, July 2000. doi:10.1145/347476.347480.

27 Christos H. Papadimitriou and John N. Tsitsiklis. The Complexity of Markov Decision Processes. *Math. Oper. Res.*, 12(3):441–450, 1987. doi:10.1287/moor.12.3.441.

28 Christos H. Papadimitriou and Mihalis Yannakakis. A Note on Succinct Representations of Graphs. *Information and Control*, 71(3):181–185, 1986. doi:10.1016/S0019-9958(86)80009-2.

29 Martin L. Puterman. *Markov Decision Processes*. Wiley-Interscience, 2005.

30 Tim Quatmann and Joost-Pieter Katoen. Sound Value Iteration. In Hana Chockler and Georg Weissenbacher, editors, *Computer Aided Verification*, pages 643–661. Springer International Publishing, 2018.

31 Manfred Schäl. Markov decision processes in finance and dynamic options. In *Handbook of Markov Decision Processes*, International Series in Operations Research & Management Science, pages 461–487. Springer, 2002.

32 Aaron Sidford, Mengdi Wang, Xian Wu, and Yinyu Ye. *Variance Reduced Value Iteration and Faster Algorithms for Solving Markov Decision Processes*, pages 770–787. SIAM, 2018. doi:10.1137/1.9781611975031.50.

33 Olivier Sigaud and Olivier Buffet. *Markov Decision Processes in Artificial Intelligence*. John Wiley & Sons, 2013.

34 R.S. Sutton and A.G Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. MIT Press, 2018.

35 Aviv Tamar, YI WU, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value Iteration Networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2154–2162. Curran Associates, Inc., 2016. URL: http://papers.nips.cc/paper/6046-value-iteration-networks.pdf.

36 Paul Tseng. Solving H-horizon, Stationary Markov Decision Problems in Time Proportional to log(H). *Operations Research Letters*, 9(5):287–297, 1990.

37 Zhimin Wu, Ernst Moritz Hahn, Akin Günay, Lijun Zhang, and Yang Liu. GPU-accelerated value iteration for the computation of reachability probabilities in mdps. In Gal A. Kaminka, Maria Fox, Paolo Bouquet, Eyke Hüllermeier, Virginia Dignum, Frank Dignum, and Frank van Harmelen, editors, *ECAI 2016 - 22nd European Conference on Artificial Intelligence*, pages 1726–1727. IOS Press, 2016. doi:10.3233/978-1-61499-672-9-1726.

38 Yinyu Ye. A New Complexity Result on Solving the Markov Decision Problem. *Mathematics of Operations Research*, 30(3):733–749, 2005.

39 Yinyu Ye. The Simplex and Policy-Iteration Methods are Strongly Polynomial for the Markov Decision Problem with a Fixed Discount Rate. *Mathematics of Operations Research*, 36(4):593–603, 2011.

Monadic Decomposability of Regular Relations

Pablo Barceló
Department of Computer Science, University of Chile, Santiago, Chile
IMFD, Santiago, Chile
pbarcelo@dcc.uchile.cl

Chih-Duo Hong
Department of Computer Science, University of Oxford, UK
chih-duo.hong@st-hughs.ox.ac.uk

Xuan-Bach Le
Department of Computer Science, University of Oxford, UK
bachdylan@gmail.com

Anthony W. Lin
Technische Universität Kaiserslautern, Germany
anthony.lin@cs.uni-kl.de

Reino Niskanen
Department of Computer Science, University of Oxford, UK
reino.niskanen@cs.ox.ac.uk

Abstract

Monadic decomposibility – the ability to determine whether a formula in a given logical theory can be decomposed into a boolean combination of monadic formulas – is a powerful tool for devising a decision procedure for a given logical theory. In this paper, we revisit a classical decision problem in automata theory: given a regular (a.k.a. synchronized rational) relation, determine whether it is recognizable, i.e., it has a monadic decomposition (that is, a representation as a boolean combination of cartesian products of regular languages). Regular relations are expressive formalisms which, using an appropriate string encoding, can capture relations definable in Presburger Arithmetic. In fact, their expressive power coincide with relations definable in a universal automatic structure; equivalently, those definable by finite set interpretations in WS1S (Weak Second Order Theory of One Successor). Determining whether a regular relation admits a recognizable relation was known to be decidable (and in exponential time for binary relations), but its precise complexity still hitherto remains open. Our main contribution is to fully settle the complexity of this decision problem by developing new techniques employing infinite Ramsey theory. The complexity for DFA (resp. NFA) representations of regular relations is shown to be NLOGSPACE-complete (resp. PSPACE-complete).

2012 ACM Subject Classification Theory of computation → Regular languages; Theory of computation → Transducers; Theory of computation → Complexity classes; Theory of computation → Logic and verification; Theory of computation → Automated reasoning

Keywords and phrases Transducers, Automata, Synchronized Rational Relations, Ramsey Theory, Variable Independence, Automatic Structures

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.103

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version A full version of the paper is available at `https://arxiv.org/abs/1903.00728`.

Funding Barceló is funded by the Millennium Institute for Foundational Research on Data (IMFD) and Fondecyt grant 1170109. Le, Lin, and Niskanen are supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no 759969).

Acknowledgements We thank Leonid Libkin for the useful discussion.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 103; pp. 103:1–103:14

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

1 Introduction

Monadic decompositions for computable relations have been studied in many different guises, and applied to many different problem domains, e.g., see [17, 25, 38, 12, 27, 28, 37]. The notion of "monadic decomposability" essentially captures the intuitive notion that the components in a given n-ary relation $R \subseteq U^n$ are sufficiently independent from (i.e. not tightly coupled, or interdependent, with) each other. Some examples are in order. Given two subsets $X, Y \subseteq U$, then $X \times Y$ is an instance of relations whose two components are completely independent from each other. On the other hand, the equality relation $\{(x, x) : x \in U\}$ is an example of relations whose two components are tightly coupled. In this paper, we will adopt the commonly studied notion of component-independence[1] (e.g. [25, 38, 7, 37]) in a relation $R \subseteq U^n$ that lies between the extremes as exemplified in the above examples, i.e., that R is expressible as a *finite union* $\bigcup_{i=1}^{r} X_{i,1} \times \cdots \times X_{i,n}$ of products, where each $X_{i,j}$ is expressible in the same language $\mathcal{L}$ (e.g. a logic or a machine model) wherein R is expressed.

Why should one care about monadic decomposable relations? The main reason is that applying appropriate monadic restrictions could make an undecidable problem decidable, and in general turn a difficult problem into one more amenable to analysis. Several examples are in order. Firstly, the well-known cartesian abstractions in abstract interpretation [17] overapproximate the set $R \subseteq U^n$ of reachable states at a certain program point by a relation $R' \subseteq X_1 \times \cdots \times X_m$ such that $R \subseteq R'$. Having R' instead of R sometimes allows a static analysis tool to prove correctness properties about a program that is otherwise difficult to do with only R. Another example includes restrictions to monadic predicates in undecidable logics that result in decidability, e.g., monadic first-order logic and extensions ([9, 10, 4]), as well as monadic second-order theory of successors [10]. Monadic decomposability also found applications in more efficient variable elimination in constraint logic programming (e.g. [23]), as well as constraint processing algorithms for constraint database queries (e.g. [25, 24]). Finally, monadic decompositions in the context of SMT (Satisfiability Modulo Theories), whose study was recently initiated in [38], have numerous applications, including constraint solving over strings [38, 14].

The focus of this paper is to revisit a classical problem of determining monadic decomposability of *regular relations*, which are also known as *synchronized rational relations* [20, 6, 8]. The study of classes of relations over words definable by different classes of multi-tape (finite) automata is by now a well-established subfield of formal language theory. This study was initiated by Elgot, Mezei, and Nivat in the 1960s [18, 30]; also see the surveys [7, 15]. In particular, we have a strict hierarchy of classes of relations as follows: recognizable relations, synchronized rational relations, deterministic rational relations, and rational relations. All these classes over unary relations (i.e. languages) coincide with the class of regular languages. *Rational relations* are relations $R \subseteq (\Sigma^*)^n$ definable by multi-tape automata, where the tape heads move from left to right (in the usual way for finite automata) but possibly at different speeds (e.g. in a transition, the first head could stay at the same position, whereas the second head moves to the right by one position). *Deterministic rational relations* are simply those rational relations that can be described by deterministic multi-tape automata. So far, the heads of the tapes can move at different speeds. *Regular relations* (a.k.a. *synchronized rational relations*) are those relations that are definable by multi-tape automata, all of whose heads move to the right in each transition. Unlike (non)deterministic rational relations, regular relations are extremely well-behaved, e.g., they are closed under first-order operations

[1] Also called variable-independence.

and, therefore, have decidable first-order theories [22]. Regular relations are also known to coincide with those relations that are first-order definable over a universal automatic structure [6, 8]; equivalently, those relations that are definable by finite-set interpretations in the weak-monadic theory of one successor (WS1S) [16]. Finally, the weakest class of relations in the hierarchy are *recognizable relations*: those relations that are definable as a finite union of products of regular languages or, equivalently, relations that can be defined as a boolean combination of regular constraints (i.e. atomic formulas of the form $x \in L$, where L is a regular language, asserting that the word x is in L). Recognizable relations are, therefore, those relations definable by multi-tape automata that exhibit monadic decomposability.

One of the earliest results on deciding whether a relation is monadic decomposable follows from Stearns in 1967 [33] and the characterization of a binary relation $R \subseteq A^* \times B^*$ by $L_R = \{\mathsf{rev}(u)\#v \mid (u, v) \in R\}$, where $\mathsf{rev}(u)$ is the mirror image of u. In [12] it was proven that L_R is a regular language if and only if R has a monadic decomposition and if R is a deterministic rational relation, then L_R is a deterministic context-free language. Due to this characterization, Stearns's result implies that whether a deterministic n-ary rational relation is monadic decomposable (i.e. recognizable) is decidable in the case when $n = 2$. Shortly thereafter, Fischer and Rosenberg [19] showed that the same problem is unfortunately undecidable for the full class of binary rational relations. A few years later Valiant [37] improved the upper bound complexity for the case solved by Stearns to double exponential-time. This is still the best known upper bound for the monadic decomposability problem for deterministic binary rational relations to date and, furthermore, no specific lower bounds are known. More recently Carton *et al.* [12] adapted the techniques from [33, 37] to show that this decidability extends to general n-ary relations, though no complexity analysis was provided. The problem of monadic decomposability for regular relations has also been studied in the literature. Of course decidability with a double exponential-time upper bound for the binary case follows from [37]. In 2000 Libkin [25] gave general conditions for monadic decomposability for first-order theories, which easily implies decidability for monadic decomposability for general k-ary regular relations. This is because regular relations are simply those relations that are definable in a universal automatic structures [6, 8]. The result of Libkin was not widely known in the automata theory community and in fact the problem was posed as an open problem in French version of [31] in 2003 and later on, Carton *et al.* [12] provided a double-exponential-time algorithm for deciding whether an n-ary regular relation is monadic decomposable. More precisely, even though it was claimed in the paper that the algorithm runs in single-exponential time, it was noted in a recent paper by Löding and Spinrath [27, 28] (with which the authors of [12] also agreed, as claimed in [28]) that the algorithm actually runs in double-exponential time. Löding and Spinrath [27, 28] gave a single-exponential-time algorithm (inspired by techniques from [37]) for monadic decomposability of *binary* regular relations.

Contributions

In this paper we provide the precise complexity of monadic decomposability of regular relations, closing the open questions left by Carton *et al.* [12] and Löding and Spinrath [27, 28]. In particular, we show the following.

▶ **Theorem 1.** *Deciding whether a given regular relation R is monadic decomposable is* NLOGSPACE*-complete (resp.* PSPACE*-complete), if R is given by a DFA (resp. an NFA).*

The lower bounds hold already for binary relations (Lemma 5 and Lemma 6 in Section 3). To prove the upper bounds, we first prove the upper bounds for binary relations (Lemma 10 in Section 4) and then extend them to n-ary relations for any given $n > 2$ (Lemma 11 in Section 5).

The existing proof techniques (e.g. in [12, 28, 25]) for deciding monadic decomposability typically aim for finding proofs that the relations are monadic decomposable. In contrast, our proof technique relies on finding a proof that a relation is *not* monadic decomposable. As a brief illustration, suppose we want to show that the regular relation $R = \{(v, v) : v \in \Sigma^*\}$ is not monadic decomposable. We define an equivalence relation $\sim \subseteq \Sigma^* \times \Sigma^*$ as

$$x \sim y := \forall z([R(x, z) \leftrightarrow R(y, z)] \wedge [R(z, x) \leftrightarrow R(z, y)]).$$

This relation is regular since regular relations are closed under first-order operations [31] (a fact that was also used in [12]), but the size of the automaton for this relation is unfortunately quite large; see [27] for detailed discussion. Therefore, we will only use the complement $\not\sim$, which has a substantially smaller representation: polynomial (resp. exponential) size if R is given as a DFA (resp. an NFA). Now, that R is not monadic decomposable amounts to the existence of an ω-sequence $\sigma = \{v_i\}_{i\in\mathbb{N}}$ of words such that $v_i \not\sim v_j$ for each pair $i, j \in \mathbb{N}$. By applying the pigeonhole principle and König's lemma, we will first construct a nicer sequence α (see the top half of Figure 2) and then by exploiting Ramsey Theorem over infinite graphs, we will show that there is an even nicer sequence α' (see the bottom half of Figure 2), where the automaton for $\not\sim$ synchronizes its states in particular points of the computation, no matter which pair of words from the sequence is being read. Moreover, we prove that one of the synchronizing states has a pumping property. This leads to our NLOGSPACE algorithm as we can guess the synchronizing states and verify that there is an accepting run that can be pumped. This technique was inspired by a technique for proving recurrent reachability in regular model checking [34, 35].

The exponential-time upper bound for the binary case from Löding and Spinrath [28] (which is inspired by the techniques used by Stearns [33] and Valiant [37]) relied on characterization of a relation R using the language $L_R = \{\mathsf{rev}(u)\#v \mid (u, v) \in R\}$ and used a suitable machinery that is able to decide whether L_R is regular or not. Their result is not easily extensible to n-ary relations as the encoding of a binary rational relation as a context-free language L_R does not generalize to n-ary relations. In Section 5, we show that proving monadic decomposability for an n-ary regular relation is LOGSPACE-reducible to testing whether linearly many induced binary relations are monadic decomposable.

We conclude in Section 6 with some perspectives from formal verification and a future research direction. The proofs omitted due to length constraints can be found in [5].

2 Preliminaries

A finite alphabet is denoted by Σ and the free monoid it generates by Σ^*. That is, Σ^* consists of all finite words over Σ. The empty word is ε. We denote by $|w|$ the length of word $w \in \Sigma^*$. We have that $|\varepsilon| = 0$. The word $u \in \Sigma^*$ is a *prefix* of $w \in \Sigma^*$ if $w = uv$ for some $v \in \Sigma^*$. We denote this by $u \leq w$. We also write $v = u^{-1}w$, when u is a prefix of w, to state that v is the suffix of w that is obtained after prefix u is removed. Sometimes we want to consider a suffix of w after a prefix of particular length is removed without specifying the actual prefix as defined above. To this end, we define partial function $\sigma : \Sigma^* \times \mathbb{N} \to \Sigma^*$ such that $\sigma(w, i) = v$, where $w = uv$ for some $u \in \Sigma^*$ such that $|u| = i$. In particular, for $u \leq w$, $\sigma(w, |u|) = u^{-1}w$. Similarly, we define partial function $\tau : \Sigma^* \times \mathbb{N} \to \Sigma^*$ such that $\tau(w, i) = u$, where $|u| = i$ and $u \leq w$.

In this paper we study relations $R \subseteq \Sigma^* \times \cdots \times \Sigma^*$ with particular structural properties. Namely, *monadic decomposable* relations that are a finite union of direct products of regular languages, and *regular* relations defined by n-tape finite automata, where the heads move in synchronized manner. See, for example, [31] for more details on such relations.

▶ **Definition 2.** *An n-ary relation $R \subseteq \Sigma^* \times \cdots \times \Sigma^*$ is a* monadic decomposable *relation iff it is of the form $\bigcup_{i=1}^{m}(X_{1,i} \times \cdots \times X_{n,i})$, where m is finite and each $X_{j,i} \subseteq \Sigma^*$ is a regular language.*

As mentioned earlier, this can be intuitively seen as the components of R being independent in some sense. Note that in the literature, monadic decomposable relations are sometimes called *recognizable*. The monadic decomposable relations can be defined using multi-tape automata as is done, e.g., in [12]. The above definition is more suitable for our considerations.

Let $\perp$ be a fresh symbol not found in Σ. We use it to pad words in a relation $R \subseteq \Sigma^* \times \cdots \times \Sigma^*$ in order for each component to be of the same length. Formally, a tuple $(w_1, \ldots, w_n)$ is transformed into $(w_1\perp^{\ell_1}, \ldots, w_n\perp^{\ell_n})$, where $\ell_i = -|w_i| + \max_{1\leq j\leq n}|w_j|$ for each $i = 1, \ldots, n$. We extend this to the relation $R_\perp$ in the expected way. We also denote $\Sigma \cup \{\perp\}$ by $\Sigma_\perp$. An *n-tape automaton over alphabet $\Sigma_\perp$* is a tuple $(Q, \rightarrow_{\mathcal{A}}, q_0, F)$, where Q is the finite set of states, q_0 is the initial state, F is the set of final states, and $\rightarrow_{\mathcal{A}} \subseteq Q \times (\Sigma_\perp)^n \times \mathcal{P}(Q)$.

▶ **Definition 3.** *An n-ary relation $R \subseteq \Sigma^* \times \cdots \times \Sigma^*$ is* regular *iff $R_\perp$ is recognized by some n-tape automaton $\mathcal{A}_\perp$ over alphabet $\Sigma_\perp$.*

That is, in a regular relation the n heads of the automaton are moving in synchronized manner and the n-tuple of symbols seen determines the state transition. Naturally, the state transition can be deterministic or non-deterministic. We say that a regular relation is defined by an NFA if the underlying n-tape automaton is non-deterministic, otherwise we say that the relation is defined by a DFA. Note that in the literature, regular relations are sometimes called *synchronous rational* or *automatic* relations.

We recall a useful characterization from [12]. Consider an n-ary regular relation $R \subseteq \Sigma^* \times \cdots \times \Sigma^*$. For each $j = 1, \ldots, n-1$, let $\sim_j$ be the following induced equivalence relation:

$$\begin{aligned}(u_1, \ldots, u_j) \sim_j (v_1, \ldots, v_j) := \forall(w_{j+1}, \ldots, w_n) \in \Sigma^* \times \cdots \times \Sigma^* \text{ we have that}\\ (u_1, \ldots, u_j, w_{j+1}, \ldots, w_n) \in R \iff (v_1, \ldots, v_j, w_{j+1}, \ldots, w_n) \in R \text{ and}\\ (w_{j+1}, \ldots, w_n, u_1, \ldots, u_j) \in R \iff (w_{j+1}, \ldots, w_n, v_1, \ldots, v_j) \in R.\end{aligned}$$

▶ **Lemma 4** ([12])**.** *The n-ary regular relation R is monadic decomposable iff $\sim_j$ has finite index for each $j = 1, \ldots, n-1$. That is, there are finitely many equivalence classes over $\sim_j$.*

In other words, R is not monadic decomposable iff for some $j = 1, \ldots, n-1$, there is an infinite sequence $\{u_i\}_{i\geq 0}$, where each u_i is a j-tuple of words, such that for each $0 \leq i < \ell$ it is the case that $u_i \neq u_\ell$ and $u_i \not\sim_j u_\ell$.

In Section 4, we focus on binary relations for which we simplify the notation as there is only one possible value of j. We write $\sim$ instead of $\sim_j$ and $R^{\not\sim}$ for the binary regular relation

$$\begin{aligned}R^{\not\sim}(w, w') := \exists u \big((R(w, u) \wedge \neg R(w', u)) \vee (\neg R(w, u) \wedge R(w', u)) \vee\\ (R(u, w) \wedge \neg R(u, w')) \vee (\neg R(u, w) \wedge R(u, w'))\big).\end{aligned}$$

That is, $R^{\not\sim}$ consists of all words $w, w' \in \Sigma^*$ for which there exists a word $u \in \Sigma^*$ such that one of $R(w, u)$ and $R(w', u)$ is accepted while the other is not, or one of $R(u, w)$ and $R(u, w')$ is accepted while the other is not.

We assume that the reader is familiar with complexity classes and logarithmic space reductions via logarithmic space transducers; see for example [32].

3 Hardness of deciding monadic decomposability of regular relations

In this section, we consider binary regular relations given by NFA and provide a PSPACE lower bound for deciding if such a relation is monadic decomposable. Then, we prove that the same problem for DFA is NLOGSPACE-hard.

▶ **Lemma 5.** *The problem of deciding whether a binary regular relation given by an NFA is monadic decomposable is* PSPACE*-hard.*

Proof. We give a logarithmic space reduction from the universality problem for NFA, which is PSPACE-hard [29]. Recall that in this problem, we are asked to decide whether $L(\mathcal{A}) = \Sigma^*$ given an NFA $\mathcal{A}$ over Σ.

Let $\mathcal{A}$ be an NFA over alphabet Σ, and let $\{\#\}$ be a fresh symbol that we will use as a separator symbol. We assume that $\# \neq \bot$. We construct relation $R = R_1 \cup R_2$ using the language L of $\mathcal{A}$, where

$$R_1 = \{(u, u) \mid u \in (\Sigma \cup \{\#\})^*\} \qquad \text{and} \qquad R_2 = (L \cdot \{\#\})^* \times (\Sigma^* \cdot \{\#\})^*.$$

Intuitively, R_1 contains all pairs (w_1, w_2) such that $w_1 = w_2 = u_0\#u_1\# \cdots \#u_n\#$, where $u_i \in \Sigma^*$, and R_2 contains all pairs (w_1, w_2) such that $w_1 = v_0\#v_1\# \cdots \#v_m\#$, where $v_i \in L$, and $w_2 = u'_0\#u'_1\# \cdots \#u'_n\#$, where $u'_i \in \Sigma^*$. It is easy to construct an NFA that recognizes R in LOGSPACE. Next we show that $L = \Sigma^*$ iff R is monadic decomposable.

Assume first that $L = \Sigma^*$. Then $R_1 \subseteq R_2$, and thus $R = (\Sigma^* \cdot \{\#\})^* \times (\Sigma^* \cdot \{\#\})^*$ which has a trivial monadic decomposition.

For the other direction, assume that R is monadic decomposable, i.e., $R = \bigcup_{i=1}^{n}(A_i \times B_i)$ for some regular languages A_i, B_i. Let $w \in \Sigma^*$. We show that $w \in L$ as well. Consider a set $\{((w\#)^i, (w\#)^i) \mid i = 1, \ldots, n+1\} \subseteq R_1 \subseteq R$. By the pigeonhole principle, there are two elements $((w\#)^j, (w\#)^j)$ and $((w\#)^k, (w\#)^k)$ that belong to the same component of $\bigcup_{i=1}^{n}(A_i \times B_i)$, say to $A_1 \times B_1$. Therefore, $(w\#)^j \in A_1$ and $(w\#)^k \in B_1$, and hence their direct product, $((w\#)^j, (w\#)^k)$, is in $A_1 \times B_1 \subseteq R$. Recall that $R = R_1 \cup R_2$. Clearly, $((w\#)^j, (w\#)^k) \notin R_1$ as the lengths of the two words are different. It follows that $((w\#)^j, (w\#)^k) \in R_2$ and hence $(w\#)^j \in (L \cdot \{\#\})^*$. This implies that $w \in L$. ◀

▶ **Lemma 6.** *The problem of deciding whether a binary regular relation given by a DFA is monadic decomposable is* NLOGSPACE*-hard.*

The proof is straightforward by a reduction from reachability problem for directed acyclic graphs.

4 Deciding monadic decomposability of binary regular relations

In this section we prove our main technical result.

▶ **Lemma 7.** *There is an* NLOGSPACE *algorithm that takes as input an NFA for $R^{\curlywedge}$, where R is a binary regular relation, and decides whether R is monadic decomposable.*

We start by defining some notation. We assume any binary regular relation $R^{\curlywedge}$ to be given as an NFA with set of states Q. The *$R^{\curlywedge}$-type* of a pair (w_1, w_2) of words over Σ is an element of the *transition monoid*. Recall that the transition monoid transforms any given state $q \in Q$ to a set $Q' \subseteq Q$ of states when reading (w_1, w_2). We denote this by $R^{\curlywedge}_{w_1,w_2}(q)$ for each $q \in Q$. We write $\mathsf{types}(R^{\curlywedge})$ for the set of all $R^{\curlywedge}$-types.

Consider an infinite sequence $\{w_i\}_{i\geq 0}$ of words over Σ as defined in Lemma 4. Additionally, we assume that the words in the sequence are of strictly increasing length and that for each $i > 0$ the words w_i and w_{i+1} have a common prefix of length $|w_{i-1}|$. That is, w_i can be written as $\beta_0 \cdots \beta_{i-1}\alpha_i$, where each β_j and α_i is a non-empty word. To simplify notation, we denote $\rho(w_i) = \beta_0 \cdots \beta_i$. That is, $\rho(w_i)$ is of length $|w_i|$ and is a prefix of w_j, for each $0 \leq i < j$. We will show how to construct such sequence in Proposition 8. The words w_i, w_j and w_k are illustrated in the top of Figure 1.

With each pair (i, j), where $i < j$, we associate the following quinary tuple over $\mathsf{types}(R^\asymp)$:

$$\mathfrak{C}_{i,j} \;=\; \big(\, R^\asymp_{w_i,\rho(w_i)},\; R^\asymp_{\rho(w_i),\rho(w_i)},\; R^\asymp_{\sigma(w_j,|w_i|),\sigma(\rho(w_j),|w_i|)},\; R^\asymp_{\varepsilon,\sigma(w_j,|w_i|)},\; R^\asymp_{\varepsilon,\sigma(\rho(w_j),|w_i|)} \,\big).$$

Intuitively, the first component corresponds to the computation of $(\beta_0 \cdots \beta_{i-1}\alpha_i, \beta_0 \cdots \beta_{i-1}\beta_i)$, the second to $(\beta_0 \cdots \beta_{i-1}\beta_i, \beta_0 \cdots \beta_{i-1}\beta_i)$ needed in order to compute the third component, $(\beta_{i+1} \cdots \beta_{j-1}\alpha_j, \beta_{i+1} \cdots \beta_{j-1}\beta_j)$. The final two components are used to compute the set of states reachable after the whole word in the first component is read. That is $(\bot^{|\beta_{i+1}\cdots\beta_{j-1}\alpha_j|}, \beta_{i+1} \cdots \beta_{j-1}\alpha_j)$ and $(\bot^{|\beta_{i+1}\cdots\beta_{j-1}\beta_j|}, \beta_{i+1} \cdots \beta_{j-1}\beta_j)$. See Figure 1 for a pictorial depiction.

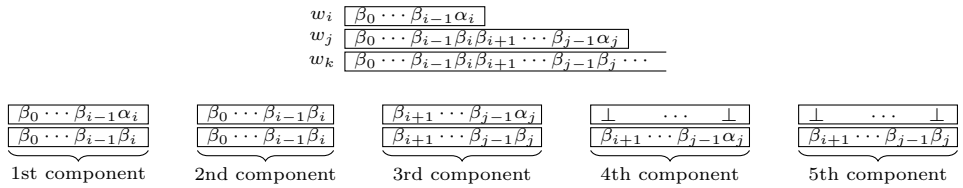


Figure 1 Correspondence between components of $\mathfrak{C}_{i,j}$ and parts of computation on w_i, w_j and w_k, where $i < j < k$.

We can then establish the following important proposition. Consider an infinite sequence of words that are pairwise from different equivalence classes as in Lemma 4. We show next that we can extract an infinite subsequence with additional structural properties. Perhaps the most important property is that $\mathfrak{C}_{i,j}$ is the same for all i, j. This subsequence will allow us to prove the main lemma.

▶ **Proposition 8.** *A binary regular relation R over $\Sigma^* \times \Sigma^*$ is not monadic decomposable iff there are infinite sequences $\{u_i\}_{i\geq 0}$, $\{\gamma_i\}_{i\geq 0}$, and $\{\delta_i\}_{i\geq 0}$ of words over Σ and a quinary tuple $\mathfrak{C}$ over $\mathsf{types}(R^\asymp)$ such that for each $i \geq 0$ it is the case that*

1. $|\gamma_i| = |\delta_i| > 0$,
2. $u_i = \delta_0 \cdots \delta_{i-1}\gamma_i$,
3. $(u_i, u_j) \in R^\asymp$, *for each* $j > i$, *and*
4. $\mathfrak{C}_{i,j} = \mathfrak{C}$, *for each* $j > i$.

Proof. By Lemma 4, the existence of such sequences directly implies that the relation is not monadic decomposable. Assume then that R is not monadic decomposable. By Lemma 4, there exists a sequence $\{v_i\}_{i\geq 0}$ such that $R^\asymp(v_j, v_\ell)$ for all $j \neq \ell$. It remains to show how to construct the three sequences satisfying the additional properties from $\{v_i\}_{i\geq 0}$. First, we construct an auxiliary sequence $\{w_i\}_{i\geq 0}$ in the following way. Let v_j be the first non-empty word of $\{v_i\}_{i\geq 0}$. Denote $v_j = w'_0 = \alpha_0$. Consider prefixes of v_i of length $|\alpha_0|$. Since $|\alpha_0|$ is finite and the sequence is infinite, there exists a prefix that appears infinitely often by the pigeonhole principle. Denote this prefix by β_0. Now we consider an infinite subsequence

$\{w'_i\}_{i\geq 0}$ of $\{v_i\}_{i\geq 0}$ where $w'_0 = v_j$ and w'_i, where $i > 0$, has β_0 as the proper prefix. We can write $w'_1 = \beta_0\alpha_1$ and repeat the procedure. By König's Lemma, we can always repeat the procedure and obtain the desired auxiliary sequence $\{w_i\}_{i\geq 0}$ in the limit.

From Infinite Ramsey's Theorem, there is an infinite sequence $0 \leq \ell_0 < \ell_1 < \cdots$ and a tuple $\mathfrak{C} \in \mathsf{types}(R^\curlywedge)^5$ such that for each $0 \leq i < j$ we have $\mathfrak{C}_{\ell_i,\ell_j} = \mathfrak{C}$. Namely, we consider a complete infinite graph with natural numbers as vertices. An edge between vertices i and j is coloured with $\mathfrak{C}_{i,j} \in \mathsf{types}(R^\curlywedge)^5$. Now there is an infinite clique coloured with $\mathfrak{C}$ which gives us our infinite sequence $0 \leq \ell_0 < \ell_1 < \cdots$.

We then define the u_is, γ_is, and δ_is, for $i \geq 0$, as follows.

- $\gamma_0 = w_{\ell_0}$ and γ_{i+1}, for $i > 0$, is the word $\sigma(w_{\ell_{i+1}}, |w_{\ell_i}|)$.
- δ_i is defined as $\rho(\gamma_i)$.
- $u_i = \delta_0 \cdots \delta_{i-1}\gamma_i$, for each $i \geq 0$.

It is easy to see then that $u_i = w_{\ell_i}$ and $\rho(u_i) = \delta_0 \cdots \delta_{i-1}\delta_i = \rho(w_{\ell_i})$, for each $i \geq 0$. Therefore, $\{u_i\}_{i\geq 0}$, $\{\gamma_i\}_{i\geq 0}$, $\{\delta_i\}_{i\geq 0}$, and $\mathfrak{C}$ satisfy the conditions in the statement of the proposition. See Figure 2 for a pictorial depiction of the construction. ◀

In other words, by Proposition 8, there is a sequence $\{u_i\}_{i\geq 0}$ and a $\mathfrak{C}$ such that for each i, j, the runs on $R^\curlywedge$ are synchronized after (γ_i, δ_i), (δ_i, δ_i), $(\delta_i^{-1}\gamma_j, \delta_i^{-1}\delta_j)$, $(\varepsilon, \delta_i^{-1}\gamma_j)$ and $(\varepsilon, \delta_i^{-1}\delta_j)$ have been read. In particular, the runs are synchronized in states of $R^\curlywedge_{\gamma_i,\delta_i}$, $R^\curlywedge_{\delta_i,\delta_i}$, $R^\curlywedge_{\delta_i^{-1}\gamma_j,\delta_i^{-1}\delta_j}$, $R^\curlywedge_{\varepsilon,\delta_i^{-1}\gamma_j}$ and $R^\curlywedge_{\varepsilon,\delta_i^{-1}\delta_j}$, respectively.

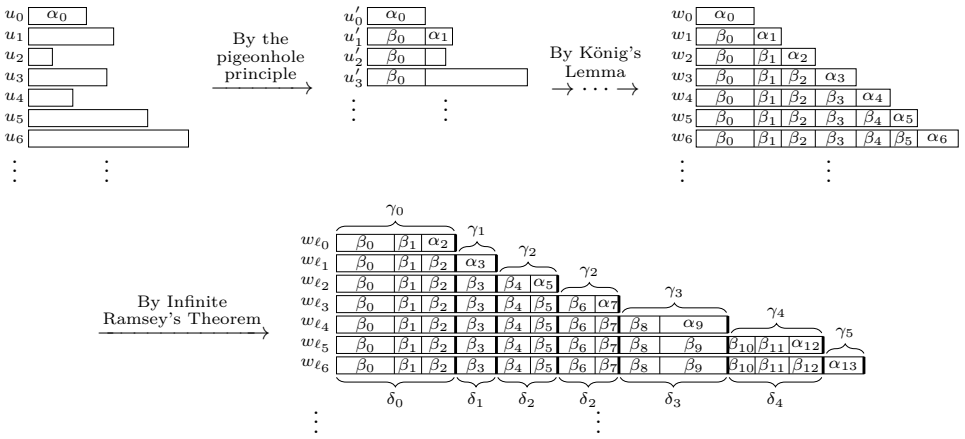


Figure 2 An illustration of construction of sequence $\{u_i\}_{i\geq 0}$ of Proposition 8 in two steps. Here $R^\curlywedge(u_i, u_j)$, $R^\curlywedge(u'_i, u'_j)$ and $R^\curlywedge(w_i, w_j)$ for every $i \neq j$. Moreover as $\mathfrak{C} = \mathfrak{C}_{i,j}$, the sets of states reachable after each δ_i and γ_i are the same (indicated by thick lines).

We can then prove the following crucial result. We assume here that R is a binary regular relation over $\Sigma \times \Sigma$ such that $R^\curlywedge$ is given as an NFA over $\Sigma \times \Sigma$ whose set of states is Q. We further assume that q_0 is the initial state of $R^\curlywedge$ and F its set of final states.

▶ **Lemma 9.** *Relation R is not monadic decomposable iff there are an infinite sequence $\{(x_i, y_i)\}_{i\geq 0}$ of pairs of words over Σ and states $q, q', p, r \in Q$, such that $p \in F$, it is the case that $q \in R^\curlywedge_{x_0,y_0}(q_0)$, and the following statements hold for each $i \geq 0$.*

1. *$|x_i| = |y_i|$ and y_i is a prefix of both x_{i+1} and y_{i+1}.*
2. *$q' \in R^\curlywedge_{y_i,y_i}(q_0)$; $\quad q \in R^\curlywedge_{y_i^{-1}x_{i+1},y_i^{-1}y_{i+1}}(q')$; $\quad p \in R^\curlywedge_{\varepsilon,y_i^{-1}x_{i+1}}(q)$; $\quad r \in R^\curlywedge_{\varepsilon,y_i^{-1}y_{i+1}}(q)$.*
3. *If $i > 0$, we also have that $p \in R^\curlywedge_{\varepsilon,y_i^{-1}x_{i+1}}(r)$ and $r \in R^\curlywedge_{\varepsilon,y_i^{-1}y_{i+1}}(r)$.*

Proof. Assume first that R is not monadic decomposable. By Proposition 8, there are infinite sequences $\{u_i\}_{i\geq 0}$, $\{\gamma_i\}_{i\geq 0}$, and $\{\delta_i\}_{i\geq 0}$ of words over Σ and a quinary tuple $\mathfrak{C}$ over $\mathsf{types}(R^\curlywedge)$ such that for each $i \geq 0$ it is the case that

1. $|\gamma_i| = |\delta_i| > 0$,
2. $u_i = \delta_0 \cdots \delta_{i-1}\gamma_i$,
3. $(u_i, u_j) \in R^\curlywedge$, for each $j > i$, and
4. $\mathfrak{C}_{i,j} = \mathfrak{C}$, for each $j > i$.

We then define a sequence $\{(x_i, y_i)\}_{i\geq 0}$ such that $x_i := u_i$, for each $i \geq 0$, and y_i is the prefix of $x_{i+1} = u_{i+1}$ that has the same length as $x_i = u_i$, i.e., $y_i = \tau(x_{i+1}, |x_i|)$. Hence, $y_i = \rho(u_i) = \delta_0 \cdots \delta_i$. Clearly, $|x_i| = |y_i| \geq 0$ and y_i is a prefix of both x_{i+1} and y_{i+1}, for each $i \geq 0$. We prove next that the sequence $\{(x_i, y_i)\}_{i\geq 0}$ also satisfies the remaining conditions.

Before defining $q, q', p, r \in Q$, let us highlight the intuition why such states exist for every i. We can find such states because by our assumption $\mathfrak{C}_{i,j} = \mathfrak{C}$ for each $i < j$. Further, whether q is reachable from q_0 is stored in the first component of $\mathfrak{C}$. Similarly, the second and third components of $\mathfrak{C}$ allow us to find q' that is reachable from q_0 and such that q is reachable from q'. Finally, the fourth component is for checking whether p is reachable from q and r, while the fifth component for checking that r is reachable from both q and r.

Let us define $q, q', p, r \in Q$ as follows.

- q and p are states such that $p \in F$ and it is the case that $q \in R^\curlywedge_{x_0,y_0}(q_0)$ and $p \in R^\curlywedge_{\varepsilon,y_0^{-1}x_1}(q)$. Notice that such q and p must exist as $(x_0, x_1) \in R^\curlywedge$, i.e., it holds that $R^\curlywedge_{x_0,x_1}(q_0) \cap F \neq \emptyset$, and $R^\curlywedge_{x_0,x_1}(q_0) = R^\curlywedge_{x_0,y_0}(q_0) \circ R^\curlywedge_{\varepsilon,y_0^{-1}x_1}$.
- q' is a state such that $q' \in R^\curlywedge_{y_0,y_0}(q_0)$ and $q \in R^\curlywedge_{y_0^{-1}x_1,y_0^{-1}y_1}(q')$. Notice that such a q' must exist. Indeed, since $\mathfrak{C}_{0,1} = \mathfrak{C}_{1,2} = \mathfrak{C}$, we have $R^\curlywedge_{u_0,\rho(u_0)} = R^\curlywedge_{x_0,y_0} = R^\curlywedge_{u_1,\rho(u_1)} = R^\curlywedge_{x_1,y_1}$. This implies that $q \in R^\curlywedge_{x_1,y_1}(q_0) = R^\curlywedge_{y_0,y_0}(q_0) \circ R^\curlywedge_{y_0^{-1}x_1,y_0^{-1}y_1}$, as we know that $q \in R^\curlywedge_{x_0,y_0}(q_0)$ and there must be an intermediate state q' that is reached after reading (y_0, y_0).
- We have that r is a state such that

$$r \in R^\curlywedge_{\varepsilon,y_0^{-1}y_1}(q); \qquad p \in R^\curlywedge_{\varepsilon,y_1^{-1}x_2}(r); \qquad \text{and} \qquad r \in R^\curlywedge_{\varepsilon,y_1^{-1}y_2}(r).$$

  The existence of such state r is not obvious but straightforward; see [5].

We now prove that q, q', p, r satisfy all the requirements in the statement of the Lemma. By definition, $q \in R^\curlywedge_{x_0,y_0}(q_0)$ and $p \in F$. We can then prove by induction that for each $i \geq 0$ it is the case that

$$q' \in R^\curlywedge_{y_i,y_i}(q_0); \quad q \in R^\curlywedge_{y_i^{-1}x_{i+1},y_i^{-1}y_{i+1}}(q'); \quad p \in R^\curlywedge_{\varepsilon,y_i^{-1}x_{i+1}}(q); \quad r \in R^\curlywedge_{\varepsilon,y_i^{-1}y_{i+1}}(q);$$

and, in addition, that for each $i > 0$ it is the case that $p \in R^\curlywedge_{\varepsilon,y_i^{-1}x_{i+1}}(r)$ and $r \in R^\curlywedge_{\varepsilon,y_i^{-1}y_{i+1}}(r)$. The base case $i = 0$ holds by definition. The inductive case is straightforward.

Let us assume now that there are an infinite sequence $\{(x_i, y_i)\}_{i\geq 0}$ of pairs of words over Σ and states $q, q', p, r \in Q$ that satisfy the conditions stated in the statement of the lemma. We prove next that R is not monadic decomposable by showing that there are infinite sequences $\{w_i\}_{i\geq 0}$, $\{\alpha_i\}_{i\geq 0}$ and $\{\beta_i\}_{i\geq 0}$ of words over Σ such that $\{w_i\}_{i\geq 0}$, $\{\alpha_i\}_{i\geq 0}$, and $\{\beta_i\}_{i\geq 0}$ satisfy the conditions stated in Lemma 4.

We define $w_i := x_i$ for each $i \geq 0$. Furthermore, $\alpha_0 := x_0$, $\beta_0 := y_0$, and for each $i > 0$ we set $\alpha_i := y_{i-1}^{-1}x_i$ and $\beta_i := y_{i-1}^{-1}y_i$. Clearly $|\alpha_i| = |\beta_i| > 0$ and $w_i = x_i = \beta_0 \cdots \beta_{i-1}\alpha_i$, for each $i \geq 0$. We prove next that $(w_i, w_j) \in R^\curlywedge$ for each $0 \leq i < j$. Actually, we prove a stronger claim: $p \in R^\curlywedge_{w_i,w_j}(q_0)$ and $r \in R^\curlywedge_{w_i,\rho(w_j)}(q_0)$, for each $0 \leq i < j$, where as before $\rho(w_j) = \tau(w_{j+1}, |w_j|) = \beta_0\beta_1 \cdots \beta_j$. The claim can be proved by induction. ◀

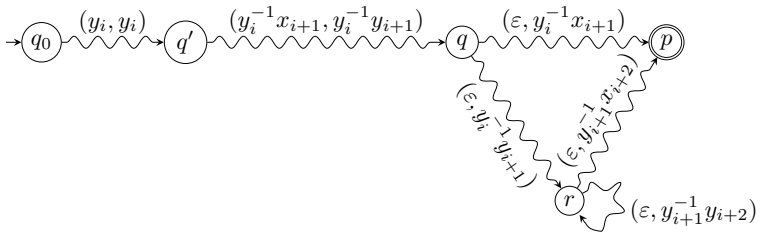


Figure 3 Runs in $R^{\nearrow}$ on states q, q', p and r as defined in Lemma 9. The runs exist for every $i \geq 0$.

The runs as extracted from the sequence $\{(x_i, y_i)\}_{i\geq 0}$ satisfying the conditions of Lemma 9 are depicted in Figure 3.

Lemma 9 allows us to reduce the monadic decomposability problem to a set of reachability checks on types. With the help of this property, we can then prove Lemma 7.

Proof of Lemma 7. For each $(q, q', p, r) \in Q \times Q \times Q \times Q$ with $p \in F$ do the following.

- Check if there are words w_0, v_0, w_1, v_1 such that $|w_0| = |v_0| > 0$, $|w_1| = |v_1| > 0$, and it holds that (i) $q \in R^{\nearrow}_{w_0,v_0}(q_0)$, (ii) $q' \in R^{\nearrow}_{v_0,v_0}(q_0)$, (iii) $q \in R^{\nearrow}_{w_1,v_1}(q')$, (iv) $q' \in R^{\nearrow}_{v_1,v_1}(q')$, (v) $p \in R^{\nearrow}_{\varepsilon,w_1}(q)$, and (vi) $r \in R^{\nearrow}_{\varepsilon,v_1}(q)$.
- Check if there are words w, v such that $|w| = |v| > 0$, and it holds that (i) $q \in R^{\nearrow}_{w,v}(q')$, (ii) $q' \in R^{\nearrow}_{v,v}(q')$, (iii) $p \in R^{\nearrow}_{\varepsilon,w}(q)$, (vi) $r \in R^{\nearrow}_{\varepsilon,v}(q)$, (v) $p \in R^{\nearrow}_{\varepsilon,w}(r)$, and (vi) $r \in R^{\nearrow}_{\varepsilon,v}(r)$.

If this holds for any such a tuple, then R is not monadic decomposable. Else, R is monadic decomposable. It is easy to see that this algorithm can be implemented in NLOGSPACE. ◀

We have the necessary ingredients to prove a part of Theorem 1.

▶ **Lemma 10.** *Deciding whether a given binary regular relation R is monadic decomposable is in* NLOGSPACE *(resp. in* PSPACE*), if R is given by a DFA (resp. an NFA).*

Proof. The claim follows from Lemma 7. Namely, from the definition of $R^{\nearrow}$, it follows that, if R is given by a DFA, then $R^{\nearrow}$ can be constructed in LOGSPACE. Indeed, this can be done as disjunctions, conjunctions and projections can all be done in LOGSPACE and then via composability of LOGSPACE transducers we can construct $R^{\nearrow}$ of logarithmic size. (Note that the output of a LOGSPACE transducer is of at most polynomial size.) Then by Lemma 7, we obtain the decidability of monadic decomposability in NLOGSPACE for R given by a DFA.

Similarly, if R is given by an NFA, we construct $R^{\nearrow}$ of polynomial size since an NFA can be transformed into a DFA using a PSPACE transducer. (Again, the output of a PSPACE transducer is of at most exponential size.) Thus monadic decomposability is in PSPACE. ◀

5 Deciding monadic decomposability of regular relations

In this section, we finish the proof of Theorem 1. The remaining component is showing that monadic decomposability of n-ary regular relations is decidable in NLOGSPACE for DFA and PSPACE for NFA.

▶ **Lemma 11.** *Deciding whether a given n-ary regular relation R is monadic decomposable is in* NLOGSPACE *(resp. in* PSPACE*), if R is given by a DFA (resp. an NFA).*

Proof of Theorem 1. The upper bounds follow from Lemma 11 and the lower bound follows from Lemma 5 for NFA and from Lemma 6 for DFA. ◀

In order to prove Lemma 11, we extend Lemma 10 to n-ary relations. Let us first define some helpful notation used throughout the section.

Recall that words of regular relations are padded to be of the same length using $\perp$. We denote this function by $\mathrm{PAD}_\perp$. For example, $\mathrm{PAD}_\perp((a,\varepsilon,ab)) = (a\perp,\perp\perp,ab)$. Let us now define a padding function δ_n that acts slightly differently. Instead of padding the words in a tuple to make them of the same length, the new function pads a sequence of tuples with tuples where some elements are $\perp$. Let us describe δ_n in more details. Define $\Sigma_n = (\Sigma_\perp)^n \setminus \{\perp^n\}$, i.e., an alphabet consisting of n-tuples of letters from $\Sigma_\perp$, excluding $(\perp,\ldots,\perp)$. Now $\delta_n : (\Sigma^*)^n \to \Sigma_n^*$ is an injective mapping that uses $\perp$ to extend the shorter words to the same length as the longest word. For example, δ_3 maps $(a,\varepsilon,ab) \in (\Sigma^*)^3$ to $(a,\perp,a)(\perp,\perp,b) \in \Sigma_3^*$ as follows:

$$(a,\varepsilon,ab) \to \begin{pmatrix} a \\ \varepsilon \\ ab \end{pmatrix} \to \begin{pmatrix} a\perp \\ \perp\perp \\ ab \end{pmatrix} \to \begin{pmatrix} a \\ \perp \\ a \end{pmatrix} \begin{pmatrix} \perp \\ \perp \\ b \end{pmatrix} \to (a,\perp,a)(\perp,\perp,b).$$

▶ **Lemma 12.** *For $n \geq 1$, $\{(x_1,\ldots,x_n,y) \mid \delta_n(x_1,\ldots,x_n) = y\} \subseteq (\Sigma^*)^n \times \Sigma_n^*$ is regular.*

Given an n-ary relation $R \subseteq (\Sigma^*)^n$ and positive integers $x_1,\ldots,x_m$ such that $\sum_{i=1}^m x_i = n$, an m-ary relation $R_{x_1,\ldots,x_m} \subseteq \Sigma_{x_1}^* \times \cdots \times \Sigma_{x_m}^*$ can be uniquely determined via the mappings $\delta_{x_1},\ldots,\delta_{x_m}$. More precisely, there exists a one-to-one correspondence $\Delta_{x_1,\ldots,x_m}$ between relations R and $R_{x_1,\ldots,x_m}$ that maps each $(w_1,\ldots,w_n) \in R$ to

$$(\delta_{x_1}(w_1,\ldots,w_{x_1}), \delta_{x_2}(w_{x_1+1},\ldots,w_{x_1+x_2}),\ldots,\delta_{x_m}(w_{x_1+\cdots+x_{m-1}+1},\ldots,w_n)) \in R_{x_1,\ldots,x_m}.$$

For example, a ternary relation $R = \{(a,\varepsilon,ab)\}$ over $(\Sigma^*)^3$ uniquely determines a binary relation $R_{1,2} = \{(a,(\perp,a)(\perp,b))\}$ over $\Sigma_1^* \times \Sigma_2^*$ through the correspondence $\Delta_{1,2}$. For the sake of readability, if the integers $x_1,\ldots,x_m$ have a constant subsequence of length k, i.e., $x_i = x_{i+1} = \cdots = x_{i+k-1}$ for some i, we write the relation as $R_{x_1,\ldots,x_{i-1},x_i^k,x_{i+k},\ldots,x_m}$.

In the following, we shall use R_k to denote the binary relation $R_{k,n-k}$ induced by R. It turns out that being able to check monadic decomposability for binary relations is sufficient to check monadic decomposability for general n-ary relations.

▶ **Lemma 13.** *Let R be an n-ary regular relation and let $R_1,\ldots,R_{n-1}$ be the induced binary relations. Then R is monadic decomposable iff $R_1,\ldots,R_{n-1}$ are monadic decomposable.*

Proof. Define $\delta_i(S) = \{\delta_i(s_1,\ldots,s_i) \mid (s_1,\ldots,s_i) \in S\}$. The only-if part of the lemma is immediate, since $R = \bigcup_i X_{i,1} \times \cdots \times X_{i,n}$ implies that $R_k = \bigcup_i \delta_k(X_{i,1} \times \cdots \times X_{i,k}) \times \delta_{n-k}(X_{i,k+1} \times \cdots \times X_{i,n})$ for $1 \leq k \leq n-1$, namely, $R_1,\ldots,R_{n-1}$ are monadic decomposable.

To see the other direction, we say that an n-ary relation R is *k-decomposable* if the induced k-ary relation $R_{1^{k-1},n-k+1}$ of R is monadic decomposable. Now it suffices to show that R is n-decomposable since $R = R_{1^n}$. We shall prove this by induction on $k \in \{2,\ldots,n\}$. Note that R is 2-decomposable by the assumption that R_1 is monadic decomposable. For $2 \leq k \leq n-1$, suppose that $R_k = \bigcup_j A_j \times B_j$ and R is k-decomposable, say $R_{1^{k-1},n-k+1} = \bigcup_i X_{i,1} \times \cdots \times X_{i,k-1} \times Y_i$. Then R is $(k+1)$-decomposable as we have

$$R_{1^k,n-k} = \bigcup_i \bigcup_j X_{i,1} \times \cdots \times X_{i,k-1} \times A_{i,j} \times B_j,$$

where $A_{i,j} = \{x \in \Sigma^* \mid \exists x_1 \in X_{i,1} \cdots \exists x_{k-1} \in X_{i,k-1}.\ \delta_k(x_1,\ldots,x_{k-1},x) \in A_j\}$, i.e., $A_{i,j}$ is the projection of $\delta_k^{-1}(A_j) \cap (X_{i,1} \times \cdots \times X_{i,k-1} \times \Sigma^*)$ on the k-th component. Note that $\delta_k^{-1}(A_j)$ is regular since A_j and $\{(x_1,\ldots,x_k,y) \mid \delta_k(x_1,\ldots,x_k) = y\}$ are regular (cf. [8]). Hence $A_{i,j}$ is also regular. The claim that R is n-decomposable then follows by induction. ◀

Proof (sketch) of Lemma 11. To prove the lemma, we show that if R is regular, then so are the induced relations $R_1, \ldots, R_{n-1}$. Moreover, given the automaton of R, one can construct the automaton for each R_i in logarithmic space from R. We then check if each R_i is monadic decomposable for $i = 1, \ldots, n-1$. From Lemma 10 the latter is in NLOGSPACE (resp. PSPACE), and thus the whole procedure is in NLOGSPACE (resp. PSPACE) if R is given by a DFA (resp. an NFA). ◀

6 Concluding Remarks

Monadic decomposability for rational relations (and subclasses thereof) is a classical problem in automata theory that dates back to the late 1960s (the work of Stearns [33] and Fischer and Rosenberg [19]). While the general problem is undecidable, the subcase of regular relations (i.e. those recognized by synchronized multi-tape automata) provides a good balance between decidability [25, 12] and expressiveness. The complexity of this subcase remained open for over a decade (exponential-time upper bound for the binary case [27, 28], double exponential-time upper bound in the general case [12], and no specific lower bounds). This paper closes this question by providing the precise complexity for the problem: NLOGSPACE (resp. PSPACE) for DFA (resp. NFA) representations.

Some perspectives from formal verification and future work. Researchers from the area of formal verification have increasingly understood the importance of the monadic decompositions techniques, e.g., see [38]. Directly pertinent to monadic decomposability of regular relations is the line of work of constraint solving over strings, wherein increasingly more complex string operations are needed and thus added to solvers [36, 3, 26, 1, 13, 2, 14]. As an example, let us take a look at the recent work of Chen *et al.* [14], which spells out a string constraint language with semantic conditions for decidability that directly use the notion of monadic decomposability of relations over strings. Loosely speaking, a constraint is simply a sequence of program statements, each being either an assignment or a conditional:

$$S ::= \quad y := f(x_1, \ldots, x_r) \mid \mathbf{assert}(g(x_1, \ldots, x_r)) \mid S; S$$

where $f : (\Sigma^*)^r \to \Sigma^*$ is a partial string function and $g \subseteq (\Sigma^*)^r$ is a string relation. The meaning of a constraint is what one would expect in a program written in a standard imperative programming language, which should support assignments and assertions. Note that loops are not allowed in the language since their target application is symbolic executions (e.g. see [11]). They provided two semantic conditions for ensuring decidability, one of which requires that each conditional g is effectively monadic decomposable. There is evidence (e.g. [21, 14]) that some form of length reasoning in g is indeed required for many applications of symbolic executions of string-manipulating programs, but much of the length constraints could be (not yet fully automatically) translated to regular constraints. A potential application for our results is therefore to provide support for complex string relations for g in the form of regular relations, which permit a rather expressive class of conditionals (e.g. some form of length reasoning, etc.). Despite this, this application also highlights what is currently missing in the entire literature of monadic decomposability of rational relations: a study of the problem of *outputting* the monadic decompositions of the relations, if monadic decomposable. (In fact, this is also true of other logical theories before the recent work of Veanes *et al.* [38].) What is the complexity of this problem with various representations of recognizable relations (e.g. finite unions of products, boolean combinations of regular constraints, etc.)? Although our results provide *a first step* towards solving this function problem, we strongly believe this to be a highly challenging open problem in its own right that deserves more attention.

References

1 Parosh Aziz Abdulla, Mohamed Faouzi Atig, Yu-Fang Chen, Bui Phi Diep, Lukás Holík, Ahmed Rezine, and Philipp Rümmer. Flatten and conquer: a framework for efficient analysis of string constraints. In *Proceedings of PLDI 2017*, pages 602–617. ACM, 2017. doi:10.1145/3062341.3062384.

2 Parosh Aziz Abdulla, Mohamed Faouzi Atig, Yu-Fang Chen, Bui Phi Diep, Lukás Holík, Ahmed Rezine, and Philipp Rümmer. TRAU: SMT solver for string constraints. In *Formal Methods in Computer Aided Design, FMCAD 2018*, 2018.

3 Parosh Aziz Abdulla, Mohamed Faouzi Atig, Yu-Fang Chen, Lukás Holík, Ahmed Rezine, Philipp Rümmer, and Jari Stenman. String Constraints for Verification. In *Proceedings of CAV 2014*, volume 8559 of *LNCS*, pages 150–166. Springer, 2014. doi:10.1007/978-3-319-08867-9_10.

4 James Bailey, Guozhu Dong, and Anthony Widjaja To. Logical queries over views: Decidability and expressiveness. *ACM Trans. Comput. Log.*, 11(2):8:1–8:35, 2010. doi:10.1145/1656242.1656243.

5 Pablo Barceló, Chih-Duo Hong, Xuan-Bach Le, Anthony W. Lin, and Reino Niskanen. Monadic Decomposability of Regular Relations. *CoRR*, abs/1903.00728, 2019. arXiv:1903.00728.

6 Michael Benedikt, Leonid Libkin, Thomas Schwentick, and Luc Segoufin. Definable relations and first-order query languages over strings. *J. ACM*, 50(5):694–751, 2003. doi:10.1145/876638.876642.

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

8 Achim Blumensath. *Automatic Structures*. PhD thesis, RWTH Aachen, 1999.

9 George S. Boolos, John P. Burgess, and Richard C. Jeffrey. *Computability and Logic*. Cambridge University Press, fifth edition, 2007.

10 Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Springer, 1997.

11 Cristian Cadar and Koushik Sen. Symbolic Execution for Software Testing: Three Decades Later. *Commun. ACM*, 56(2):82–90, 2013. doi:10.1145/2408776.2408795.

12 Olivier Carton, Christian Choffrut, and Serge Grigorieff. Decision problems among the main subfamilies of rational relations. *RAIRO – Theoretical Informatics and Applications*, 40(2):255–275, 2006. doi:10.1051/ita:2006005.

13 Taolue Chen, Yan Chen, Matthew Hague, Anthony W. Lin, and Zhilin Wu. What is decidable about string constraints with the ReplaceAll function. *PACMPL*, 2(POPL):3:1–3:29, 2018. doi:10.1145/3158091.

14 Taolue Chen, Matthew Hague, Anthony W. Lin, Philipp Rümmer, and Zhilin Wu. Decision procedures for path feasibility of string-manipulating programs with complex operations. *PACMPL*, 3(POPL):49:1–49:30, 2019. doi:10.1145/3290362.

15 Christian Choffrut. Relations over words and logic: A chronology. *Bull. of the EATCS*, 89:159–163, 2006.

16 Thomas Colcombet and Christof Löding. Transforming structures by set interpretations. *Logical Methods in Computer Science*, 3(2), 2007. doi:10.2168/LMCS-3(2:4)2007.

17 Patrick Cousot and Radhia Cousot. Systematic Design of Program Analysis Frameworks. In *Proceedings of POPL 1979*, pages 269–282, 1979. doi:10.1145/567752.567778.

18 Calvin C. Elgot and Jorge E. Mezei. On Relations Defined by Generalized Finite Automata. *IBM J. Res. Dev.*, 9(1):47–68, 1965. doi:10.1147/rd.91.0047.

19 Patrick C. Fischer and Arnold L. Rosenberg. Multitape One-Way Nonwriting Automata. *J. Comput. Syst. Sci.*, 2(1):88–101, 1968. doi:10.1016/S0022-0000(68)80006-6.

20 Christiane Frougny and Jacques Sakarovitch. Synchronized Rational Relations of Finite and Infinite Words. *Theor. Comput. Sci.*, 108(1):45–82, 1993. doi:10.1016/0304-3975(93)90230-Q.

21 Vijay Ganesh, Mia Minnes, Armando Solar-Lezama, and Martin C. Rinard. Word Equations with Length Constraints: What's Decidable? In *Proceedings of HVC 2012*, pages 209–226. Springer, 2012. doi:10.1007/978-3-642-39611-3_21.

22 Bernard R. Hodgson. Decidabilite par automate fini. *Ann. Sc. Math. Quebec*, 7:39–57, 1983.

23 Jean-Louis Imbert. Redundancy, Variable Elimination and Linear Disequations. In *Proceedings of ILPS 1994*, pages 139–153, 1994.

24 Gabriel Kuper, Leonid Libkin, and Jan Paredaens. *Constraint Databases*. Springer Publishing Company, Incorporated, first edition, 2010.

25 Leonid Libkin. Variable Independence, Quantifier Elimination, and Constraint Representations. In *Proceedings of ICALP 2000*, volume 1853 of *LNCS*, pages 260–271. Springer, 2000. doi:10.1007/3-540-45022-X_23.

26 Anthony W. Lin and Pablo Barceló. String Solving with Word Equations and Transducers: Towards a Logic for Analysing Mutation XSS. In *Proceedings POPL 2016*, pages 123–136. ACM, 2016. doi:10.1145/2837614.2837641.

27 Christof Löding and Christopher Spinrath. Decision Problems for Subclasses of Rational Relations over Finite and Infinite Words. In *Proceedings of FCT 2017*, volume 10472 of *LNCS*, pages 341–354. Springer, 2017. doi:10.1007/978-3-662-55751-8_27.

28 Christof Löding and Christopher Spinrath. Decision Problems for Subclasses of Rational Relations over Finite and Infinite Words. *Discrete Mathematics & Theoretical Computer Science*, 21(3), 2019. URL: https://dmtcs.episciences.org/5141.

29 Albert R. Meyer and Larry J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *13th Annual Symposium on Switching and Automata Theory (SWAT 1972)*, pages 125–129. IEEE, 1972. doi:10.1109/swat.1972.29.

30 M. Nivat. Transduction des langages de Chomsky. *Ann. Inst. Fourier*, 18:339–455, 1968.

31 Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

32 Michael Sipser. *Introduction to the Theory of Computation*. Thomson Course Technology Boston, second edition, 2006.

33 Richard Edwin Stearns. A Regularity Test for Pushdown Machines. *Information and Control*, 11(3):323–340, 1967. doi:10.1016/S0019-9958(67)90591-8.

34 A. W. To. *Model Checking Infinite-State Systems: Generic and Specific Approaches*. PhD thesis, LFCS, School of Informatics, University of Edinburgh, 2010.

35 A. W. To and Leonid Libkin. Recurrent Reachability Analysis in Regular Model Checking. In *LPAR*, pages 198–213, 2008. doi:10.1007/978-3-540-89439-1_15.

36 Minh-Thai Trinh, Duc-Hiep Chu, and Joxan Jaffar. S3: A symbolic string solver for vulnerability detection in web applications. In *Proceedings of CCS 2014*, pages 1232–1243. ACM, 2014. doi:10.1145/2660267.2660372.

37 Leslie G. Valiant. Regularity and Related Problems for Deterministic Pushdown Automata. *Journal of the ACM*, 22(1):1–10, 1975. doi:10.1145/321864.321865.

38 Margus Veanes, Nikolaj Bjørner, Lev Nachmanson, and Sergey Bereg. Monadic Decomposition. *Journal of the ACM*, 64(2):1–28, 2017. doi:10.1145/3040488.

Boundedness of Conjunctive Regular Path Queries

Pablo Barceló
Department of Computer Science, University of Chile, Santiago, Chile
IMFD, Santiago, Chile
pbarcelo@dcc.uchile.cl

Diego Figueira
CNRS & LaBRI, Talence, France
diego.figueira@labri.fr

Miguel Romero
Department of Computer Science, University of Oxford, Oxford, UK
miguel.romero@cs.ox.ac.uk

Abstract

We study the boundedness problem for unions of conjunctive regular path queries with inverses (UC2RPQs). This is the problem of, given a UC2RPQ, checking whether it is equivalent to a union of conjunctive queries (UCQ). We show the problem to be ExpSpace-complete, thus coinciding with the complexity of containment for UC2RPQs. As a corollary, when a UC2RPQ is bounded, it is equivalent to a UCQ of at most triple-exponential size, and in fact we show that this bound is optimal. We also study better behaved classes of UC2RPQs, namely *acyclic UC2RPQs of bounded thickness*, and *strongly connected UCRPQs*, whose boundedness problem is, respectively, PSpace-complete and Π_2^P-complete. Most upper bounds exploit results on limitedness for distance automata, in particular extending the model with alternation and two-wayness, which may be of independent interest.

2012 ACM Subject Classification Theory of computation → Database query languages (principles); Theory of computation → Quantitative automata

Keywords and phrases regular path queries, boundedness, limitedness, distance automata

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.104

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version A full version of this paper can be found at `https://arxiv.org/abs/1904.00850` and `https://hal.archives-ouvertes.fr/hal-02056388`.

Funding Barceló is funded by the Millennium Inst. for Foundational Research on Data and Fondecyt 1170109, and Figueira by ANR project DELTA (grant ANR-16-CE40-0007) and ANR project QUID (grant ANR-18-CE40-0031). This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 714532). The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein.

Acknowledgements We are grateful to Thomas Colcombet for helpful discussions and valuable ideas in relation to the results of Section 5.

1 Introduction

Boundedness is an important property of formulas in logics with fixed-point features. At the intuitive level, a formula φ in any such logic is bounded if its *fixed-point depth*, *i.e.*, the number of iterations that are needed to evaluate φ on a structure $\mathbf{A}$, is fixed (and thus it is independent of $\mathbf{A}$). In databases and knowledge representation, boundedness is regarded as an interesting theoretical phenomenon with relevant practical implications [25, 8]. In

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 104; pp. 104:1–104:15


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

fact, while several applications in these areas require the use of recursive features, actual real-world systems are either not designed or not optimized to cope with the computational demands that such features impose. Bounded formulas, in turn, can be reformulated in non-recursive logics, such as FO, or even as a *union of conjunctive queries* (UCQ) when φ itself is positive. UCQs form the core of most systems for data management and ontological query answering, and, in addition, are the focus of advanced optimization techniques. It has also been experimentally verified in some contexts that recursive features encountered in practice are often used in a somewhat "harmless" way, and that many of such queries are in fact bounded [23]. Thus, checking if a recursive formula φ is bounded, and building an equivalent non-recursive formula φ' when the latter holds, are important optimization tasks.

The study of boundedness for *Datalog* programs, *i.e.*, the least fixed-point extension of the class of UCQs, received a lot of attention during the late 80s and early 90s. Two seminal results established that checking boundedness is undecidable in general for Datalog [22], but becomes decidable for *monadic* Datalog, *i.e.*, those programs in which each intensional predicate is monadic [19]. The past few years have seen a resurgence of interest in boundedness problems. This is due, in part, to the development of the theory of *cost automata* over trees (both finite and infinite) in a series of landmark results, in particular relating to its *limitedness* problem. In a few words, cost automata are generalizations of finite automata associating a *cost* from $\mathbb{N} \cup \{\infty\}$ to every input tree (instead of simply accepting or rejecting). The limitedness problem asks, given a cost automata, whether there is a uniform bound on the cost over all (accepting) input trees. Some deep results establish that checking limitedness is decidable for well-behaved classes of cost automata over trees [18, 35, 36, 7]. Remarkably, for several logics of interest the boundedness problem can be reduced to the limitedness for cost automata in such well-behaved classes. Those reductions have enabled powerful decidability results for the boundedness problem. As an example, it has been shown in this way that boundedness is decidable for monadic second-order logic (MSO) over structures of bounded treewidth [11], which corresponds to an extension of Courcelle's Theorem, and also for the *guarded negation* fragment of least fixed-point logic (LFP), even in the presence of unguarded parameters [6]. Cost automata have also been used to study the complexity of boundedness for guarded Datalog programs [7, 3].

Graph databases is a prominent area of study within database theory, in which the use of recursive queries is crucial [2, 1]. A graph database is a finite edge-labeled directed graph. The most basic navigational querying mechanism for graph databases corresponds to the class of *regular path queries* (RPQs), which check whether two nodes of the graph are connected by a path whose label belongs to a given regular language. RPQs are often extended with the ability to traverse edges in both directions, giving rise to the class of *two-way* RPQs, or 2RPQs [15]. The core of the most popular recursive query languages for graph databases is defined by *conjunctive* 2RPQs, or C2RPQs, which are the closure of 2RPQs under conjunction and existential quantifications [14]. We also consider *unions* of C2RPQs, or UC2RPQs. It can be shown that a UC2RPQ is bounded iff it is equivalent to some UCQ. In spite of the inherent recursive nature of UC2RPQs, their boundedness problem has not been studied in depth. Here we develop such a study by showing the following:

- The boundedness problem for UC2RPQs is ExpSpace-complete. The lower bound holds even for CRPQs. This implies that boundedness is not more difficult than *containment* for UC2RPQs, which was shown to be ExpSpace-complete in [14].
- From our upper bound construction it follows that if a UC2RPQ is bounded, then it is equivalent to a UCQ of triple exponential size. We show that this bound is optimal.
- Finally, we obtain better complexity bounds for some subclasses of UC2RPQs; namely, for acyclic UC2RPQs of *bounded thickness*, in which case boundedness becomes PSpace-complete, and for *strongly connected* UCRPQs, for which it is Π_2^P-complete.

It is important to stress that UC2RPQs can be easily translated into guarded LFP with unguarded parameters, for which boundedness was shown to be decidable by applying sophisticated cost automata techniques as mentioned above. However, the complexity of the boundedness problem for such a logic is currently not well-understood – and it is at least 2ExpTime-hard [7] – and hence this translation does not yield, in principle, optimal complexity bounds for our problem. To study the boundedness for UC2RPQs, we develop instead techniques especially tailored to UC2RPQs. In fact, since the recursive structure of UC2RPQs is quite tame, their boundedness problem can be translated into the limitedness problem for a much simpler automata model than cost automata on trees; namely, *distance* automata on finite words. Distance automata are nothing more than usual NFAs with two sorts of transitions: costly and non-costly. Such an automaton is limited if there is an integer $k \geq 1$ such that every word accepted by the NFA has an accepting run with at most k costly transitions. A beautiful result in automata theory established the decidability of the limitedness problem for distance automata [24], which is actually in PSpace [29]. While being a difficult result, by now we have quite transparent proofs of this fact (see, *e.g.*, [26]). We exploit our translation to obtain tight complexity upper bounds for boundedness of UC2RPQs. Some of the proofs in the paper require extending the study of limitedness to *alternating* and *two-way* distance automata, while preserving the PSpace bound for the limitedness problem. We believe these results to be of independent interest.

Organization of the paper. Section 2 contains preliminaries. We present characterizations of boundedness for UC2RPQs in Section 3 and an application of those to pinpoint the complexity of Boundedness for RPQs in Section 4. Distance automata and results about them are given in Section 5. We analyze the complexity of Boundedness for general UC2RPQs in Section 6 and present some classes of UC2RPQs with better complexity of Boundedness in Section 7. We finish with a discussion in Section 8.

2 Preliminaries

We assume familiarity with *non-deterministic finite automata* (NFA), *two-way* NFA (2NFA), and *alternating finite automata* (AFA) over finite words. We often blur the distinction between an NFA $\mathcal{A}$ and the language $L(\mathcal{A})$ it defines; similarly for regular expressions.

Graph databases and conjunctive regular path queries. A *graph database* over a finite alphabet $\mathbb{A}$ is a finite edge-labelled graph $G = (V, E)$ over $\mathbb{A}$, where V is a finite set of vertices and $E \subseteq V \times \mathbb{A} \times V$ is the set of labelled edges. We write $u \xrightarrow{a} v$ to denote an edge $(u, a, v) \in E$. We define the alphabet $\mathbb{A}^{\pm} := \mathbb{A} \dot{\cup} \mathbb{A}^{-1}$ that extends $\mathbb{A}$ with the set $\mathbb{A}^{-1} := \{a^{-1} \mid a \in \mathbb{A}\}$ of "inverses" of symbols in $\mathbb{A}$. An *oriented path* from u to v in a graph database $G = (V, E)$ over alphabet $\mathbb{A}$ is a pair $\pi = (\sigma, \ell)$ where σ and ℓ are (possibly empty) sequences $\sigma = (v_0, a_1, v_1), (v_1, a_2, v_2), \ldots, (v_{k-1}, a_k, v_k) \in V \times \mathbb{A} \times V$, and $\ell = \ell_1, \ldots, \ell_k \in \{-1, 1\}$, for $k \geq 0$, such that $u = v_0$, $v = v_k$, and for each $1 \leq i \leq k$, we have that $\ell_i = 1$ implies $(v_{i-1}, a_i, v_i) \in E$; and $\ell_i = -1$ implies $(v_i, a_i, v_{i-1}) \in E$. The *label* of π is the word $b_1 \ldots b_k \in (\mathbb{A}^{\pm})^*$, where $b_i = a_i$ if $\ell_i = 1$; otherwise $b_i = a_i^{-1}$. When $k = 0$ the label of π is the empty word ε. If $\ell_i = 1$ for every $1 \leq i \leq k$, we say that π is a *directed path*. Note that in this case, the label of π belongs to $\mathbb{A}^*$.

A *regular path query* (RPQ) over $\mathbb{A}$ is a regular language $L \subseteq \mathbb{A}^*$, which we assume to be given as an NFA. The evaluation of L on a graph database $G = (V, E)$ over $\mathbb{A}$, written $L(G)$, is the set of pairs $(u, v) \in V \times V$ such that there is a directed path from u to v in G

whose label belongs to L. 2RPQs extend RPQs with the ability to traverse edges in both directions. Formally, a 2RPQ L over $\mathbb{A}$ is simply an RPQ over $\mathbb{A}^{\pm}$. The evaluation $L(G)$ of L over a graph database $G = (V, E)$ over $\mathbb{A}$ is the set of pairs $(u, v) \in V \times V$ such that there is an oriented path from u to v in G whose label belongs to L.

Conjunctive 2RPQs (C2RPQs) are obtained by taking the closure of 2RPQs under conjunction and existential quantification, *i.e.*, a C2RPQ over $\mathbb{A}$ is an expression $\gamma := \exists \bar{z} \left((x_1 \xrightarrow{L_1} y_1) \wedge \cdots \wedge (x_m \xrightarrow{L_m} y_m) \right)$, where each L_i is a 2RPQ over $\mathbb{A}$ and $\bar{z}$ is a tuple of variables among those in $\{x_1, y_1, \ldots, x_m, y_m\}$. We say that γ is a CRPQ if each L_i is an RPQ. If $\bar{x} = (x^1, \ldots, x^n)$ is the tuple of *free variables* of γ, *i.e.*, those that are not existentially quantified in $\bar{z}$, then the evaluation $\gamma(G)$ of the C2RPQ γ over a graph database G is the set of all tuples $h(\bar{x}) = (h(x^1) \ldots, h(x^n))$, where h ranges over all mappings $h : \{x_1, y_1, \ldots, x_m, y_m\} \to V$ such that $(h(x_i), h(y_i)) \in L_i(G)$ for each $1 \leq i \leq m$.

A *union* of C2RPQs (UC2RPQ) is an expression of the form $\Gamma := \bigvee_{1 \leq i \leq n} \gamma_i$, where the γ_i's are C2RPQ, all of which have exactly the same free variables. The evaluation $\Gamma(G)$ of Γ over a graph database G is $\bigcup_{1 \leq i \leq n} \gamma_i(G)$. We often write $\Gamma(\bar{x})$ to denote that $\bar{x}$ is the tuple of free variables of Γ. A UC2RPQ Γ is *Boolean* if it contains no free variables.

Given UC2RPQs Γ and Γ', we write $\Gamma \subseteq \Gamma'$ if $\Gamma(G) \subseteq \Gamma'(G)$ for each graph database G. Hence, Γ and Γ' are *equivalent* if $\Gamma \subseteq \Gamma'$ and $\Gamma' \subseteq \Gamma$, *i.e.*, $\Gamma(G) = \Gamma'(G)$ for every G.

Boundedness of UC2RPQs. CRPQs, and even UC2RPQs, can easily be expressed in *Datalog*, the least fixed-point extension of the class of *union of conjunctive queries* (UCQs). Hence, we can directly define the boundedness of a UC2RPQ in terms of the boundedness of its equivalent Datalog program, which is a well-studied problem [25]. The latter, however, coincides with being equivalent to some UCQ [31]. In the setting of graph databases, a *conjunctive query* (CQ) over $\mathbb{A}$ is simply a CRPQ over $\mathbb{A}$ of the form $\exists \bar{z} \bigwedge_{1 \leq i \leq m} (x_i \xrightarrow{a_i} y_i)$ where the a_is range over $\mathbb{A} \cup \{\varepsilon\}$. Notice that atoms of the form $x \xrightarrow{\varepsilon} y$ correspond to *equality atoms* $x = y$. Analogously, one can define unions of CQs (UCQs). Note that, modulo equality atoms, a CQ over $\mathbb{A}$ can be seen as a graph database over $\mathbb{A}$. Hence, we shall slightly abuse notation and, in the setting of CQs, use notions defined for graph databases (such as oriented paths).

A UC2RPQ Γ is *bounded* if it is equivalent to some UCQ Φ. In this article we study the complexity of the problem Boundedness, which takes as input a UC2RPQ Γ and asks whether Γ is bounded.

▶ **Example 1.** Consider the Boolean UCRPQ $\Gamma = \gamma_1 \vee \gamma_2$ over the alphabet $\mathbb{A} = \{a, b, c, d\}$ such that $\gamma_1 = \exists x, y \, (x \xrightarrow{L_b} y \wedge x \xrightarrow{L_{b,d}} y)$ and $\gamma_2 = \exists x, y \, (x \xrightarrow{L_d} y \wedge x \xrightarrow{L_{b,d}} y)$, where $L_b := a^+b^+c$, $L_d := ad^+c^+$, and $L_{b,d} := a^+(b + d)c^+$. For $e \in \mathbb{A}$, recall that e^+ denotes the language $e(e^*)$. As we shall explain in Example 4, we have that γ_1 and γ_2 are unbounded. However, Γ is bounded, and in particular, it is equivalent to the UCQ $\Phi = \varphi_1 \vee \varphi_2$, where φ_1 and φ_2 correspond to $\exists x, y \, (x \xrightarrow{abc} y)$ and $\exists x, y \, (x \xrightarrow{adc} y)$, respectively. ◀

3 Characterizations of Boundedness for UC2RPQs

In this section we provide two simple characterizations of when a UC2RPQ is bounded that will be useful to analyze the complexity of Boundedness. Let $\varphi(\bar{x})$ and $\varphi'(\bar{x})$ be CQs over $\mathbb{A}$ with variable sets $\mathcal{V}$ and $\mathcal{V}'$, respectively. Let $=_\varphi$ and $=_{\varphi'}$ be the binary relations induced on $\mathcal{V}$ and $\mathcal{V}'$ by the equality atoms of φ and φ', respectively, and $=_\varphi^*$ and $=_{\varphi'}^*$ be their reflexive-transitive closure. A *homomorphism* from φ to φ' is a mapping $h : \mathcal{V} \to \mathcal{V}'$

such that: (i) $x =^*_\varphi y$ implies $h(x) =^*_{\varphi'} h(y)$; (ii) $h(\bar{x}) = \bar{x}$; and (iii) for each atom $x \xrightarrow{a} y$ in φ with $a \in \mathbb{A}$, there is an atom $x' \xrightarrow{a} y'$ in φ' such that $h(x) =^*_{\varphi'} x'$ and $h(y) =^*_{\varphi'} y'$. We write $\varphi \to \varphi'$ if such a homomorphism exists. It is known that $\varphi \to \varphi'$ iff $\varphi' \subseteq \varphi$ [16].

An *expansion* of a C2RPQ $\gamma(\bar{x})$ over $\mathbb{A}$ is a CQ $\lambda(\bar{x})$ over $\mathbb{A}$ with minimal number of variables and atoms such that (i) λ contains each variable of γ, (ii) for each atom $A = x \xrightarrow{L} y$ of γ, there is an oriented path π_A in λ from x to y with label $w_A \in L$ whose intermediate variables (*i.e.*, those not in $\{x, y\}$) are distinct from one another, and (iii) intermediate variables of different oriented paths π_A and $\pi_{A'}$ are disjoint. Note that the free variables of λ and γ coincide. Intuitively, the expansion λ is obtained from γ by choosing for each atom $A = x \xrightarrow{L} y$ a word $w_A \in L$, and "expanding" $x \xrightarrow{L} y$ into the "fresh oriented path" π_A from x to y with label w_A. When $w_A = \varepsilon$ then λ contains the equality atom $x = y$. An expansion of a UC2RPQ Γ is an expansion of some C2RPQ in Γ. Observe that a (U)C2RPQ is always equivalent to the (potentially infinite) UCQ given by its set of expansions. Even more, it is equivalent to the UCQ defined by its *minimal* expansions, as introduced below.

If λ is an expansion of a UC2RPQ Γ, we define the *size* of λ, denoted by $\|\lambda\|$, to be the number of (non-equality) atoms in λ. We say that λ is *minimal*, if there is no expansion λ' such that $\lambda' \to \lambda$ and $\|\lambda'\| < \|\lambda\|$. Intuitively, an expansion is minimal if its answers cannot be covered by a smaller expansion. We can then establish the following.

▶ **Lemma 2.** *Every UC2RPQ Γ is equivalent to the (potentially infinite) UCQ given by its set of minimal expansions.*

We can now provide our basic characterizations of boundedness.

▶ **Proposition 3.** *The following conditions are equivalent for each UC2RPQ Γ.*

1. *Γ is bounded.*
2. *There is $k \geq 1$ such that for every expansion λ of Γ there exists an expansion λ' of Γ with $\|\lambda'\| \leq k$ such that $\lambda \subseteq \lambda'$ (i.e., such that $\lambda' \to \lambda$).*
3. *Γ has finitely many minimal expansions.*

▶ **Example 4.** Consider the Boolean UCRPQ $\Gamma = \gamma_1 \vee \gamma_2$ over $\mathbb{A} = \{a, b, c, d\}$ from Example 1. To see that γ_1 is unbounded (the case of γ_2 is similar) we can apply Proposition 3. Indeed, the expansions of γ_1 corresponding to $\{\exists x, y\,(x \xrightarrow{ab^n c} y \wedge x \xrightarrow{adc} y) : n \geq 1\}$ are all minimal. On the other hand, Γ is bounded as its minimal expansions correspond to $\exists x, y\,(x \xrightarrow{abc} y \wedge x \xrightarrow{abc} y)$ and $\exists x, y\,(x \xrightarrow{adc} y \wedge x \xrightarrow{adc} y)$. ◀

4 Boundedness for Existentially Quantified RPQs

As a first application of Proposition 3, we study Boundedness for CRPQs consisting of a single RPQ; that is, RPQs or existentially quantified RPQs. Let v, w be words over $\mathbb{A}$. Recall that a word v is a *prefix* [resp. *suffix* and *factor*] of w if $w \in v \cdot \mathbb{A}^*$ [resp. $w \in \mathbb{A}^* \cdot v$ and $w \in \mathbb{A}^* \cdot v \cdot \mathbb{A}^*$]. If in addition we have $v \neq w$, then we say that v is a *proper* prefix [resp. suffix and factor] of w. For a language $L \subseteq \mathbb{A}^*$, we define its *prefix-free* sub-language L_{pf} to be the set of words $w \in L$ such that w has no proper prefix in L. Similarly, we define L_{sf} and L_{ff} with respect to the suffix and factor relation. We have the following:

▶ **Proposition 5.** *The following statements hold.*

1. *An RPQ L is bounded iff L is finite.*
2. *A CRPQ $\exists y(x \xrightarrow{L} y)$ [resp. $\exists x(x \xrightarrow{L} y)$] with $x \neq y$ is bounded iff L_{pf} [resp. L_{sf}] is finite.*
3. *A Boolean CRPQ $\exists x, y(x \xrightarrow{L} y)$ with $x \neq y$ is bounded iff L_{ff} is finite.*

▶ **Theorem 6.** *The problem of, given an NFA accepting the language L, checking whether L_{pf} is finite is* PSPACE*-complete. The same holds if we replace L_{pf} by L_{sf} or L_{ff}.*

Proof. We focus on upper bounds, the lower bounds are in the appendix. Given an NFA $\mathcal{A}$ accepting the language L, we can construct an NFA $\mathcal{B}$ of polynomial size in $\mathcal{A}$ that accepts precisely those words that have a proper prefix in L. By complementing and intersecting with $\mathcal{A}$, we obtain an NFA $\mathcal{B}'$ of exponential size in $\mathcal{A}$ that accepts the language L_{pf}. Hence, we only need to check whether the language accepted by $\mathcal{B}'$ is finite, which can be done *on-the-fly* in NL w.r.t. $\mathcal{B}'$, and hence in PSPACE. The other two cases are analogous. ◀

By applying Theorem 6 and Proposition 5, we can now pinpoint the complexity of BOUNDEDNESS for CRPQs with a single RPQ.

▶ **Corollary 7.** *The following statements hold.*

1. BOUNDEDNESS *for RPQs is* NL*-complete.*
2. BOUNDEDNESS *for CRPQs of the form $\exists y(x \xrightarrow{L} y)$, with $x \neq y$, is* PSPACE*-complete. The same holds for CRPQs $\exists x(x \xrightarrow{L} y)$ and Boolean CRPQs $\exists x, y(x \xrightarrow{L} y)$, where $x \neq y$.*

It is not clear, though, how usual automata techniques, as the ones applied in the proof of Theorem 6, can be used to solve BOUNDEDNESS for more complex CRPQs. To solve this problem we develop an approach based on distance automata, as introduced next. Our approach also handles inverses and unions, thus dealing with arbitrary UC2RPQs.

5 Distance Automata

Distance automata [24] (equivalent to weighted automata over the $(\min, +)$-semiring [21], min-automata [12], or $\{\varepsilon, ic\}$-B-automata [17]) are an extension of finite automata which associate to each word in the language a natural number or "cost". They can be represented as non-deterministic finite automata with two sorts of transitions: costly and non-costly. For a given distance automaton, the cost of a run on a word is the number of costly transitions, and the cost of a word $w \in \mathbb{A}^*$ is the minimum cost of an accepting run on w. We will use this automaton model to encode boundedness as the problem of whether there is a uniform bound on the cost of words, known as the *limitedness problem*.

Formally, a *distance automaton* (henceforth *DA*) is a tuple $\mathcal{A} = (\mathbb{A}, Q, q_0, F, \delta)$, where $\mathbb{A}$ is a finite alphabet, Q is a finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of finals states and $\delta \subseteq Q \times \mathbb{A} \times \{0, 1\} \times Q$ is the transition relation. A word $w \in \mathbb{A}^*$ is *accepted* by $\mathcal{A}$ if there is an *accepting run* of $\mathcal{A}$ on w, *i.e.*, a (possibly empty) sequence of transitions $\rho = (p_1, a_1, c_1, r_1) \cdots (p_n, a_n, c_n, r_n) \in \delta^*$ with the usual properties: (1) if $\rho = \varepsilon$ then $q_0 \in F$ and $w = \varepsilon$, (2) $p_1 = q_0$ and $r_n \in F$, (3) for every $1 \leq i < n$ we have $r_i = p_{i+1}$, and (4) $w = a_1 \cdots a_n$. The *cost* of the run ρ is $cost(\rho) = c_1 + \cdots + c_n$ (or 0 if $\rho = \varepsilon$); and the cost $cost_{\mathcal{A}}(w)$ of a word w accepted by $\mathcal{A}$ is the minimum cost of an accepting run of $\mathcal{A}$ on w. For convenience, we assume the cost of words not accepted by $\mathcal{A}$ to be 0.

The *limitedness problem* for DA is defined as follows: given a DA $\mathcal{A}$, determine whether $\sup_{w \in \mathbb{A}^*} cost_{\mathcal{A}}(w) < \infty$. This problem is known to be PSPACE-complete.

▶ **Theorem 8** ([28, 29])**.** *The following statements hold:*

1. *The limitedness problem for DA is* PSPACE*-complete.*
2. *If a DA with n states is limited, then $\sup_{w \in \mathbb{A}^*} cost_{\mathcal{A}}(w) \leq 2^{O(n^3)}$.*

We use two extensions of DA: *alternating* and *two-way*. Two-way DA is defined as for NFA, extending the cost function accordingly. The cost of a word is still the minimum over the cost of all (potentially infinitely many) runs. Alternating DA is defined as usual by having

two sorts of states: universal and existential. Existential states can be seen as computing the minimum among the cost of all possible continuations of the run, and universal states as computing the maximum (or supremum if the automaton is also two-way). As we will see, these extensions preserve the above PSPACE upper bound for the limitedness problem.

Formally, an *alternating two-way DA with epsilon transitions* (A2DA$^\varepsilon$) over $\mathbb{A}$ is a tuple $\mathcal{A} = (\mathbb{A}, Q_\exists, Q_\forall, q_0, F, \delta)$ is an A2DA$^\varepsilon$ if $q_0 \in Q_\exists$, $F \subseteq Q_\exists$ and

$$\delta \subseteq (Q_\exists \cup Q_\forall) \times (\mathbb{A}^\pm \cup \{\varepsilon\}) \times \{end, \overline{end}\} \times \{0,1\} \times (Q_\exists \cup Q_\forall);$$

where *end* indicates that after reading the letter we arrive at the end of the word (*i.e.*, either the leftmost or the rightmost end) and $\overline{end}$ indicates that we do not. When the automaton $\mathcal{A}$ is two-way, it is convenient to think of its head as being *between* the letter positions of the word, so an *end*-flagged transition can be applied only if it moves the head to be right before the first letter of the word, or right after the last one.

For any given word $w \in \mathbb{A}^*$, consider the edge-labelled graph $G_{\mathcal{A},w} = (V, E)$ over δ, where $V = Q \times \{0, \ldots, |w|\}$, with $Q = Q_\exists \cup Q_\forall$, and $E \subseteq V \times \delta \times V$ consists of all edges $(q,i) \xrightarrow{(q,a,e,c,p)} (p,j)$ such that $e = end$ iff $j = 0$ or $j = |w|$ and either (a) $i < |w|$, $a = w[i+1]$, and $j = i+1$; (b) $i > 0$, $a = (w[i])^{-1}$, and $j = i-1$; or (c) $a = \varepsilon$ and $j = i$.

An *accepting run of* $\mathcal{A}$ *on* w *from* $(q,i) \in Q \times \{0, \ldots, |w|\}$ is a finite (possibly empty) edge-labelled directed rooted tree[1] t over δ and a labelling h from the nodes of t to the nodes of $G_{\mathcal{A},w}$, such that if t is empty then $q \in F$, and otherwise h maps the root of t to (q,i), every leaf of t to $F \times \{0, \ldots, |w|\}$, and for every node x of t:

- if (x, α, y) is an (labeled) edge in t for some y, then $(h(x), \alpha, h(y))$ is an edge in $G_{\mathcal{A},w}$;
- if $h(x) \in Q_\forall \times \{0, \ldots, |w|\}$, then for every edge $(h(x), \alpha, c)$ in $G_{\mathcal{A},w}$, there is an edge (x, α, y) in t so that $h(y) = c$;
- if $h(x) \in Q_\exists \times \{0, \ldots, |w|\}$, then x has at most one child.

Each branch of t with label $(q_1, a_1, e_1, c_1, p_1), \ldots, (q_n, a_n, e_n, c_n, p_n)$ has an associated cost of $c_1 + \cdots + c_n$; and the cost associated with t is the maximum among the costs of its branches, or 0 if t is empty. The cost $cost_\mathcal{A}(w, q, i)$ is the minimum cost of an accepting run on w from (q,i), or 0 if none exists; $cost_\mathcal{A}(w)$ is defined as $cost_\mathcal{A}(w, q_0, 0)$.

An A2DA$^\varepsilon$ with $\delta \subseteq Q \times (\mathbb{A} \cup \{\varepsilon\}) \times \{end, \overline{end}\} \times \{0,1\} \times Q$ is an *alternating DA* with ε transitions (ADA$^\varepsilon$). An A2DA$^\varepsilon$ with $Q_\forall = \emptyset$ is a *two-way DA* with ε transitions (2DA$^\varepsilon$). An A2DA with both the aforementioned conditions is (equivalent to) a DA with ε transitions (DA$^\varepsilon$). Notice that in the last two cases, accepting runs can be represented as words from δ^* rather than trees. By A2DA (resp., ADA, 2DA, DA) we denote an A2DA$^\varepsilon$ (resp., ADA$^\varepsilon$, 2DA$^\varepsilon$, DA$^\varepsilon$) with no ε-transitions. Note that DA as just defined is in every sense equivalent to the distance automata model we have defined at the beginning of this section – this is why we overload the same "DA" name.

We first observe that 2DA can be transformed into DA while preserving both the language and limitedness problems by adapting the standard "crossing sequence" construction for translating 2NFA into NFA [34]. This fact will be useful for proving the EXPSPACE upper bound for BOUNDEDNESS of general UC2RPQs in Section 6.

▶ **Proposition 9.** *There is an exponential time procedure which for every 2DA* $\mathcal{A}$ *over* $\mathbb{A}$ *produces a DA* $\mathcal{B}$ *over* $\mathbb{A}$ *such that the languages accepted by* $\mathcal{A}$ *and* $\mathcal{B}$ *are the same, and* $cost_\mathcal{B}(w) \leq cost_\mathcal{A}(w) \leq f(cost_\mathcal{B}(w))$ *for every* $w \in \mathbb{A}^*$*, where* f *is a polynomial function that depends on the number of states of* $\mathcal{A}$*.*

[1] That is, a tree-shaped finite edge-labelled graph over δ with edges directed in the root-to-leaf sense.

Recall that the universality problem for NFAs is known to be PSPACE-complete [27]; and that this bound actually extends to two-way and even alternating automata. We show that, likewise, the limitedness problem remains in PSPACE for A2DA$^\varepsilon$. This result will be useful to show in Section 7 that BOUNDEDNESS for the class of acyclic UC2RPQs of bounded thickness is in PSPACE.

▶ **Theorem 10.** *The limitedness problem for A2DA$^\varepsilon$ is* PSPACE*-complete.*

The novelty of this result is the PSPACE upper bound. In fact, decidability follows from known results, and in particular [7, Theorem 14] claims EXPTIME-membership in the more challenging setup of infinite trees. However, this is obtained via an involved construction spanning through several papers. The proof of Theorem 10, instead, is obtained by the composition of the following reductions:

$$\text{lim. A2DA}^\varepsilon \xrightarrow{(1)} \text{lim. A2DA} \xrightarrow{(2)} \text{lim. 2DA} \xrightarrow{(3)} \text{lim. ADA}^\varepsilon \xrightarrow{(4)} \text{lim. ADA} \xrightarrow{(5)} \text{lim. DA.}$$

Reductions (1), (3) and (4) are in polynomial time, while reductions (2) and (5), which are basically the same, are in exponential time. Specifically, reductions (2) and (5) preserve the statespace but the size of the alphabet grows exponentially in the number of states and linearly in the size of the source alphabet. However, the alphabet and transition set resulting from these reductions can be succinctly described: letters are encoded in polynomial space, and checking for membership in the transition set is polynomial time computable.

In summary, the composition (1)+(2)+(3)+(4)+(5) yields a DA with the following characteristics: (i) it has a polynomial number of states Q; (ii) it runs on an exponential alphabet $\mathbb{A}$ –and every letter is encoded in polynomial space–; and (iii) one can check in polynomial time whether a tuple $t \in Q \times \mathbb{A} \times \{end, \overline{end}\} \times \{0,1\} \times Q$ is in its transition relation. This, coupled with Theorem 8, item (2) (which offers a bound depending only on the number of states), provides a polynomial space algorithm for the limitedness of A2DA$^\varepsilon$: We can non-deterministically check the existence of a word with cost greater than the single exponential bound N using only polynomial space, by guessing one letter at a time and keeping the set of reachable states together with the associated costs, where each cost is encoded in binary using polynomial space if it is smaller than N, or with a "∞" flag otherwise. The algorithm accepts if at least one final state is reached and the costs of all reachable final states are marked ∞. Since NPSPACE =PSPACE (Savitch's Theorem), Theorem 10 follows.

We now provide a brief description of the reductions used in the proof of Theorem 10.

(1) From A2DA$^\varepsilon$ to A2DA. This is a trivial reduction obtained by simulating ε-transitions by reading $a \cdot a^{-1}$ for some $a \in \mathbb{A}$.

(2) From A2DA to 2DA. Given an A2DA $\mathcal{A} = (\mathbb{A}, Q_\forall, Q_\exists, q_0, F, \delta)$, we build a 2DA $\mathcal{B}$ over a larger alphabet $\mathbb{B}$, where we trade alternation for extra alphabet letters. The alphabet $\mathbb{B}$ consists of triples $(f^\rightarrow, a, f^\leftarrow)$, where $a \in \mathbb{A}$ and $f^\rightarrow, f^\leftarrow : Q_\forall \to \delta$. The idea is that $f^\rightarrow, f^\leftarrow$ are "choice functions" for the alternation: whenever we are to the left (resp., right) of a position of the word labelled $(f^\rightarrow, a, f^\leftarrow)$ in state $q \in Q_\forall$, instead of exploring all transitions departing from q and taking the maximum cost over all such runs (this is what alternation does in $\mathcal{A}$), $\mathcal{B}$ chooses to just take the transition $f^\rightarrow(q)$ (resp., $f^\leftarrow(q)$). Note that $\mathbb{B}$ is exponential in the number of states but not in the size of $\mathbb{A}$. In this way, we build a 2DA $\mathcal{B}$ having the same set of states as $\mathcal{A}$ but with a transition function which is essentially deterministic on the states of $Q_\forall$. In the end we obtain that

- for every $w \in \mathbb{B}^*$, $cost_\mathcal{B}(w) \leq cost_\mathcal{A}(w_\mathbb{A})$; and
- for every $w \in \mathbb{A}^*$ there is $\widetilde{w} \in \mathbb{B}^*$ so that $\widetilde{w}_\mathbb{A} = w$ and $cost_\mathcal{A}(w) = cost_\mathcal{B}(\widetilde{w})$,

where $w_\mathbb{A}$ and $\widetilde{w}_\mathbb{A}$ denote the projections onto the alphabet $\mathbb{A}$. This implies that the limitedness problem is preserved.

(3) From 2DA to ADA$^\varepsilon$. We show a polynomial-time translation from 2DA to ADA$^\varepsilon$ which preserves limitedness. In the case of finite automata, there are language-preserving reductions from 2NFA to AFA with a quadratic blowup in the statespace [9, 32]. However, these translations, when applied blindly to reduce from 2DA to ADA$^\varepsilon$, preserve neither the cost semantics nor the limitedness of languages. On the other hand, [10] shows an involved construction that results in a reduction from 2DA to ADA$^\varepsilon$ on *infinite trees*, which preserves limitedness but it is not polynomial in the number of states. We show a translation from 2DA to ADA$^\varepsilon$ which serves our purpose: it preserves limitedness and it is polynomial time computable. The translation is close to the language-preserving reduction from 2NFA to AFA of [32], upgraded to take into account the cost of different alternation branches, somewhat in the same spirit as the *history summaries* from [10].

(4) From ADA$^\varepsilon$ to ADA. This is a straightforward polynomial time reduction which preserves limitedness but – as opposed to (1) – does not preserve the language: we need to add an extra letter to the alphabet in order to make the reduction work in polynomial time.

(5) From ADA to DA. This is exactly the same reduction as (2), noticing that the alphabet will still be single exponential in the original A2DA$^\varepsilon$.

6 Complexity of Boundedness for UC2RPQs

Here we show that Boundedness for UC2RPQs is ExpSpace-complete. We do so by applying distance automata results presented in the previous section on top of the semantic characterizations presented in Section 3. The lower bound applies even for CRPQs. We further show that there is a triple exponential tight bound for the size of the equivalent UCQ of a UC2RPQ (and even CRPQ), whenever this exists. This is summarized in the following theorem. If Γ is a UC2RPQ, we write $\|\Gamma\|$ for the length of an arbitrary reasonable encoding of Γ – in particular, encodings in which regular languages are described through NFA or regular expressions.

▶ **Theorem 11.** *The following statements hold.*

1. Boundedness *for UC2RPQs is* ExpSpace*-complete. The problem remains* ExpSpace*-hard even for Boolean CRPQs.*
2. *If a UC2RPQ Γ is bounded, there is a UCQ Φ that is equivalent to Γ and such that Φ has at most triple-exponentially many CQs, each one of which is at most of double exponential size with respect to $\|\Gamma\|$.*
3. *There is a family $\{\Gamma_n\}_{n\geq 1}$ of Boolean CRPQs such that for each $n \geq 1$ it is the case that: (1) $\|\Gamma_n\| = O(n)$, (2) Γ_n is bounded, and (3) every UCQ that is equivalent to Γ_n has at least triple-exponentially many CQs with respect to n.*

6.1 Upper bounds

Our upper bound proof builds on top of techniques developed by Calvanese et al. [14] for studying the *containment problem for UC2RPQs*: Given UC2RPQs Γ, Γ', is it the case that $\Gamma \subseteq \Gamma'$? It is shown in [14] that from Γ, Γ' it is possible to construct exponentially sized NFAs $\mathcal{A}_{\Gamma,\Gamma'}$ and $\mathcal{A}'_{\Gamma,\Gamma'}$, such that $\Gamma \subseteq \Gamma'$ iff there is a word in $\mathcal{A}_{\Gamma,\Gamma'} \cap \overline{\mathcal{A}'_{\Gamma,\Gamma'}}$. It is a well-known result that the latter is solvable in NL in the combined size of $(\mathcal{A}_{\Gamma,\Gamma'}, \overline{\mathcal{A}'_{\Gamma,\Gamma'}})$, *i.e.*, in ExpSpace. We modify this construction to study the boundedness of a given UC2RPQ Γ. In particular, we construct from Γ in exponential time a DA $\mathcal{D}_\Gamma$ such that Γ is bounded iff $\mathcal{D}_\Gamma$ is limited. The result then follows from Theorem 8, which establishes that limitedness for $\mathcal{D}_\Gamma$ can be solved in polynomial space on the number of its states, and thus in ExpSpace.

▶ **Proposition 12.** *There is a single exponential time procedure that takes as input a UC2RPQ* Γ *and constructs a DA* $\mathcal{D}_\Gamma$ *such that* Γ *is bounded iff* $\mathcal{D}_\Gamma$ *is limited.*

Proof. Similarly as done in [14], the DA $\mathcal{D}_\Gamma$ will run over encodings of expansions of the UC2RPQ Γ, *i.e.*, words over the alphabet $\mathbb{A}_1 := \mathbb{A}^\pm \cup \mathcal{V} \cup \{\$\}$, where $\mathbb{A}$ is the alphabet of Γ, $\mathcal{V}$ is the set of variables of Γ, and $\$$ is a fresh symbol. If $\gamma = \exists \bar{z} \bigwedge_{1 \leq i \leq m} (x_i \xrightarrow{L_i} y_i)$ is a C2RPQ in Γ and λ is the expansion of γ obtained by expanding each $x_i \xrightarrow{L_i} y_i$ into an oriented path π_i from x_i to y_i with label $w_i \in L_i$, then we encode λ as the word

$$w_\lambda = \$x_1 w_1 y_1 \$ x_2 w_2 y_2 \$ \cdots \$ x_m w_m y_m \$ \ \in \ \mathbb{A}_1^*$$

Note how the subword $x_i w_i y_i$ encodes the oriented path π_i. Every position $j \in \{1, \ldots, |w_\lambda|\}$ with $w_\lambda[j] \neq \$$ represents a variable in λ: either x_i or y_i if $w_\lambda[j] = x_i$ or $w_\lambda[j] = y_i$, respectively; or the $(\ell+1)$-th variable in the oriented path π_i if $w_\lambda[j]$ is the ℓ-th symbol in the subword w_i. Hence different positions in w_λ could represent the same variable in λ, *e.g.*, in the encoding $\$xabcy\$$, the $5th$ position containing a "c" and the $6th$ position containing a "y", represent the same variable, namely, the last vertex y of the oriented path. It is then easy to build, in polynomial time, an NFA $\mathcal{A}_1$ over $\mathbb{A}_1$ recognizing the language of all such encodings of expansions of Γ. Our automaton $\mathcal{D}_\Gamma$ is the product of $\mathcal{A}_1$ and the DA $\mathcal{C}_\Gamma$ defined below. In particular, $\mathcal{D}_\Gamma$ is limited iff $\mathcal{C}_\Gamma$ is limited over words of the form w_λ, for λ an expansion of Γ.

Fix a disjunct γ of Γ. As in [14], we consider words over the alphabet $\mathbb{A}_2 := \mathbb{A}_1 \times (2^\mathcal{V} \cup \{\#\})$ of the form $(\ell_1, \alpha_1) \cdots (\ell_n, \alpha_n)$, such that $w_\lambda = \ell_1 \cdots \ell_n$, for some expansion λ of Γ, and the α_i's are *valid* γ*-annotations*, *i.e.*, (1) $\alpha_i = \#$ if $\ell_i = \$$, (2) $\alpha_1, \ldots, \alpha_n \in 2^\mathcal{V}$ induce a partition of the variable set $\mathcal{V}_\gamma$ of γ, and (3) for each free variable $x \in \mathcal{V}_\gamma$ there is some (ℓ_i, α_i) such that $\ell_i = x$ and $x \in \alpha_i$. It is easy to construct an NFA $\mathcal{B}_1^\gamma$ of exponential size that given $w = (\ell_1, \alpha_1) \cdots (\ell_n, \alpha_n)$ with $w_\lambda = \ell_1 \cdots \ell_n$, checks if the α_i's are valid γ-annotations. Note that if the latter holds, then the annotations encode a mapping h_w from $\mathcal{V}_\gamma$ to the variables of λ such that $h_w(\bar{x}) = \bar{x}$, where $\bar{x}$ are the free variables of γ.

Now, given $w = (\ell_1, \alpha_1)(\ell_2, \alpha_2) \cdots (\ell_n, \alpha_n)$ with $w_\lambda = \ell_1 \cdots \ell_n$ and the α_i's being valid γ-annotations, it is shown in [14] that one can construct in polynomial time a 2NFA $\mathcal{B}_2^\gamma$ that checks the existence of an expansion λ' of γ and a homomorphism h from λ' to λ consistent with h_w. For each atom $x \xrightarrow{L} y$ of γ, the automaton $\mathcal{B}_2^\gamma$ guesses an oriented path π in λ from $h_w(x)$ to $h_w(y)$ with label $w' \in L$, directly over the encoding w_λ starting at a position j_x and ending at a position j_y in $\{0, \ldots, n\}$ (recall that the head moves in $\{0, \ldots, n\}$) with $j_x, j_y > 0$, $w[j_x] = (\ell, \alpha)$, $w[j_y] = (\ell', \alpha')$, $x \in \alpha$ and $y \in \alpha'$. Note that we have two types of transitions: (1) transitions that consume $a \in \mathbb{A}^\pm$ and actually guess an atom of π, and (2) transitions to "jump" from position j to j' in $\{0, \ldots, n\}$ representing *equivalent* variables of λ. The latter means that $j, j' > 0$ and either $w_\lambda[j]$ and $w_\lambda[j']$ represents exactly the *same* variable of λ, or $w_\lambda[j]$ and $w_\lambda[j']$ represent variables z, z' of λ such that $z =_\lambda^* z'$, where $=_\lambda^*$ is the reflexive-transitive closure of the relation induced by the equality atoms in λ.

Let $\mathcal{D}_2^\gamma$ be the 2DA obtained from the 2NFA $\mathcal{B}_2^\gamma$ by setting to 0 and 1 the cost of transitions of type (2) and (1), respectively. Hence, for a word w such that the projection of w to $\mathbb{A}_1$ is w_λ, and the one to $(2^\mathcal{V} \cup \{\#\})$ is a valid γ-annotation, we have that $cost_{\mathcal{D}_2^\gamma}(w)$ is precisely the minimum size of an expansion λ' that can be mapped to λ via a homomorphism compatible with h_w. By Proposition 9, we can construct in exponential time in $\mathcal{D}_2^\gamma$ a DA $\mathcal{C}_2^\gamma$ accepting the same language as $\mathcal{D}_2^\gamma$ and having an exponential number of states, so that for every word w', we have $cost_{\mathcal{C}_2^\gamma}(w') \leq cost_{\mathcal{D}_2^\gamma}(w') \leq f(cost_{\mathcal{C}_2^\gamma}(w'))$ for some polynomial function f. Let $\exists \mathcal{C}^\gamma$ be the result of taking the product of $\mathcal{B}_1^\gamma$ and $\mathcal{C}_2^\gamma$ and then projecting over the alphabet $\mathbb{A}_1$.

For every expansion λ of Γ, if λ' is a minimal size expansion of γ such that $\lambda' \to \lambda$, then we obtain that $cost_{\exists\mathcal{C}^\gamma}(w_\lambda) \leq \|\lambda'\| \leq f(cost_{\exists\mathcal{C}^\gamma}(w_\lambda))$. We define our desired $\mathcal{C}_\Gamma$ to be the union of $\exists\mathcal{C}^\gamma$ over all γ in Γ. We have that for every expansion λ, if λ_{min} is a minimal size expansion of Γ such that $\lambda_{min} \to \lambda$, then $cost_{\mathcal{C}_\Gamma}(w_\lambda) \leq \|\lambda_{min}\| \leq f(cost_{\mathcal{C}_\Gamma}(w_\lambda))$. By Proposition 3, item (2), Γ is bounded iff $\|\lambda_{min}\|$ is bounded over all λ. The latter condition holds iff $\mathcal{C}_\Gamma$ is limited over words w_λ, for all expansion λ. By definition, the latter is equivalent to $\mathcal{D}_\Gamma$ being limited. Summing up, we obtain that Γ is bounded iff $\mathcal{D}_\Gamma$ is limited, as required. Note that the whole construction can be done in exponential time. ◀

As a corollary to Proposition 12 and Theorem 8 we obtain the desired upper bound for part (1) of Theorem 11.

▶ **Corollary 13.** Boundedness *for UC2RPQs is in* ExpSpace.

Size of equivalent UCQs. Here we prove part (2) of Theorem 11. Since Γ is bounded we have from Proposition 12 that $\mathcal{D}_\Gamma$ is limited. Then, from Theorem 8 we obtain that the maximum cost that it takes $\mathcal{D}_\Gamma$ over a word is N, where N is exponential in the number of states of $\mathcal{D}_\Gamma$, and thus double exponential in $\|\Gamma\|$ by construction. Therefore, for every expansion λ of Γ, if λ_{min} is a minimal size expansion Γ such that $\lambda_{min} \to \lambda$, then $\|\lambda_{min}\| \leq f(N)$, where f is the polynomial function of the proof of Proposition 12. In particular, all minimal expansions of Γ are of size $\leq f(N)$. By Lemma 2, the UC2RPQ Γ is equivalent to the union of all its minimal expansions. The number of such minimal expansions is thus at most exponential in $f(N)$, and hence triple exponential in $\|\Gamma\|$.

6.2 Lower bounds

We reduce from the 2^n-tiling problem, that is, a tiling problem restricted to 2^n many columns, which is ExpSpace-complete (see, *e.g.*, [14]). We show that for every 2^n-tiling problem T there is a CRPQ γ, computable in polynomial time from T, whose number of minimal expansions is essentially the number of solutions to T in the following sense.

▶ **Lemma 14.** *For every 2^n-tiling problem T with m solutions there is a Boolean CRPQ γ, computable in polynomial time from T, such that the number of minimal expansions of γ is $O((g(|T|) + m)^{n+1})$ and $\Omega(m)$, for some double exponential function g. Further, γ consists of a Boolean CRPQ of the form $\exists x, y \bigwedge_{0 \leq i \leq n}(x \xrightarrow{L_i} y)$, where each L_i is given as a regular expression.*

As a corollary, this yields an ExpSpace lower bound for the boundedness problem (part (1) of Theorem 11), as well as a triple exponential lower bound for the size of the UCQ equivalent to any bounded CRPQ (part (3) of Theorem 11), since one can produce 2^n-tiling problems having triple-exponentially many solutions.

7 Better-behaved Classes of UC2RPQs

Here we present two restrictions of UC2RPQs that exhibit a better behavior in terms of the complexity of Boundedness than the general case, namely, *acyclic UC2RPQs of bounded thickness* and *strongly connected UCRPQs*. The improved bounds are PSpace and Π_2^P, respectively, which turn out to be optimal.

Acyclic UC2RPQs of Bounded Thickness. For any two distinct variables x, y of a C2RPQ γ, we denote by $\text{Atoms}_\gamma(x, y)$ the set of atoms in γ of the form $x \xrightarrow{L} y$ or $y \xrightarrow{L} x$. The *thickness* of a C2RPQ γ is the maximum cardinality of a set of the form $\text{Atoms}_\gamma(x, y)$, for x, y variables of γ with $x \neq y$. The thickness of a UC2RPQ Γ is the maximum thickness over all the C2RPQs in Γ. The *underlying undirected graph* of γ has as vertex set the set of variables of γ and contains an edge $\{x, y\}$ iff $x \neq y$ and $\text{Atoms}_\gamma(x, y) \neq \emptyset$. A C2RPQ γ is *acyclic* if its underlying undirected graph is an acyclic graph (*i.e.*, a forest). A UC2RPQ Γ is acyclic if each C2RPQ in Γ is.

We show next that Boundedness for acyclic UC2RPQs of bounded thickness is PSpace-complete. These classes of UC2RPQs have been previously studied in the literature [4, 5]. In particular, it follows from [5, Theorem 4.2] that the containment problem for the acyclic UC2RPQs of bounded thickness is PSpace-complete, and hence Theorem 15 below shows that Boundedness is not more costly than containment for these classes.

▶ **Theorem 15.** *Fix $k \geq 1$. The problem* Boundedness *is* PSpace*-complete for acyclic UC2RPQs of thickness at most k.*

Proof (sketch). The lower bound follows directly from PSpace-hardness of Boundedness for RPQs (see Corollary 7). For the PSpace upper bound, we follow a similar strategy as in the case of arbitrary UC2RPQs (Section 6.1), *i.e.*, we reduce boundedness of Γ to DA limitedness. The main difference is that, since Γ is acyclic, we can exploit the power of alternation and construct an A2DA$^\varepsilon$ $\mathcal{B}$ (instead of a 2DA, as in the proof of Proposition 12), such that Γ is bounded iff $\mathcal{B}$ is limited. The constant upper bound on the thickness of Γ implies that $\mathcal{B}$ is actually of polynomial size. The result follows then as limitedness of an A2DA$^\varepsilon$ can be decided in PSpace in virtue of Theorem 10. ◀

Both conditions in Theorem 15, *i.e.*, acyclicity and bounded thickness, are necessary. Indeed, it follows from Lemma 14 that Boundedness is ExpSpace-hard even for:

- Boolean acyclic CRPQs.
- Boolean CRPQs of thickness one, whose underlying undirected graph is of *treewidth* two. Recall that the treewidth is a measure of how much a graph resembles a tree (cf., [20]) – acyclic graphs are precisely the graphs of treewidth one.

Indeed, the CRPQs of the form $\exists x, y \bigwedge_i (x \xrightarrow{L_i} y)$ used in Lemma 14 are Boolean and acyclic (but have unbounded thickness). Replacing each $(x \xrightarrow{L_i} y)$ with $(x \xrightarrow{\varepsilon} z_i) \wedge (z_i \xrightarrow{L_i} y)$, yields an equivalent CRPQ of thickness one whose underlying undirected graph has treewidth two.

Strongly Connected UCRPQs. We conclude this section with an even better behaved class of CRPQs in terms of Boundedness. Unlike the previous case, the definition of this class depends on the underlying *directed* graph of a CRPQ γ. This contains a directed edge from variable x to y iff there is an atom in γ of the form $x \xrightarrow{L} y$. A CRPQ γ is *strongly connected* if its underlying directed graph is strongly connected, *i.e.*, every pair of variables is connected by some directed path. A UCRPQ Γ is strongly connected if every CRPQ in Γ is. We can then establish the following.

▶ **Theorem 16.** Boundedness *is Π_2^P-complete for strongly connected UCRPQs.*

8 Discussion and Future Work

The main conclusion of our work is that techniques previously used in the study of containment of UC2RPQs can be naturally leveraged to pinpoint the complexity of Boundedness by using DA instead of NFA. This, however, requires extending results on limitedness to alternating and two-way DA. For all the classes of UC2RPQs studied in the paper we show in fact that the complexity of Boundedness coincides with that of the containment problem. We leave open what is the exact size of UCQ rewritings for the classes of acyclic UC2RPQs of bounded thickness and the strongly connected UCRPQs that are bounded.

The most natural next step is to study Boundedness for the class of *regular queries* (RQs), which are the closure of UC2RPQs under binary transitive closure. RQs are one of the most powerful recursive languages for which containment is decidable in elementary time. In fact, containment of RQs has been proved to be 2EXPSPACE-complete by applying sophisticated techniques based on NFA [33]. We will study if it is possible to settle the complexity of Boundedness for RQs with the help of DA techniques.

Another interesting future line of work is the study of Boundedness for UC2RPQs based on the restricted classes of regular expressions often found in practical applications [13]. As it has been shown lately, the complexity of some query evaluation problems is alleviated under this restriction [30], and it would be nice to see if the same holds for the boundedness problem. This would be good news for the applicability of boundedness techniques in practical applications. In fact, it would be an indication that the high complexity lower bounds obtained in this paper are mostly witnessed by complicated interactions between regular expressions not commonly arising in practice.

References

1 Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan L. Reutter, and Domagoj Vrgoč. Foundations of Modern Query Languages for Graph Databases. *ACM Computing Surveys*, 50(5):68:1–68:40, 2017.

2 Pablo Barceló. Querying graph databases. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 175–188, 2013.

3 Pablo Barceló, Gerald Berger, Carsten Lutz, and Andreas Pieris. First-Order Rewritability of Frontier-Guarded Ontology-Mediated Queries. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1707–1713, 2018.

4 Pablo Barceló, Miguel Romero, and Moshe Y. Vardi. Does Query Evaluation Tractability Help Query Containment? In *ACM Symposium on Principles of Database Systems (PODS)*, pages 188–199, 2014.

5 Pablo Barceló, Miguel Romero, and Moshe Y. Vardi. Semantic Acyclicity on Graph Databases. *SIAM Journal on computing*, 45(4):1339–1376, 2016.

6 Michael Benedikt, Pierre Bourhis, and Michael Vanden Boom. A Step Up in Expressiveness of Decidable Fixpoint Logics. In *Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 817–826, 2016.

7 Michael Benedikt, Balder ten Cate, Thomas Colcombet, and Michael Vanden Boom. The Complexity of Boundedness for Guarded Logics. In *Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 293–304. IEEE Computer Society Press, 2015. doi:10.1109/LICS.2015.36.

8 Meghyn Bienvenu, Peter Hansen, Carsten Lutz, and Frank Wolter. First Order-Rewritability and Containment of Conjunctive Queries in Horn Description Logics. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 965–971, 2016.

9 Jean-Camille Birget. State-complexity of finite-state devices, state compressibility and incompressibility. *Mathematical systems theory*, 26(3):237–269, 1993.

10 Achim Blumensath, Thomas Colcombet, Denis Kuperberg, Pawel Parys, and Michael Vanden Boom. Two-way cost automata and cost logics over infinite trees. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14*, pages 16:1–16:9. ACM Press, 2014. doi:10.1145/2603088.2603104.

11 Achim Blumensath, Martin Otto, and Mark Weyer. Decidability Results for the Boundedness Problem. *Logical Methods in Computer Science (LMCS)*, 10(3), 2014.

12 Mikołaj Bojańczyk and Szymon Toruńczyk. Deterministic Automata and Extensions of Weak MSO. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS)*, volume 4 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 73–84. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2009. doi:10.4230/LIPIcs.FSTTCS.2009.2308.

13 Angela Bonifati, Wim Martens, and Thomas Timm. An Analytical Study of Large SPARQL Query Logs. *PVLDB*, 11(2):149–161, 2017.

14 Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. Containment of Conjunctive Regular Path Queries with Inverse. In *Principles of Knowledge Representation and Reasoning (KR)*, pages 176–185, 2000.

15 Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. Rewriting of Regular Expressions and Regular Path Queries. *Journal of Computer and System Sciences (JCSS)*, 64(3):443–465, 2002.

16 Ashok K. Chandra and Philip M. Merlin. Optimal Implementation of Conjunctive Queries in Relational Data Bases. In *Symposium on Theory of Computing (STOC)*, pages 77–90, 1977.

17 Thomas Colcombet. The Theory of Stabilisation Monoids and Regular Cost Functions. In *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 5556 of *Lecture Notes in Computer Science*, pages 139–150. Springer, 2009. doi:10.1007/978-3-642-02930-1_12.

18 Thomas Colcombet and Christof Löding. The Nesting-Depth of Disjunctive μ-Calculus for Tree Languages and the Limitedness Problem. In *EACSL Annual Conference on Computer Science Logic (CSL)*, pages 416–430, 2008.

19 Stavros S. Cosmadakis, Haim Gaifman, Paris C. Kanellakis, and Moshe Y. Vardi. Decidable Optimization Problems for Database Logic Programs (Preliminary Report). In *Symposium on Theory of Computing (STOC)*, pages 477–490, 1988.

20 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

21 Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of weighted automata*. Springer Science & Business Media, 2009.

22 Haim Gaifman, Harry G. Mairson, Yehoshua Sagiv, and Moshe Y. Vardi. Undecidable Optimization Problems for Database Logic Programs. *J. ACM*, 40(3):683–713, 1993.

23 Peter Hansen, Carsten Lutz, Inanç Seylan, and Frank Wolter. Efficient Query Rewriting in the Description Logic EL and Beyond. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3034–3040, 2015.

24 Kosaburo Hashiguchi. Limitedness theorem on finite automata with distance functions. *Journal of Computer and System Sciences (JCSS)*, 24(2):233–244, 1982.

25 Gerd G. Hillebrand, Paris C. Kanellakis, Harry G. Mairson, and Moshe Y. Vardi. Tools for Datalog Boundedness. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 1–12, 1991.

26 Daniel Kirsten. Distance desert automata and the star height problem. *Informatique Théorique et Applications (ITA)*, 39(3):455–509, 2005.

27 Dexter Kozen. Lower Bounds for Natural Proof Systems. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 254–266, 1977.

28 Hing Leung. Limitedness Theorem on Finite Automata with Distance Functions: An Algebraic Proof. *Theoretical Computer Science (TCS)*, 81(1):137–145, 1991. doi:10.1016/0304-3975(91)90321-R.

29 Hing Leung and Viktor Podolskiy. The limitedness problem on distance automata: Hashiguchi's method revisited. *Theoretical Computer Science (TCS)*, 310(1-3):147–158, 2004. doi:10.1016/S0304-3975(03)00377-3.

30 Wim Martens and Tina Trautner. Evaluation and Enumeration Problems for Regular Path Queries. In *International Conference on Database Theory (ICDT)*, pages 19:1–19:21, 2018.

31 Jeffrey F. Naughton. Data Independent Recursion in Deductive Databases. *Journal of Computer and System Sciences (JCSS)*, 38(2):259–289, 1989.

32 Nir Piterman and Moshe Y. Vardi. From bidirectionality to alternation. *Theoretical Computer Science (TCS)*, 295:295–321, 2003. doi:10.1016/S0304-3975(02)00410-3.

33 Juan L. Reutter, Miguel Romero, and Moshe Y. Vardi. Regular Queries on Graph Databases. *Theoretical Computer Science (TCS)*, 61(1):31–83, 2017.

34 John C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM Journal of Research and Development*, 3(2):198–200, 1959.

35 Michael Vanden Boom. Weak Cost Monadic Logic over Infinite Trees. In *International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 580–591, 2011.

36 Michael Vanden Boom. *Weak cost automata over infinite trees*. PhD thesis, University of Oxford, UK, 2012.

Polynomially Ambiguous Probabilistic Automata on Restricted Languages

Paul C. Bell
Department of Computer Science, Byrom Street, Liverpool John Moores University, Liverpool, L3-3AF, UK
p.c.bell@ljmu.ac.uk

Abstract

We consider the computability and complexity of decision questions for Probabilistic Finite Automata (PFA) with sub-exponential ambiguity. We show that the emptiness problem for non-strict cut-points of polynomially ambiguous PFA remains undecidable even when the input word is over a bounded language and all PFA transition matrices are commutative. In doing so, we introduce a new technique based upon the Turakainen construction of a PFA from a Weighted Finite Automata which can be used to generate PFA of lower dimensions and of subexponential ambiguity. We also study freeness/injectivity problems for polynomially ambiguous PFA and study the border of decidability and tractability for various cases.

2012 ACM Subject Classification Theory of computation → Quantitative automata; Theory of computation → Probabilistic computation

Keywords and phrases Probabilistic finite automata, ambiguity, undecidability, bounded language, emptiness

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.105

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Acknowledgements We thank the referees for their careful reading of this manuscript and their helpful improvements.

1 Introduction

Probabilistic finite automata (PFA) are a simple yet expressive model of computation, obtained by extending nondeterministic finite automata so that transitions from each state (and for each input letter) form probability distributions. As input letters are read from some alphabet Σ, the automaton transitions among states according to these probabilities. The probability of accepting a word $w \in \Sigma^*$ is given by the probability of the automaton being in one of its final states, denoted $f_{\mathcal{P}}(w) = \mathbf{x}^T M_{w_1} M_{w_2} \cdots M_{w_k} \mathbf{y}$, where $\mathbf{x}$ represents the initial state, $\mathbf{y}$ represents the final state and each M_{w_i} is a row stochastic matrix representing the transition probabilities for letter $w_i \in \Sigma$.

The PFA model has been studied extensively over the years, ever since its introduction by Rabin [27]; for example see [10] for a survey of 416 research papers related to PFA in the eleven years since their introduction to just 1974. They have been used to study Arthur-Merlin games [2], space bounded interactive proofs [15], quantum complexity theory [33], the joint spectral radius and semigroup boundedness [8], Markov decision processes and planning questions [9], and text and speech processing [24] among many others.

There are a variety of interesting questions that one may ask about PFA. A central question is the emptiness problem for cut-point languages; given some probability $\lambda \in [0, 1]$, does there exist a finite input word whose probability of acceptance is greater than λ (i.e. does there exist $w \in \Sigma^*$ such that $f_{\mathcal{P}}(w) > \lambda$, see Section 2.2). This problem is known to be undecidable [26], even for a fixed number of dimensions and for two input matrices [7, 19].

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 105; pp. 105:1–105:14


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

A second natural question is the *freeness problem* (or *injectivity problem*) for PFA, studied in [3] – given a PFA $\mathcal{P}$ over alphabet Σ determine whether the acceptance function $f_{\mathcal{P}}(w)$ is injective (i.e. do there exist two distinct words with the same acceptance probability).

When studying the frontiers of decidability of a problem, there are two competing objectives, namely, determine the most general version of the problem which is decidable, and the most restricted specialization which is undecidable; the latter being the focus of this paper.

Various classes of restrictions may be studied for PFA depending upon the structure of the PFA or on possible input words. Some restrictions relate to the number of states of the automaton, the alphabet size and whether one defined the PFA over the algebraic real numbers or the rationals. One may also study PFA with finite, polynomial or exponential ambiguity (in terms of the underlying NFA), PFA defined for restricted input words (for example those coming from regular, bounded or letter monotonic languages), PFA with isolated thresholds (a probability threshold is isolated if it cannot be approached arbitrarily closely) and PFA where all matrices commute, for which cut-point languages and non-free languages generated by such automata necessarily become commutative.

The cut-point emptiness problem for PFA is known to be undecidable for rational matrices [26], even over a binary alphabet when the PFA has dimension 46 in [7]; later improved to dimension 25 [19]. The authors of [6] show that the problem of determining if a threshold is isolated (resp. if a PFA has any isolated threshold) is undecidable and this was shown to hold even for PFA with 420 (resp. 2354) states over a binary alphabet [7].

A natural restriction on PFA was studied in [4], where possible input words of the PFA are restricted to be from some letter monotonic language of the form $\mathcal{L} = a_1^* a_2^* \cdots a_k^*$ with each $a_i \in \Sigma$ (analogous to a 1.5 way PFA, whose read head may "stay put" on an input word letter but never moves left), then the problem remains undecidable. In other words, does there exist $w \in \mathcal{L}$ such that $f_{\mathcal{P}}(w) > \lambda$? This restriction is inspired by the well-known property that many language-theoretic problems become decidable or tractable when restricted to bounded languages, and especially letter-monotonic languages [13]. Nevertheless, the emptiness problem for PFA on letter-monotonic languages was shown to be undecidable for high (but finite) dimensional matrices over the rationals via an encoding of Hilbert's tenth problem on the solvability of Diophantine equations and the utilization of Turakainen's method to transform weighted integer automata to a PFA [4].

The authors of [17] recently studied decision problems for PFA of various degrees of ambiguity in order to map the frontier of decidability for restricted classes of PFA. The degree of ambiguity of a PFA is defined as the maximum number of accepting runs over all possible words and can be used to give various classifications of ambiguity including finite, polynomial and exponential ambiguity. The ambiguity of a PFA is a property of the underlying NFA and is independent of the transition probabilities in so much as we only need care if the probability is zero or positive. The degree of ambiguity of automata is a well-known and well-studied property in automata theory [31]. The authors of [17] show that the emptiness problem for PFA remains undecidable even for polynomially ambiguous automata (quadratic ambiguity), before going on to show **PSPACE**-hardness results for finitely ambiguous PFA and that emptiness is in **NP** for the class of k-ambiguous PFA for every $k > 0$. The emptiness problem for PFA was later shown to also be undecidable even for linearly ambiguous automata in [16].

1.1 Our Contributions

In this paper, we show that the emptiness problem is undecidable even for polynomially ambiguous PFA defined over letter monotonic languages when all matrices are rational and commutative. This combination of restrictions on the PFA significantly increases the

difficulty of proving undecidability. The study of PFA over letter monotonic languages is a particularly interesting intermediate model, lying somewhere between single letter alphabets, for which we have decidability results, and PFA defined with multi-letter alphabets, for which most decision problems are undecidable.

▶ **Theorem 1.** *The emptiness problem for polynomially ambiguous probabilistic finite automata on letter monotonic languages is undecidable for non-strict cut-points, even when all matrices are commutative.*

We note a few difficulties with proving this result. Firstly, Post's correspondence problem, whose variants are often used for showing undecidability results in such settings, is actually decidable over letter monotonic languages [18][1]. Secondly, although other reductions of undecidable computational problems to matrices are possible, the standard technique of Turakainen (shown in [30]) to modify such matrices to stochastic matrices introduces exponential ambiguity (indeed all such matrices are strictly positive, and thus we might think of such matrices as being *maximally exponentially ambiguous*)[2]. Finally, we note that matrix problems for commutative matrices are often decidable; indeed there are polynomial time algorithms for solving the orbit problem [22, 14] and the vector reachability problem for commutative matrices [1]. Since the matrices commute, it is the Parikh vector of letters of the input word which is important.

We use a reduction of Hilbert's tenth problem and various new encoding techniques to avoid the use of Turakainen's method for converting from weighted to probabilistic automata, so as to retain polynomial ambiguity. We then move on to the freeness/injectivity problem to show the following two results.

▶ **Theorem 2.** *The injectivity problem for linearly ambiguous four state probabilistic finite automata is undecidable.*

▶ **Theorem 3.** *The injectivity problem for linearly ambiguous three-state probabilistic finite automata over letter-monotonic languages is NP-hard.*

These results are proven via an encoding of the mixed modification PCP and our new encoding technique and the injectivity problem for three state PFA over letter monotonic languages is **NP**-hard via an encoding of the variant of the subset sum problem and a novel encoding technique. We conclude with some open problems.

2 Preliminaries

2.1 Linear Algebra

Given $A = (a_{ij}) \in \mathbb{F}^{m\times m}$ and $B \in \mathbb{F}^{n\times n}$, we define the direct sum $A \oplus B$ and Kronecker product $A \otimes B$ of A and B by:

$$A \oplus B = \left[\begin{array}{c|c} A & \mathbf{0}_{m,n} \\ \hline \mathbf{0}_{n,m} & B \end{array}\right], \quad A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1m}B \\ a_{21}B & a_{22}B & \cdots & a_{2m}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mm}B \end{bmatrix},$$

[1] Although it is undecidable in general (i.e. not over a letter monotonic language) with an alphabet with at least five letters [25].

[2] This is due to an essential step of the Turakainen procedure that adds a positive constant offset to each element of every generator matrix, thus making all matrices strictly positive [30].

where $\mathbf{0}_{i,j}$ denotes the zero matrix of dimension $i \times j$. Note that neither $\oplus$ nor $\otimes$ are commutative in general. Given a finite set of matrices $\mathcal{G} = \{G_1, G_2, \ldots, G_m\} \subseteq \mathbb{F}^{n\times n}$, $\langle \mathcal{G} \rangle$ is the semigroup generated by $\mathcal{G}$. We will use the following notations:

$$\bigoplus_{j=1}^{m} G_j = G_1 \oplus G_2 \oplus \cdots \oplus G_m, \qquad \bigotimes_{j=1}^{m} G_j = G_1 \otimes G_2 \otimes \cdots \otimes G_m$$

Given a single matrix $G \in \mathbb{F}^{n\times n}$, we inductively define $G^{\otimes k} = G \otimes G^{\otimes(k-1)} \in \mathbb{F}^{n^k \times n^k}$ for $k > 0$ with $G^{\otimes 0} = 1$ as the k-fold Kronecker power of G. Similarly, $G^{\oplus k} = G \oplus G^{\oplus(k-1)} \in \mathbb{F}^{n^k \times n^k}$ for $k > 0$ with $G^{\oplus 0}$ being a zero dimensional matrix. The rationalle for the base cases is that $G \otimes G^{\otimes 0} = 1 \otimes G = G$ and that $G \oplus G^{\oplus 0} = G$ as expected.

The following properties of $\oplus$ and $\otimes$ are well known, see [20] for proofs.

▶ **Lemma 4.** *Let $A, B, C, D \in \mathbb{F}^{n\times n}$. We note that:*

- *Associativity - $(A \otimes B) \otimes C = A \otimes (B \otimes C)$ and $(A \oplus B) \oplus C = A \oplus (B \oplus C)$, thus $A \otimes B \otimes C$ and $A \oplus B \oplus C$ are unambiguous.*
- *Mixed product properties: $(A \otimes B)(C \otimes D) = (AC \otimes BD)$ and $(A \oplus B)(C \oplus D) = (AC \oplus BD)$.*
- *If A and B are stochastic matrices, then so are $A \oplus B$ and $A \otimes B$.*

It is trivial to prove that if $A, B \in \mathbb{F}^{n\times n}$ are both upper-triangular then so are $A \oplus B$ and $A \otimes B$. This follows directly from the definition of the Kronecker sum and product.

2.2 Probabilistic Finite Automata (PFA)

Probabilistic Finite Automata (PFA) with n states over an alphabet Σ are defined as $\mathcal{P} = (\mathbf{x}, \{M_a | a \in \Sigma\}, \mathbf{y})$ where $\mathbf{x} \in \mathbb{R}^n$ is the initial probability distribution; $\mathbf{y} \in \{0,1\}^n$ is the final state vector and each $M_a \in \mathbb{R}^{n\times n}$ is a (row) stochastic matrix. For a word $w = w_1 w_2 \cdots w_k \in \Sigma^*$, we define the acceptance probability $f_{\mathcal{P}} : \Sigma^* \to \mathbb{R}$ of $\mathcal{P}$ as:

$$f_{\mathcal{P}}(w) = \mathbf{x}^T M_{w_1} M_{w_2} \cdots M_{w_k} \mathbf{y},$$

which denotes the acceptance probability of w.

For any $\lambda \in [0,1]$ and PFA $\mathcal{A}$ over alphabet Σ, we define a cut-point language to be: $L_{\geq\lambda}(\mathcal{A}) = \{w \in \Sigma^* | f_{\mathcal{A}}(w) \geq \lambda\}$, and a strict cut-point language $L_{>\lambda}(\mathcal{A})$ by replacing $\geq$ with $>$. The (strict) emptiness problem for a cut-point language is to determine if $L_{\geq\lambda}(A) = \emptyset$ (resp. $L_{>\lambda}(A) = \emptyset$).

Let $\Sigma_\ell = \{x_1, x_2, \ldots, x_\ell\}$ be an ℓ-letter alphabet for some $\ell > 0$. A language $L \subseteq \Sigma_\ell^*$ is called a *bounded language* if and only if there exist words $w_1, w_2, \ldots, w_m \in \Sigma_\ell^+$ such that $L \subseteq w_1^* w_2^* \cdots w_m^*$. A language L is called *letter-monotonic* if there exists letters $u_1, u_2, \ldots, u_m \in \Sigma_\ell$ such that $L \subseteq u_1^* u_2^* \cdots u_m^*$. One thus sees that letter monotonic languages are more restricted than bounded languages. We will be interested in PFA which are defined over a bounded language or a letter monotonic language $\mathcal{L}$, whereby all input words necessarily come from $\mathcal{L}$. In this case a cut-point language for a PFA $\mathcal{P}$ over bounded/letter monotonic language $\mathcal{L}$ and a probability $\lambda \in [0,1]$ is defined as $L_{\geq\lambda}(\mathcal{A}) = \{w \in L | f_{\mathcal{A}}(w) \geq \lambda\}$; similarly for nonstrict cut point languages. We may then ask similar emptiness questions for such languages, as before.

We also study the *freeness/injectivity problem* for PFA. Given a PFA $\mathcal{P}$ over alphabet Σ determine whether the acceptance function $f_{\mathcal{P}}(w)$ is injective (i.e. do there exist two distinct words with the same acceptance probability). Such problems can readily be studied when the input words are necessarily derived from a bounded or letter-monotonic language.

2.3 PFA Ambiguity

The degree of ambiguity of a finite automaton is a structural parameter, roughly indicating the number of accepting runs for a given input word [31]. We here define only those notions required for our later proofs, see [31] for full details of these notions and a thorough discussion.

Let $w \in \Sigma^*$ be an input word of an NFA $\mathcal{N} = (Q, \Sigma, \delta, Q_I, Q_F)$. For each $(p, w, q) \in Q \times \Sigma^* \times Q$, let $\mathrm{da}_{\mathcal{N}}(p, w, q)$ be defined as the number of all paths for w in $\mathcal{N}$ leading from state p to state q. The degree of ambiguity of w in $\mathcal{N}$, denoted $\mathrm{da}_{\mathcal{N}}(w)$, is defined as the number of all *accepting paths* for w. The degree of ambiguity of $\mathcal{N}$, denoted $\mathrm{da}(\mathcal{N})$ is the supremum of the set $\{\mathrm{da}_{\mathcal{N}}(w) | w \in \Sigma^*\}$. $\mathcal{N}$ is called infinitely ambiguous if $\mathrm{da}(\mathcal{N}) = \infty$, finitely ambiguous if $\mathrm{da}(\mathcal{N}) < \infty$, and unambiguous if $\mathrm{da}(\mathcal{N}) \leq 1$. The degree of growth of the ambiguity of $\mathcal{N}$, denoted $\deg(M)$ is defined as the minimum degree of a univariate polynomial h with positive integral coefficients such that for all $w \in \Sigma^*$, $\mathrm{da}_{\mathcal{N}}(w) \leq h(|w|)$ if such a polynomial exists, or infinity otherwise.

The above notions relate to NFA. We may derive an analogous notation of ambiguity for PFA by considering an embedding of a PFA $\mathcal{P}$ to an NFA $\mathcal{N}$ with the property that for each letter $a \in \Sigma$, if the probability of transitioning from a state i to state j is nonzero under $\mathcal{P}$, then there is an edge from state i to j under $\mathcal{N}$ for letter a. The degree of (growth of) ambiguity of $\mathcal{P}$ is then defined as the degree of (growth of) ambiguity of $\mathcal{N}$.

We may use the following notions to determine the degree of ambiguity of a given NFA (and thus a PFA) $\mathcal{A}$ as is shown in the theorem which follows. A state $q \in Q$ is called *useful* if there exists an accepting path which visits q.

EDA. There is a useful state $q \in Q$ such that, for some word $v \in \Sigma^*$, $da_{\mathcal{A}}(q, v, q) \geq 2$.

IDA$_d$. There are useful states $r_1, s_1, \ldots, r_d, s_d \in Q$ and words $v_1, u_2, v_2, \ldots, u_d, v_d \in \Sigma^*$ such that for all $1 \leq \lambda \leq d$, r_λ and s_λ are distinct and $(r_\lambda, v_\lambda, r_\lambda), (r_\lambda, v_\lambda, s_\lambda), (s_\lambda, v_\lambda, s_\lambda) \in \delta$ and for all $2 \leq \lambda \leq d$, $(s_{\lambda-1}, u_\lambda, r_\lambda) \in \delta$.

▶ **Theorem 5** ([21, 28, 31]). *An NFA (or PFA) $\mathcal{A}$ having the EDA property is equivalent to it being exponentially ambiguous. For any $d \in \mathbb{N}$, an NFA (or PFA) $\mathcal{A}$ having property IDA$_d$ is equivalent to $\deg(\mathcal{A}) \geq d$.*

Clearly, if $\mathcal{N}$ agrees with IDA$_d$ for some $d > 0$, then it also agrees with IDA$_1, \ldots,$ IDA$_{d-1}$. One must be careful with these notions of ambiguity when considering NFA/PFA $\mathcal{A}$, where inputs are necessarily from a bounded language $\mathcal{L}$. In such cases, the above criteria do not suffice to determine the ambiguity of $\mathcal{A}$, since the number of paths must be determined not over Σ^*, but over all paths from $\mathcal{L}$. Of course, the degree of ambiguity of $\mathcal{A}$ cannot *increase* by restricting to a bounded input language, but it may decrease.

As an example, if an NFA has property EDA, then there exists three words w_1, w_2 and w_3 such that $w_1w_2w_3$ is an accepting word and $da_{\mathcal{A}}(q, w_2, q) \geq 2$, thus $w_1w_2w_3$ has at least two distinct accepting runs. However, this implies that $da_{\mathcal{A}}(w_1w_2^kw_3) \geq 2^k$ and thus $w_1w_2^kw_3$ has at least 2^k accepting runs. Now, if we are given some bounded language $\mathcal{L}$ such that $w_1w_2w_3 \in \mathcal{L}$ and $da_{\mathcal{A}}(q, w_2, q) \geq 2$ then the same implication is not possible, unless $w_2 \in \Sigma$ is a single letter, otherwise there is no guarantee that $w_1w_2^kw_3 \in \mathcal{L}$. Nevertheless, in the results of this paper we will use the standard definitions of ambiguity since the distinction is not relevant in our results as will become clear.

We note the following trivial lemma, which will be useful later.

▶ **Lemma 6.** *Probabilistic finite automata defined over upper-triangular matrices are polynomially ambiguous.*

Proof. This lemma is immediate from Theorem 5 and property (EDA), since a PFA defined over upper-triangular matrices clearly does not have property (EDA). This is since a transition matrix (for a letter "a") which is upper-triangular only defines transitions of the form $\delta(i,a) = j$ where $i \leq j$ and thus the states entered for any run are monotonically nondecreasing. ◀

2.4 Reducible Undecidable Problems

We will require the following undecidable problems for proving later results. The first is a variant of the famous *Post's Correspondence Problem* (PCP).

▶ **Problem 7** (Mixed Modification PCP (MMPCP)). *Given a binary alphabet Σ_2, a finite set of letters $\Sigma = \{s_1, s_2, \ldots, s_{|\Sigma|}\}$, and a pair of homomorphisms $h, g : \Sigma^* \to \Sigma_2^*$, the MMPCP asks to decide whether there exists a word $w = x_1 \ldots x_k \in \Sigma^+, x_i \in \Sigma$ such that*

$$h_1(x_1)h_2(x_2)\ldots h_k(x_k) = g_1(x_1)g_2(x_2)\ldots g_k(x_k),$$

where $h_i, g_i \in \{h, g\}$, and there exists at least one j such that $h_j \neq g_j$.

▶ **Theorem 8** ([12]). *The Mixed Modification PCP is undecidable for $|\Sigma| \geq 9$.*

A second useful undecidable problem is *Hilbert's tenth problem*: Let $P(n_1, n_2, \ldots, n_k)$ be an integer polynomial with k variables - determine if there exists a procedure to find if there exist $x_1, x_2, \ldots, x_k \in \mathbb{Z}$ such that: $P(x_1, x_2, \ldots, x_k) = 0$. It is well known that this may be reduced to a problem in formal power series. It was shown in [29, p.73] that the above problem can be reduced to that of determining for a $\mathbb{Z}$-rational formal power series $S \in \mathbb{Z}\langle\langle A \rangle\rangle$, whether there exists any word $w \in A^*$ such that $(S, w) = 0$. The undecidability of this problem was shown in 1970 by Y. Matiyasevich (building upon work of Davis, Putman, Robinson and others). For more details, see the excellent reference [23]. We may, without loss of generality, restrict the variables to be natural numbers [23, p.6].

3 Cut-point languages for polynomially ambiguous PFA over letter monotonic languages

It was proven in [4] that the emptiness problem is undecidable for probabilistic finite automata even when input words are given over a letter-monotonic language, i.e., given a letter-monotonic language $\mathcal{L}$, it is undecidable to determine if $\{w \in \mathcal{L} | f_{\mathcal{P}}(w) \Delta \lambda\}$ is empty for $\Delta \in \{\leq, <, >, \geq\}$. The constructed PFA $\mathcal{P}$ of [4] has exponential ambiguity, due to the well-known Turakainen conversion of arbitrary integer matrices into stochastic matrices. Here, we show that the emptiness problem for PFA over letter-monotonic languages can also be achieved even when all matrices have polynomial ambiguity by a modified Turakainen procedure.

The following property of the Kronecker product will also be required for the proof of Theorem 1.

▶ **Lemma 9.** *Let $A_1, \ldots, A_\ell \in \mathbb{F}^{n \times n}$. Then for any index sequence $(i_1, j_1), \ldots, (i_\ell, j_\ell)$ with each $(i_s, j_s) \in [1, n] \times [1, n]$ then there exists $1 \leq i, j \leq n^\ell$ such that*

$$\prod_{m=1}^{\ell} (A_m)_{i_m, j_m} = \left(\bigotimes_{m=1}^{\ell} A_m \right)_{i,j}$$

Proof. The proof proceeds by induction. For the base case when $\ell = 1$, we just set $(i, j) = (i_1, j_1)$ and we are done. Assume then that the result holds for some $\ell - 1$, then for sequence $(i_1, j_1), (i_2, j_2), \ldots, (i_{\ell-1}, j_{\ell-1})$ there exists $1 \leq i', j' \leq n^{\ell-1}$ such that:

$$\prod_{m=1}^{\ell-1} (A_m)_{i_m, j_m} = \left(\bigotimes_{m=1}^{\ell-1} A_m \right)_{i', j'}$$

By the definition of Kronecker product,

$$\left(\left(\bigotimes_{m=1}^{\ell-1} A_m \right) \otimes A_\ell \right)_{ni'+i_\ell, nj'+j_\ell} = \prod_{m=1}^{\ell-1} (A_m)_{i_m, j_m} \times (A_\ell)_{i_\ell, j_\ell}$$

as required. ◀

Note that we can of course work out the particular value of i and j, but in general the formula for i, j does not have a nice form when $\ell > 2$, and anyway will not be necessary for us, so we settle for an existential proof of such i and j.

3.1 Proof of Theorem 1

Proof. We will construct a polynomially ambiguous probabilistic finite automaton $\mathcal{P}$, a cut-point $\lambda \in [0, 1]$ and a letter monotonic language $\mathcal{L}$.

We begin by encoding an instance of Hilbert's tenth problem into a set of integer matrices. Let $P(x_1, x_2, \ldots, x_t) = 0$ be a Diophantine equation. Homogenenization of polynomials is a well known technique, as is used for example in the study of Gröbner bases [11], which allows us to convert such a Diophantine equation to $P^h(x_0, x_1, x_2, \ldots, x_t) = 0$ with a new dummy variable x_0 such that P^h is a homogeneous polynomial (each term having the same degree d) and for which $P^h(x_0, x_1, \ldots, x_t) = P(x_1, x_2, \ldots, x_t)$ when $x_0 = 1$. We thus assume a homogeneous Diophantine equation $P^h(x_0, x_1, \ldots, x_t) = 0$ with implied constraint $x_0 = 1$ which will be dealt with later. Furthermore, we assume that P^h gives nonnegative values, which may be assumed by redefining $P^h = (P^h)^2$, which clearly does not affect whether a zero exists for such a polynomial.

Notice that given $A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$, then $A^k = \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}$. We will generalise this property to a set of $t+1$ matrices $A_0, A_1, \ldots, A_t \in \mathbb{Z}^{(t+3)\times(t+3)}$ so that given any tuple $(x_0, x_1, x_2, \ldots, x_t)$, then x_i appears as an element on the superdiagonal of $A_0^{x_0} A_1^{x_1} \cdots A_t^{x_t}$ for each $0 \leq i \leq t$. We will also have the property that each A_i has the same row sum of 2 for every row, which will be useful when we later convert to stochastic matrices.

We define each matrix A_i for $0 \leq i \leq t+1$ in the following way:

$$A_i = \begin{pmatrix} 1 & \delta_{0,i} & 0 & \cdots & 0 & 0 & 1-\delta_{0,i} \\ 0 & 1 & \delta_{1,i} & \cdots & 0 & 0 & 1-\delta_{1,i} \\ 0 & 0 & 1 & \cdots & 0 & 0 & 1-\delta_{2,i} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & \delta_{t,i} & 1-\delta_{t,i} \\ 0 & 0 & 0 & \cdots & 0 & 1 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 2 \end{pmatrix} \in \mathbb{N}^{(t+3)\times(t+3)}, \tag{1}$$

where $0 \leq j \neq i \leq t$ and $\delta_{\ell,i} \in \{0, 1\}$ is the Kronecker delta (thus $\delta_{i,i} = 1$ and $\delta_{\ell,i} = 0$ for $\ell \neq i$). We also denote $J = A_{t+1}$, noting that this is the matrix (1) when all $\delta_{\ell,i}$ have the

value 0. Notice then that every row sum of A_i and J is 2. This structure is retained under matrix powers and it is easy to see that:

$$A_i^k = \begin{pmatrix} 1 & k\delta_{0,i} & 0 & \cdots & 0 & 0 & 2^k - k\delta_{0,i} - 1 \\ 0 & 1 & k\delta_{1,i} & \cdots & 0 & 0 & 2^k - k\delta_{1,i} - 1 \\ 0 & 0 & 1 & \cdots & 0 & 0 & 2^k - k\delta_{2,i} - 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & k\delta_{t,i} & 2^k - k\delta_{t,i} - 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 2^k - 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 2^k \end{pmatrix} \in \mathbb{N}^{(t+3)\times(t+3)} \tag{2}$$

All row sums of A_i^k are 2^k and exactly one element of the superdiagonal is equal to k, with all other elements on the superdiagonal (excluding that on row $t+2$) zero. Taking powers of A_i will allow us to choose any positive value of variable x_i. Note that J^k has the same form as the matrix of (2) with all $\delta_{\ell,i} = 0$ and acts as a kind of identity matrix, (in its upperleft block) while retaining the 2^k row sum. Indeed, one sees that for all $0 \leq i, j \leq t+1$, then $A_iA_j = A_jA_i$, i.e. these matrices commute (as does J since $J = A_{t+1}$). We now show how to compute terms of P^h.

We may write $P^h(x_0, x_1, \ldots, x_t) = \sum_{j=1}^{r} T_j(x_0, x_1, \ldots, x_t)$, where T_j denotes the j'th term of P^h, with P^h having r terms. Since P^h is a homogeneous polynomial, each term has the same degree d. We may thus write each term as:

$$T_j(x_0, x_1, \ldots, x_t) \quad = \quad c_j R_j(x_0, x_1, \ldots, x_t), \tag{3}$$

with $c_j \in \mathbb{Z}$ and $R_j(x_0, x_1, \ldots, x_t) = \prod_{\ell=0}^{t} x_\ell^{r_{j,\ell}}$ with $r_{j,\ell} \geq 0$ and $\sum_{\ell=0}^{t} r_{j,\ell} = d$. For convenience, we define a d-dimensional vector $s_j = \bigotimes_{\ell=0}^{t} \ell^{\otimes r_{j,\ell}} \in [0,t]^d$. For example, if $t = 4, d = 8$ and $T_j(x_0, x_1, x_2, x_3, x_4) = 6x_1^2x_3^5x_4$, then $R_j(x_0, x_1, x_2, x_3, x_4) = x_0^0x_1^2x_2^0x_3^5x_4^1$ and thus $s_j = (1,1,3,3,3,3,3,4)^T \in [0,4]^8$. By $s_j[i]$ we denote the i'th element of vector s_j.

We now define $t+1$ matrices corresponding to term T_j:

$$X_{j,i} = \bigotimes_{\ell=0}^{i-1} J^{\otimes r_{j,\ell}} \otimes A_i^{\otimes r_{j,i}} \otimes \bigotimes_{\ell=i+1}^{d} J^{\otimes r_{j,\ell}},$$

where $0 \leq i \leq t$. The dimension of such matrices is $(t+3)^d \times (t+3)^d$ since each submatrix has dimension $(t+3) \times (t+3)$ and we take the d-fold Kronecker product. Similarly, we see that the row sum of each $X_{j,i}$ is 2^d since the row sum of each A_i and J is 2 and we take a d-fold Kronecker product. Clearly then, by the mixed product property (see Lemma 4):

$$X_{j,i}^k = \bigotimes_{\ell=0}^{i-1} (J^k)^{\otimes r_{j,\ell}} \otimes (A_i^k)^{\otimes r_{j,i}} \otimes \bigotimes_{\ell=i+1}^{d} (J^k)^{\otimes r_{j,\ell}},$$

for any $k \geq 0$. In the example when $r_{j,0} = 0$, $r_{j,1} = 2$, $r_{j,2} = 0$, $r_{j,3} = 5$ and $r_{j,4} = 1$, then $X_{j,3} = J^{\otimes 0} \otimes J^{\otimes 2} \otimes J^{\otimes 0} \otimes A_i^{\otimes 5} \otimes J^{\otimes 1} = J^{\otimes 2} \otimes A_i^{\otimes 5} \otimes J$. We then see that $X_{j,3}^k = (J^k)^{\otimes 2} \otimes (A_3^k)^{\otimes 5} \otimes J^k$.

Now, we see that:

$$X_{j,0}^{x_0} X_{j,1}^{x_1} \cdots X_{j,t}^{x_t} \quad = \quad \prod_{i=0}^{t} \left(\bigotimes_{\ell=0}^{i-1} (J^{x_i})^{\otimes r_{j,\ell}} \otimes (A_i^{x_i})^{\otimes r_{j,i}} \otimes \bigotimes_{\ell=i+1}^{d} (J^{x_i})^{\otimes r_{j,\ell}} \right) \tag{4}$$

$$\quad = \quad \bigotimes_{\ell=0}^{d} \left(D_{\ell,0}^{x_0} D_{\ell,1}^{x_1} \cdots D_{\ell,t}^{x_t} \right), \tag{5}$$

where $D_{\ell,i} \in \{J, A_i\}$ for $0 \leq i \leq t$. The derivation of Eqn (5) from Eqn (4) follows by the mixed product property of the Kronecker product (Lemma 4). For each product $D_{\ell,0}^{x_0} D_{\ell,1}^{x_1} \cdots D_{\ell,t}^{x_t}$, we see that $D_{\ell,s_j[\ell]} = A_{s_j[\ell]}$ and $D_{\ell,j} = J$ for all $0 \leq j \leq d$ with $j \neq s_j[\ell]$. As discussed earlier, matrices A_i and J commute, for any $0 \leq i \leq t$ and thus we may rewrite (5) as:

$$X_{j,0}^{x_0} X_{j,1}^{x_1} \cdots X_{j,t}^{x_t} = \bigotimes_{\ell=0}^{d} \left(A_{s_j[\ell]}^{x_{s_j[\ell]}} \otimes J^{\overline{x_{s_j[\ell]}}} \right), \text{ where } \overline{x_{s_j[\ell]}} = \sum_{\substack{0 \leq q \leq d \\ q \neq s_j[l]}} x_q \tag{6}$$

By Lemma 9, we see that some element of $X_{j,0}^{x_0} X_{j,1}^{x_1} \cdots X_{j,t}^{k_t}$ is thus equal to $R_j(x_0, x_1, \ldots, x_t)$ as required, since there is an element on the superdiagonal of $A_{s_j[\ell]}^{x_{s_j[\ell]}}$ equal to $x_{s_j[\ell]}$ for each $0 \leq \ell \leq d$. Let us assume that $R_j(x_0, x_1, \ldots, x_t)$ appears at row i_1 and column i_2. Now, we may define a vector $u'_j = c_j e_{i_1}$ and $v'_j = e_{i_2}$ where c_j is the coefficient of term T_j as in Eqn (3) and $e_{i_1}, e_{i_2} \in \mathbb{Z}^{(t+3)^d}$ are basis vectors. We may now see that

$$(u'_j)^T X_{j,0}^{x_0} X_{j,1}^{x_1} \cdots X_{j,t}^{x_t} v'_j = c_j R_j(x_0, x_1, \ldots, x_t) = T_j(x_0, x_1, \ldots, x_t) \tag{7}$$

In order to derive the sum of the r such terms $\sum_{j=1}^{r} T_j(x_0, x_1, \ldots, x_t)$, we will utilise the *direct sum*. For $0 \leq \ell \leq d$, we define Y'_ℓ by:

$$Y'_\ell = \bigoplus_{j=1}^{r} X_{j,\ell} \in \mathbb{N}^{r(t+3)^d \times r(t+3)^d}$$

We shall now modify each Y'_ℓ so that they are row stochastic. We recall that the row sum of each A_ℓ and J is 2. Therefore, the row sum of each $X_{j,\ell}$ is 2^d, since $X_{j,\ell}$ is a d-fold Kronecker product of A_i and J matrices. Then the row sum of each Y'_ℓ is also 2^d since direct sums do not modify the row sum. We thus see that $Y_\ell = \frac{1}{2^d} Y'_\ell$ is row stochastic.

We now consider the coefficients of each term. We previously multiplied each initial vector u_j by c_j and we may consider taking the Kronecker sum of each u_j before normalising the resulting vector (normalising according to L^1 norm). We face an issue however, since some coefficients c_j may be negative and thus the resulting vector is not stochastic (it must be nonnegative). Fortunately we may modify a technique utilised by Bertoni [5] to solve this issue. Given a PFA for which $u^T X v = \lambda \in [0, 1]$, then by defining $v' = \mathbf{1} - v$ where $\mathbf{1}$ is the all-one vector of appropriate dimension (i.e. swapping between final and non final states), then $u^T X v = 1 - \lambda \in [0, 1]$.

Now, since each $X_{j,\ell}$ has a row sum of 2^d and u'_j is of unit length (L^1 norm), then Eqn. (7) can be adapted to the following:

$$\begin{aligned} (u'_j)^T X_{j,0}^{x_0} X_{j,1}^{x_1} \cdots X_{j,t}^{x_t} (\mathbf{1} - v'_j) &= 2^{d(x_0+x_1+\ldots+x_t)} - c_j R_j(x_0, x_1, \ldots, x_t) \\ &= 2^{d(x_0+x_1+\ldots+x_t)} - T_j(x_0, x_1, \ldots, x_t) \end{aligned} \tag{8}$$

Let us assume, without loss of generality, that we have arranged the terms of P^h such that those terms with a positive coefficient (positive terms) appear first, followed by those with a negative coefficient (negative terms). Since we have r terms in P^h, there exists some $1 \leq r' < r$ such that we have r' postive and $r - r'$ negative terms. Let us define $u_j = |c_j| e_{i_1}$, which is similar to u'_j defined previously, but using the absolute value of the corresponding coefficient.

We define $v = \bigoplus_{j=1}^{r'} v_j \oplus \bigoplus_{j=r'+1}^{r} (\mathbf{1} - v_j) \in \{0, 1\}^{r(t+3)^d}$ as the final vector, so that we take the Kronecker sum of all final vectors, but we swap final and non-final states for the negative terms.

We now define the initial vector u, which must be a probability distribution. Let $g = \sum_{j=1}^{r} |c_j|$ be the sum of absolute values of coefficients and define $u = \frac{1}{g} \bigoplus_{j=1}^{r} u_j \in [0,1]^{r(t+3)^d}$. Note that u is stochastic (a probability distribution).

We now see that:

$$u^T Y_0 Y_1^{a_1} \cdots Y_t^{a_t} v \tag{9}$$

$$= \frac{\sum_{j=1}^{r'} u_j \left(\bigotimes_{\ell=0}^{d} A_{s_j[\ell]}^{x_{s_j[\ell]}} \otimes J^{\overline{x_{s_j[\ell]}}} \right) v_j + \sum_{j=r'+1}^{r} u_j \left(\bigotimes_{\ell=0}^{d} A_{s_j[\ell]}^{x_{s_j[\ell]}} \otimes J^{\overline{x_{s_j[\ell]}}} \right) (\mathbf{1} - v_j)}{g 2^{d(1+a_1+\cdots+a_t)}}$$

Here we used the definition of matrices Y_i and Eqn. (6) to rewrite the expressions for $X_{j,0} \cdots X_{j,t}$. Notice that the power of Y_0 is set at 1, since that constraint is required by the conversion from a standard Diophantine polynomial to a homogeneous one as explained previously. Now, using Eqn. (7) and Eqn. (8), we can rewrite Eqn. (9) as:

$$\frac{\sum_{j=1}^{r'} T_j(x_0, \ldots, x_t) + \sum_{j=r'+1}^{r} \left(2^{d(1+a_1+\ldots+a_t)} - |T_j(x_0, \ldots, x_t)| \right)}{g 2^{d(1+a_1+\cdots+a_t)}} \tag{10}$$

$$= \frac{(r - r' + 1)}{g} + \frac{\sum_{j=1}^{r'} T_j(x_0, \ldots, x_t) + \sum_{j=r'}^{r} T_j(x_0, \ldots, x_t)}{g 2^{d(1+a_1+\cdots+a_t)}} \tag{11}$$

$$= \frac{(r - r' + 1)}{g} + \frac{P^h(x_0, x_1, \ldots, x_t)}{g 2^{d(1+a_1+\cdots+a_t)}} \tag{12}$$

We therefore define $\mathcal{P} = (u, \{Y_a | a \in \Sigma_t\}, v)$ and $\Sigma_t = \{0, 1, \ldots, t\}$ as our PFA, with letter monotonic language $\mathcal{L} = 01^*2^* \cdots t^*$ and $\lambda = \frac{r-r'+1}{g} \in [0,1] \cap \mathbb{Q}$ as the cut-point. There exists some word $w = 01^{x_1}2^{x_2} \cdots t^{x_t} \in \mathcal{L}$ such that $f_{\mathcal{P}}(w) \leq \lambda$ if and only if $P^h(1, x_1, x_2, \ldots, x_t) = 0$. Therefore the strict emptiness problem for $\mathcal{P}$ is undecidable on letter monotonic languages. Since $\mathcal{P}$ is upper-triangular, then it is polynomially ambiguous. We note the surprising fact that all generator matrices are in fact *commutative* (each $X_{j,i}$ is commutative and direct sums do not affect commutativity), which leads to the undecidability of non-strict cut-points for polynomially ambiguous PFA defined over commutative matrices. In this case, the order of the input word in irrelevant, only the Parikh vector of alphabet letters is important. In fact we may redefine $u = uY_0$ and $\mathcal{L} = 1^*2^* \cdots t^*$ to remove Y_0 and all constraints on $\mathcal{L}$. ◀

4 Injectivity problems for polynomially ambiguous PFA

We now study the injectivity of acceptance probabilities of polynomially ambiguous PFA. The next result begins with a proof technique from [4], where the undecidability of the injectivity problem (called the freeness problem in [4], although we here rename it injectivity) was shown for exponentially ambiguous PFA over five states. We show that the injectivity problem remains undecidable even when the PFA is polynomially ambiguous and over four states by using our new encoding technique (avoiding the Turakainen procedure which increases the matrix dimensions by two and generates an exponentially ambiguous PFA).

4.1 Proof of Theorem 2

Proof. Let $\Sigma = \{x_1, x_2, \ldots, x_{n-2}\}$ and $\Delta = \{x_{n-1}, x_n\}$ be distinct alphabets and $h, g : \Sigma^* \to \Delta^*$ be an instance of the mixed modification PCP. The naming convention will become apparent below. We define two injective mappings $\alpha, \beta : (\Sigma \cup \Delta)^* \to \mathbb{Q}$ by:

$$\alpha(x_{i_1} x_{i_2} \cdots x_{i_m}) = \Sigma_{j=1}^{m} i_j (n+1)^{j-1},$$
$$\beta(x_{i_1} x_{i_2} \cdots x_{i_m}) = \Sigma_{j=1}^{m} i_j (n+1)^{-j},$$

and $\alpha(\varepsilon) = \beta(\varepsilon) = 0$. Thus α represents $x_{i_1}x_{i_2}\cdots x_{i_m}$ as a reverse $(n+1)$-adic number and β represents $x_{i_1}x_{i_2}\cdots x_{i_m}$ as a fractional number $(0.x_{i_1}x_{i_2}\cdots x_{i_m})_{(n+1)}$ (e.g. if $n = 9$, then $x_1x_2x_3$ is represented as $\alpha(x_1x_2x_3) = 321_{10}$ and $\beta(x_1x_2x_3) = 0.123_{10}$, where subscript 10 denotes base 10). Note that $\forall w \in (\Sigma \cup \Delta)^*, \alpha(w) \in \mathbb{N}$ and $\beta(w) \in [0,1) \cap \mathbb{Q}$. It is not difficult to see that $\forall w_1, w_2 \in (\Sigma \cup \Delta)^*, (n+1)^{|w_1|}\alpha(w_2) + \alpha(w_1) = \alpha(w_1w_2)$ and $(n+1)^{-|w_1|}\beta(w_2) + \beta(w_1) = \beta(w_1w_2)$.

Define $\gamma'' : (\Sigma \cup \Delta)^* \times (\Sigma \cup \Delta)^* \to \mathbb{Q}^{3\times 3}$ by

$$\gamma''(u,v) = \begin{pmatrix} (n+1)^{|u|} & 0 & \alpha(u) \\ 0 & (n+1)^{-|v|} & \beta(v) \\ 0 & 0 & 1 \end{pmatrix}.$$

It is easy to verify that $\gamma''(u_1,v_1)\gamma''(u_2,v_2) = \gamma''(u_1u_2, v_1v_2)$, i.e., γ'' is a homomorphism.

Let $\mathcal{G}'' = \{\gamma''(x_i, g(x_i)), \gamma''(x_i, h(x_i)) | x_i \in \Sigma, 1 \leq i \leq n-2\}$, $\mathcal{S}'' = \langle \mathcal{G}'' \rangle$, $\rho'' = (1,1,0)^T$ and $\tau'' = (0,0,1)^T$. Assume that there exist $M_1 = G_{i_1}G_{i_2}\cdots G_{i_t} \in \langle \mathcal{G}'' \rangle$ and $M_2 = G_{j_1}G_{j_2}\cdots G_{j_{t'}} \in \langle \mathcal{G}'' \rangle$ such that $t \neq t'$ or else at least one $G_{i_p} \neq G_{j_p}$ where $1 \leq p \leq t$ and $\lambda = \rho''^T M_1 \tau'' = \rho''^T M_2 \tau''$. We see that:

$$\lambda = \rho''^T M_1 \tau'' = \alpha(x_{i_1}x_{i_2}\cdots x_{i_t}) + \beta(f_1(x_{i_1})f_2(x_{i_2})\cdots f_t(x_{i_t})),$$
$$\lambda = \rho''^T M_2 \tau'' = \alpha(x_{j_1}x_{j_2}\cdots x_{j_{t'}}) + \beta(f'_1(x_{j_1})f'_2(x_{j_2})\cdots f'_{t'}(x_{j_{t'}})),$$

where each $f_i, f'_i \in \{g, h\}$. Since $\alpha(w) \in \mathbb{N}$ and $\beta(w) \in (0,1) \cap \mathbb{Q}$, $\forall w \in (\Sigma \cup \Delta)^*$, injectivity of α and β implies that if $\rho''^T M_1 \tau'' = \rho''^T M_2 \tau''$, then $t = t'$ and $i_k = j_k$ for $1 \leq k \leq t$. Furthermore, if $\rho^T M_1 \tau = \rho^T M_2 \tau$, we have that $\beta(f_1(x_{i_1})f_2(x_{i_2})\cdots f_t(x_{i_t})) = \beta(f'_1(x_{i_1})f'_2(x_{i_2})\cdots f'_t(x_{i_t}))$ and since at least one $f_p \neq f'_p$ for $1 \leq p \leq t$ by our above assumption, then this corresponds to a correct solution to the MMPCP instance (h, g). On the other hand, if there does not exist a solution to (h, g), then $\beta(f_1(x_{i_1})f_2(x_{i_2})\cdots f_t(x_{i_t})) \neq \beta(f'_1(x_{i_1})f'_2(x_{i_2})\cdots f'_t(x_{i_t}))$, and injectivity of β implies that $\rho''^T M_1 \tau'' \neq \rho''^T M_2 \tau''$.

We now use our new technique to encode such matrices and vectors to a linearly ambiguous four state PFA. We first define a mapping $\gamma' : (\Sigma \cup \Delta)^* \times (\Sigma \cup \Delta)^* \to \mathbb{N}^{3\times 3}$ to make all matrices be nonnegative integral:

$$\gamma'(u,v) \quad = \quad (n+1)^{|v|}\gamma''(u,v) = \begin{pmatrix} (n+1)^{|u|+|v|} & 0 & (n+1)^{|v|}\alpha(u) \\ 0 & 1 & (n+1)^{|v|}\beta(v) \\ 0 & 0 & (n+1)^{|v|} \end{pmatrix} \in \mathbb{N}^{3\times 3}$$

We next define the following morphism $\gamma : (\Sigma \cup \Delta)^* \times (\Sigma \cup \Delta)^* \to \mathbb{Q}^{4\times 4}$ to make all such matrices be row stochastic:

$$\gamma(u,v) = (n+1)^{-k} \begin{pmatrix} (n+1)^{|u|+|v|} & 0 & (n+1)^{|v|}\alpha(u) & \delta_1 \\ 0 & 1 & (n+1)^{|v|}\beta(v) & \delta_2 \\ 0 & 0 & (n+1)^{|v|} & \delta_3 \\ 0 & 0 & 0 & \delta_4 \end{pmatrix},$$

where $\delta_j \in \mathbb{N}$ are chosen so that the row sum of each row of $\gamma(u,v)$ is $(n+1)^k$ for some k. Any sufficiently large k can be used so long as each row has the same sum $(n+1)^k$ and thus $\gamma(u,v)$ becomes row stochastic. We use the same k value for all matrices of $\mathcal{G}$ which we define as $\mathcal{G} = \{\gamma(x_i, g(x_i)), \gamma(x_i, h(x_i)) | x_i \in \Sigma, 1 \leq i \leq n-2\}$, so that $\mathcal{S} = \langle \mathcal{G} \rangle$, and finally $\rho = (1,1,0,0)^T$ and $\tau = (0,0,1,0)^T$ are the initial and final state vectors respectively.

Assume that there exist $M_1 = G_{i_1} \cdots G_{i_t} \in \langle \mathcal{G} \rangle$ and $M_2 = G_{j_1} \cdots G_{j_{t'}} \in \langle \mathcal{G} \rangle$ such that $t \neq t'$ or else at least one $G_{i_p} \neq G_{j_p}$ for $1 \leq p \leq t$ and $\lambda = \rho^T M_1 \tau = \rho^T M_2 \tau$. We see that:

$$\lambda = \rho^T M_1 \tau = (n+1)^{-kt} \left(\alpha(x_{i_1} x_{i_2} \cdots x_{i_t}) + \beta(f_1(x_{i_1}) f_2(x_{i_2}) \cdots f_t(x_{i_t}))\right),$$
$$\lambda = \rho^T M_2 \tau = (n+1)^{-kt'} \left(\alpha(x_{j_1} x_{j_2} \cdots x_{j_{t'}}) + \beta(f'_1(x_{j_1}) f'_2(x_{j_2}) \cdots f'_{t'}(x_{j_{t'}}))\right),$$

where each $f_i, f'_i \in \{g, h\}$. If $t = t'$, then the same argument as previously shows that $i_k = j_k$ for $1 \leq k \leq t$. If $t \neq t'$, assume without loss of generality that $t' < t$. In this case we see that:

$$(n+1)^{-kt''} \left(\alpha(x_{i_1} \cdots x_{i_t}) + \beta(f_1(x_{i_1}) \cdots f_t(x_{i_t}))\right) = \alpha(x_{j_1} \cdots x_{j_{t'}}) + \beta(f'_1(x_{j_1}) \cdots f'_{t'}(x_{j_{t'}})),$$

where $t'' = t - t'$. This is a contradiction however since the number of nonzero digits (where a digit is understood base $(n+1)$ here) in the left hand side of this expression is exactly $2t$, and the number of digits in the right expression is $2t' < 2t$. Note that the multiplication by $(n+1)^{-kt''}$ does not alter the number of nonzero digits, it is only a right shift of all digits, kt'' times. Thus, since the left and right sides have a different number of nonzero digits they cannot be equal and thus $t = t'$ as required. ◀

4.2 Proof of Theorem 3

Proof. We use a reduction from the equal subset sum problem, defined thus: given a set of positive integers $S = \{x_1, x_2, \ldots, x_k\} \subseteq \mathbb{N}$, do there exist two disjoint nonempty subsets $S_1, S_2 \subseteq S$ such that $\sum_{\ell \in S_1} \ell = \sum_{m \in S_2} m$? This problem is known to be NP-complete [32]. Note that although there is a requirement that the sets S_1 and S_2 be disjoint, this is not crucial so long as $S_1 \neq S_2$ (since if some element x_j is in both S_1, S_2, then the equality also holds when x_j is removed from both sets). We may therefore require that $S_1 \neq S_2$, with both nonempty such that the sum of elements of each set is identical. We define the set of matrices $M = \{A_i, B_i | 1 \leq i \leq k\} \subseteq \mathbb{Q}^{3\times 3}$ in the following way:

$$A_i = \frac{1}{x_i+1} \begin{pmatrix} 1 & x_i & 0 \\ 0 & 1 & x_i \\ 0 & 0 & x_i+1 \end{pmatrix}, \quad B_i = \frac{1}{x_i+1} \begin{pmatrix} 1 & 0 & x_i \\ 0 & 1 & x_i \\ 0 & 0 & x_i+1 \end{pmatrix}$$

Note that A_i and B_i are thus row stochastic. Let $u = (1,0,0)^T$ be the initial probability distribution, $v = (0,1,0)^T$ be the final state vector and let $\mathcal{P} = (u, \{A_i, B_i\}, v)$ be our PFA. Define letter monotonic language $\mathcal{L} = (a_1|b_1)(a_2|b_2)\cdots(a_k|b_k) \subseteq a_1^* b_1^* a_2^* b_2^* \cdots a_k^* b_k^*$ and define a morphism $\varphi : \{a_i, b_i | 1 \leq i \leq k\}^* \to \{A_i, B_i | 1 \leq i \leq k\}^*$ in the natural way (e.g. the morphism induced by $\varphi(a_i) = A_i$ and $\varphi(b_i) = B_i$). Now, for a word $w = w_1 w_2 \cdots w_k \in \mathcal{L}$, note that $w_j \in \{a_j, b_j\}$ for $1 \leq j \leq k$. Define that $\mathfrak{v}(a_i) = x_i$ and $\mathfrak{v}(b_i) = 0$. In this case, we see that (due to the structure of A_i and B_i)

$$u^T \varphi(w_1 w_2 \cdots w_k) v = \frac{1}{\sum_{j=1}^k (x_j + 1)} \sum_{\ell=1}^k \mathfrak{v}(w_\ell)$$

Note of course that the factor $\frac{1}{\sum_{j=1}^k (x_j+1)}$ is the same for any $w \in \mathcal{L}$.

Assume then that there exists two words $\alpha, \beta \in \mathcal{L}$ with $\alpha \neq \beta$ such that $u^T \varphi(\alpha) v = u^T \varphi(\beta) v$ (i.e. assume that $\mathcal{P}$ is not free). Then $\sum_{\ell=1}^k \mathfrak{v}(\alpha_\ell) = \sum_{i \in S_1}^k x_i = \sum_{i \in S_2}^k x_i = \sum_{\ell=1}^k \mathfrak{v}(\beta_\ell)$, where $S_1 = \{x_i; |\alpha|_{a_i} > 0\}$ and $S_2 = \{x_i; |\beta|_{a_i} > 0\}$. This is true if and only if the instance S of the equal subset sum problem has a solution as required (note that only the empty set has a sum of zero which has unique representation $b_1 \cdots b_k$). Since A_i and B_i are upper-triangular, with initial state 1 and final state 2, then $\mathcal{P}$ is linearly ambiguous. ◀

5 Conclusion

There are a variety of open problems remaining. For example, does Theorem 1 still hold for quadratic ambiguity, when taken alongside the other constraints (letter monotonic language and commutative matrices). Another direction is to improve the complexity lower bound of Theorem 3 to show it is either PSPACE-hard, EXPSPACE-hard or undecidable, under the same constraints as in the theorem statement.

References

1 L. Babai, R. Beals, J-Y. Cai, G. Ivanyos, and E. M. Luks. Multiplicative equations over commuting matrices. In *Proc. of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA 96, 1996.

2 L. Babai and S. Moran. Arthur–Merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36:254–276, 1988.

3 P. C. Bell, S. Chen, and L. M. Jackson. Scalar ambiguity and freeness in matrix semigroups over bounded languages. In *Language and Automata Theory and Applications*, volume LNCS 9618, pages 493–505, 2016.

4 P. C. Bell, V. Halava, and M. Hirvensalo. Decision Problems for Probabilistic Finite Automata on Bounded Languages. *Fundamenta Informaticae*, 123(1):1–14, 2012.

5 A. Bertoni. The solution of problems relative to probabilistic automata in the frame of the formal language theory. *GI Jahrestagung*, pages 107–112, 1974.

6 A. Bertoni, G. Mauri, and M. Torelli. Some recursively unsolvable problems relating to isolated cutpoints in probabilistic automata. In *Automata, Languages and Programming*, volume 52, pages 87–94, 1977.

7 V. Blondel and V. Canterini. Undecidable problems for probabilistic automata of fixed dimension. *Theory of Computing Systems*, 36:231–245, 2003.

8 V. Blondel and J. Tsitsiklis. The boundedness of all products of a pair of matrices is undecidable. *Systems and Control Letters, Elsevier*, 41:2:135–140, 2000.

9 V. Blondel and J. N. Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36:1249–1274, 2000.

10 R. G. Bukharaev. Probabilistic automata. *Journal of Mathematical Sciences*, 13(3):359–386, 1980.

11 J. Buresh-Oppenheim, M. Clegg, R. Impagliazzo, and T. Pitassi. Homogenization and the polynomial calculus. *Computational complexity*, 11(3-4):91–108, 2002.

12 J. Cassaigne, J. Karhumäki, and T. Harju. On the Decidability of the Freeness of Matrix Semigroups. *International Journal of Algebra and Computation*, 9(3-4):295–305, 1999.

13 É. Charlier and J. Honkala. The freeness problem over matrix semigroups and bounded languages. *Information and Computation*, 237:243–256, 2014.

14 V. Chonev, J. Ouaknine, and J. Worrell. On the Complexity of the Orbit Problem. *Journal of the ACM*, 63(3):1–18, 2016.

15 A. Condon and R. J. Lipton. On the complexity of space bounded interactive proofs. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 462–467, 1989.

16 L. Daviaud, M. Jurdzinski, R. Lazic, F. Mazowiecki, G. A. Pérez, and J. Worrell. When is containment decidable for probabilistic automata? In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 121:1–121:14, 2018.

17 N. Fijalkow, C. Riveros, and J. Worrell. Probabilistic automata of bounded ambiguity. In *28th International Conference on Concurrency Theory (CONCUR)*, pages 19:1–19:14, 2017.

18 V. Halava, J. Kari, and Y. Matiyasevich. On post correspondence problem for letter monotonic languages. *Theoretical Computer Science*, 410:30–32, 2009.

19 M. Hirvensalo. Improved undecidability results on the emptiness problem of probabilistic and quantum cut-point languages. *SOFSEM 2007: Theory and Practice of Computer Science, Lecture Notes in Computer Science*, 4362:309–319, 2007.

20 R. A. Horn and C. R. Johnson. *Topics in matrix analysis*. Cambridge University Press, 1991.

21 O. Ibarra and B. Ravikumar. On sparseness, ambiguity and other decision problems for acceptors and transducers. In *Proc. STACS 1986*, volume 210, pages 171–179, 1986.

22 R. Kannan and R. J. Lipton. Polynomial-time algorithm for the orbit problem. *Journal of the ACM*, 33(4):808–821, 1986.

23 Yu. Matiyasevich. *Hilbert's Tenth Problem*. MIT Press, 1993.

24 M. Mohri, F. Pereira, and M. Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, 2002.

25 T. Neary. Undecidability in Binary Tag Systems and the Post Correspondence Problem for Five Pairs of Words. In *STACS15*, pages 649–661, 2015.

26 A. Paz. *Introduction to Probabilistic Automata*. Academic Press, 1971.

27 M. O. Rabin. Probabilistic automata. *Information and Control*, 6:230–245, 1963.

28 C. Reutenauer. *Propertiétés arithmétiques et topologiques de séries rationnelles en variables non commutatives*. Thèse troisième cycle, Université Paris VI, 1977.

29 A. Salomaa and M. Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Springer-Verlag, 1978.

30 P. Turakainen. Generalized automata and stochastic languages. *Proceedings of the American Mathematical Society*, 21:303–309, 1969.

31 A. Weber and H. Seidl. On the degree of ambiguity of finite automata. *Theoretical Computer Science*, 88(2):325–349, 1991.

32 H. J. Woeginger and Z. Yu. On the equal-subset-sum problem. *Information Processing Letters*, 42(6):299–302, 1992.

33 A. Yakaryilmaz and A. C. Say. Unbounded-error quantum computation with small space bounds. *Information and Computation*, 209(6):873–892, 2011.

String-to-String Interpretations With Polynomial-Size Output

Mikołaj Bojańczyk
Institute of Informatics, University of Warsaw, Poland
bojan@mimuw.edu.pl

Sandra Kiefer
Department of Computer Science, RWTH Aachen University, Germany
kiefer@cs.rwth-aachen.de

Nathan Lhote
Institute of Informatics, University of Warsaw, Poland
nlhote@mimuw.edu.pl

Abstract

String-to-string MSO interpretations are like Courcelle's MSO transductions, except that a single output position can be represented using a tuple of input positions instead of just a single input position. In particular, the output length is polynomial in the input length, as opposed to MSO transductions, which have output of linear length. We show that string-to-string MSO interpretations are exactly the polyregular functions. The latter class has various characterisations, one of which is that it consists of the string-to-string functions recognised by pebble transducers.

Our main result implies the surprising fact that string-to-string MSO interpretations are closed under composition.

2012 ACM Subject Classification Theory of computation → Transducers

Keywords and phrases MSO, interpretations, pebble transducers, polyregular functions

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.106

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version `https://arxiv.org/abs/1905.13190`, [6]

Acknowledgements The authors would like to thank Benedikt Brütsch for helpful discussions on the topic.

1 Introduction

A string-to-string function is called *regular* if it is computed by a deterministic two-way automaton with output. There are many equivalent models for the same class of functions: string-to-string MSO transductions [11], streaming string transducers [1], and various kinds of combinator-based formalisms [2, 5, 9].

A deterministic two-way automaton can visit each input position at most once in each state, otherwise it would loop forever. This means that the length of the run – and also the size of the output word – is linear in the input string. One way to go beyond linear-sized outputs was proposed by Milo, Suciu and Vianu [17], following earlier work by Globerman and Harel [12]: equip the automaton with k pebbles which can be used to mark positions in the input word. To avoid making the model Turing-powerful, the pebbles are required to observe a so-called stack discipline: they are organised in a stack, and only the top-most pebble can be moved. In [3], it is shown that pebble transducers are equivalent to multiple other models: a higher-order functional programming language [3, Section 4], an imperative programming language with for-loops [3, Section 3], combinators [3, end of Section 4], and

compositions of certain simple atomic functions [3, Section 1]. Because of the multitude of models and their polynomial-size outputs, the class of functions recognised by these models is called *polyregular functions*.

The list of models for polyregular functions described in [3] does not include any logical model. In this paper, we fix that omission. As mentioned above, for the regular functions, which have linear-size output, the logical model consists in string-to-string MSO transductions. In an MSO transduction, each position of the output string is interpreted as a single position of the input string. A natural idea to capture polyregular functions is to consider what we call *string-to-string* MSO *interpretations*, where a position of the output string is represented by a k-tuple of positions in the input string. At first glance, this idea looks suspicious: if string-to-string MSO interpretations were equivalent to polyregular functions, then they would be closed under composition, because the class of polyregular functions is. However, composing two string-to-string MSO interpretations

$$\Sigma^* \xrightarrow{f} \Gamma^* \xrightarrow{g} \Delta^*$$

raises the following issue. Suppose that positions of the intermediate word in Γ^* are represented by k-tuples of positions in the input word from Σ^*. If an MSO formula defining g quantifies over a set of positions in the intermediate word to define a property of the output word in Δ^*, then this corresponds to quantifying over a set of k-tuples of positions in the input word. If we assume dimension $k = 1$, then the problem dissolves, and this is why MSO transductions have dimension $k = 1$, whereas dimension $k > 1$ is never used in the context of MSO (as opposed to first-order logic, where the standard notion of transformation, i.e. first-order interpretation, uses higher dimensions).

As our main result, we show that the problems discussed above only invalidate the natural construction for composing MSO interpretations, which uses substitutions of formulas. Still, and surprisingly, for structures that represent strings there exists a (less natural) construction. This follows from our main result, which states that polyregular functions are exactly the string-to-string MSO interpretations. Indeed, corollaries of the main result are that (a) string-to-string MSO interpretations are closed under composition; and (b) for every regular string language, its inverse image under a string-to-string MSO interpretation is also regular. This is because (a) and (b) are true for polyregular functions. Proving (a) and (b) directly for string-to-string MSO interpretations seems hard; in fact an understandable (but wrong) first reaction to the claims (a) and (b) would be that they are false, for the reasons discussed in the previous paragraph.

It is easy to see that every polyregular function is a special case of a string-to-string MSO interpretation. One argument is that a k-pebble automaton can be simulated using a string-to-string MSO interpretation, by representing configurations of the pebble automaton using k-tuples of positions in the input word. The difficulty lies in proving the opposite direction and it comes from the stack discipline required in a pebble automaton. A k-tuple of positions used by an MSO interpretation can of course be viewed as a configuration of a pebble automaton, but there does not seem to be any reason why the resulting pebble automaton should observe stack discipline. It turns out – and this is the main technical insight of this paper – that any MSO formula which defines a linear ordering on k-tuples of positions in strings must necessarily observe an implicit stack discipline, which makes it possible to translate a string-to-string MSO interpretation into a pebble automaton.

Outline. After describing string-to-string MSO interpretations in Section 2, we revise polyregular functions via the formalism of for-programs in Section 3. In Section 4, we show that the models are equivalent.

Due to space limits, we omit some of the technical details and proofs in this article. For a full version, we defer the reader to [6]

2 Interpretations

In this section, we revise first-order and MSO interpretations, which are transformations of relational structures using formulas.

2.1 Logic and interpretations

Relational vocabularies and logic. A *(relational) vocabulary* is a set of relation names, each one associated with a natural number called its *arity*. For short, we refer to relational vocabularies simply as *vocabularies*. A *structure* over a vocabulary σ consists of a set called the *universe* and for each relation name of σ a corresponding relation of the same arity over the universe. To define properties of relational structures, we use monadic second-order logic and its first-order fragment with the usual syntax and semantics [19]. We use the convention that lower-case variables x, y, z range over elements and upper-case variables X, Y, Z range over sets of elements.

Interpretations. Intuitively speaking, an interpretation is a function from relational structures to relational structures where each element of the universe of the output structure is a tuple of elements of the input structure, and the relations of the output structure are defined using formulas evaluated over the input structure.

▶ **Definition 1** (Interpretations over general structures). *For $k \geq 1$, the syntax of a k-*dimensional first-order interpretation *consists of:*

1. *two vocabularies, called the* input vocabulary *and the* output vocabulary
2. *an* FO *formula over the input vocabulary with k free variables, called the* universe formula.
3. *for each n and each n-ary relation name R of the output vocabulary, an associated* FO *formula φ_R over the input vocabulary, with $k \cdot n$ free variables.*

MSO interpretations *are defined analogously, except that formulas of* MSO *are used, but the free variables still range over elements and not over sets.*

The semantics of an interpretation is a function from structures over the input vocabulary to structures over the output vocabulary, defined as follows.

- The universe of the output structure is the set of k-tuples of elements in the universe of the input structure which satisfy the universe formula from item 2 in Definition 1.
- An n-ary relation name R of the output vocabulary is interpreted as the set of n-tuples of k-tuples from the input structure for which (a) each k-tuple is in the output universe, and (b) the entire $(n \cdot k)$-tuple satisfies the formula φ_R in item 3 in Definition 1.

Composition. First-order interpretations are closed under composition [14, p. 218]. Let us recall the proof. Suppose that we want to compose interpretations

$$\text{structures over } \sigma_1 \xrightarrow{\mathcal{I}_1} \text{structures over } \sigma_2 \xrightarrow{\mathcal{I}_2} \text{structures over } \sigma_3$$

of dimensions k_1 and k_2, respectively. The $(k_1 \cdot k_2)$-dimensional composition is obtained from $\mathcal{I}_2$ as follows: (a) quantification over elements of $\mathcal{I}_2$ is replaced by a quantification over k_1-tuples of elements; and (b) relation names from σ_2 that appear in the input of $\mathcal{I}_2$ are replaced by the corresponding formulas from $\mathcal{I}_1$. This idea does not work for MSO in general,

since set quantification in $\mathcal{I}_2$ would need to be replaced by quantification over sets of k_1-tuples. It does work when $k_1 = 1$. This essentially corresponds to Courcelle's transductions, for which closure under composition follows naturally [8, Theorem 7.14]. To recover closure under composition for $k_1 \geq 2$, one can use (not necessarily monadic) second-order logic, which by Fagin's Theorem [16, Corollary 9.9] corresponds to the polynomial hierarchy of computational complexity and is outside the scope of this paper.

2.2 String-to-string interpretations

In this paper we are interested in interpretations that transform structures which represent strings. While there are two natural ways to model strings as relational structures, namely with an order relation or with a successor relation, only the order relation is useful in our context.

▶ **Definition 2** (String-to-string interpretations). *For a string $w \in \Sigma^*$, its* ordered model *is defined to be the following relational structure, denoted by $\underline{w}$:*

- *the universe consists of the positions in the string, i.e., natural numbers;*
- *there is a binary relation for the natural order on positions;*
- *for each $a \in \Sigma$ there is a unary relation which is satisfied by every position with label a.*

For alphabets Σ and Γ, a function $f\colon \Sigma^* \to \Gamma^*$ is called first-order string-to-string interpretation *if the corresponding transformation on ordered models is a first-order interpretation for strings with length at least two*[1]*. Likewise we define* MSO string-to-string interpretations.

▶ **Example 3.** Consider the function $f\colon \{a,b\}^* \to \{a,b\}^*$ which maps a word to the concatenation of all of its reversed prefixes, as in the following example (with prefixes grouped for better readability):

$$abbb \quad \mapsto \quad \underbrace{a}\,\underbrace{ba}\,\underbrace{bba}\,\underbrace{bbba}.$$

This transformation is the running example in [3]. We show that f can be seen as a string-to-string first-order interpretation. The dimension is 2, i.e. positions in the output word represent pairs of positions in the input word. A pair (x_1, x_2) of positions in the input word is used in the output word if it satisfies the universe formula $x_2 \leq x_1$. The idea is that x_1 represents the length of the prefix, while x_2 is the position in that prefix. The label of a position (x_1, x_2) is inherited from the second coordinate, as expressed by the formulas corresponding to labels on the output structure:

$$\varphi_a(x_1, x_2) = a(x_2) \quad \varphi_b(x_1, x_2) = b(x_2)$$

The order on the positions of the output word is defined by the formula

$$\varphi_{\leq}\Big(\underbrace{x_1, x_2}_{\substack{\text{a position of}\\ \text{the output word}}}, \underbrace{x_1', x_2'}_{\substack{\text{another position of}\\ \text{the output word}}}\Big) = (x_1 < x_1') \vee (x_1 = x_1' \wedge x_2 \geq x_2').$$

[1] A typical operation we want to model is string duplication. When the input length is at least two, one can represent additional copies of the input string using a higher dimension. For input length $n \leq 1$, the output length will be $n^k \leq 1$ regardless of the dimension k. Another solution to this issue would be to have duplication built into the definition of interpretations.

Note that the above formula defines the lexicographic ordering on pairs of positions, with the first coordinate being used in increasing order, and the second coordinate being used in decreasing order. This, as it will turn out, is not a coincidence, since our main technical result says that it is impossible to define a linear order on tuples of positions without implicitly using some kind of lexicographic ordering.

Successor instead of order. When modelling a string as a relational structure, we use the order on positions. An alternative solution would be to use just the successor relation. The difference between the two solutions is that it is harder to define an order on k-tuples of positions than it is to define a successor relation. It turns out that the difference is crucial, and functions that output strings with successor can be ill-behaved. Note that whether or not the input string is equipped with an order or a successor relation makes no difference, since the order on the positions of the input string can be recovered in MSO, which can compute the transitive closure of binary relations on positions.

Define the *successor model* of a string in the same way as the ordered model from Definition 2, except that a binary relation for successor is used instead of the order. Define a *successor*-MSO *string-to-string interpretation* to be a string-to-string function which is computed by an MSO interpretation, assuming that strings are represented by their successor models. Likewise, we define successor-first-order string-to-string interpretations. These are closed under composition, because first-order interpretations are closed under composition. On the other hand, successor-MSO string-to-string interpretations are not closed under composition and lead to undecidability, as summarised in the following theorem.

▶ **Theorem 4.**

1. *The class of successor-*MSO *string-to-string interpretations is not closed under composition, and strictly contains the class of (order-)*MSO *string-to-string interpretations.*
2. *The following is undecidable: given a successor-first-order string-to-string interpretation f and a regular language L over the output alphabet, decide if $f^{-1}(L)$ is nonempty.*

3 Polyregular functions

Here we describe the class of polyregular functions. It has several equivalent characterisations, see [3, Theorem 4.4], one of which consists in the afore-mentioned pebble transducers. For the purposes of this paper, it will be most convenient to use a slightly more abstract characterisation in terms of for-programs, a machine model for string-to-string functions. We just explain the formalism on short examples, for a more detailed description see [3].

```
for x in first..last
  for y in last..first
    if y≤x and a(y) then
      output a
    if y≤x and b(y) then
      output b
```

(a) A for-program for the function in Example 3.

```
for x in first..last
  var P : Bool
  for y in last..first
    if y≥x then
      P := not P
  if P and a(x) then
    output a
  if P and b(x) then
    output b
```

(b) A for-program with a Boolean variable `P`.

```
for y in first..last
  if x1≤y and y≤x2 and a(y)
      P := true
```

(c) A for-program which checks if there is an `a` between the positions `x1` and `x2`.

Figure 1 Example for-programs.

Most of the syntactic constructions that can be used in a for-program are illustrated in Figure 1a: (1) variables that range over positions in the input word; (2) for-loops in which a variable iterates over all positions in the input word in increasing or decreasing order; (3) if-statements which depend on the order/labels of variables; (4) instructions which output letters. Position variables cannot be declared or written to, they are implicitly declared by for-loops and their only updates are the iterations performed by the for-loops.

The only feature of for-programs that is not used in Figure 1a is (5) Boolean variables. Figure 1b shows a program that outputs only those letters in the input word which have even distance to the last position. In the program, the Boolean variable `P` is declared in the scope of a for-loop. On each iteration of the loop, the variable is reinitialised to false.

A for-program is called *first-order definable* if Boolean variables can only be updated from `false`, which is their initial value upon declaration, to `true`. In other words, the only allowed update for Boolean variables is `P := true`. For the first-order restriction, it is important that Boolean variables can be declared inside for-loops, and that they are reinitialised to `false` at each iteration of the loop that they are declared in. The reason for the name "first-order definable" is that one can define in first-order logic the reachability relation on program states of the for-program, see [3, Lemma 5.3].

▶ **Definition 5.** *A string-to-string function is called* polyregular *if it is computed by a for-program. It is called* first-order polyregular *if it is computed by a first-order definable for-program.*

The class of polyregular functions has other characterisations, including the string-to-string pebble transducers introduced by Milo, Suciu and Vianu [17], as well as a higher-order functional programming language [3, Section 4]. The main result of this paper, Theorem 7 in the next section, adds a logical characterisation, namely string-to-string MSO interpretations.

Evaluating first-order formulas. The for-programs described above take as input strings and also output strings. One can also consider for-programs which input a string with distinguished positions and which output a Boolean value, as in Figure 1c. The distinguished positions are represented by free variables (here `x1` and `x2`), while the output value is taken from some distinguished Boolean variable, here `P`.

▶ **Lemma 6.** *Let $\varphi(x_1, \dots, x_k)$ be an* FO *formula over strings. There is a first-order for-program which computes the following.*

- ***Input.*** *A word $w \in \Sigma^*$ and positions $x_1, \dots, x_k$ in w;*
- ***Output.*** *Yes or No, depending on whether w satisfies $\varphi(x_1, \dots, x_k)$.*

Proof. The for-program implements the semantics of an FO formula. For each quantifier, it loops over all possible values for the quantified position, and a Boolean variable is used to remember if some value has already been found which renders the formula true. ◀

A similar result is true for MSO formulas, but the proof for that statement uses automata.

4 Equivalence

We show that the models defined in Sections 2 and 3 are equivalent.

▶ **Theorem 7.**

1. *String-to-string* MSO *interpretations are exactly the polyregular functions.*
2. *First-order string-to-string interpretations are exactly the first-order polyregular functions.*

Since the class of polyregular functions is closed under composition[2], we obtain:

▶ **Corollary 8.** *String-to-string* MSO *interpretations are closed under composition.*

By using Theorem 7, the proof of the corollary passes through for-programs. We are not aware of any direct proof that does not exploit the equivalence to polyregular functions.

The rest of this paper is dedicated to the proof of Theorem 7. We begin with a reduction of the first to the second item. This reduction illustrates a general phenomenon, namely that results about first-order polyregular functions often imply results about general polyregular functions, despite the latter class being larger. The reason behind this phenomenon is the following lemma, which says that for every polyregular function, all of the behaviour that is not first-order definable can be pushed into a simple preprocessing step. Define a *rational function*, see [4, Section 13.2], to be a string-to-string function which is recognised by a nondeterministic automaton, where every transition is labelled by a pair consisting of a letter from the input alphabet and a string over the output alphabet, and which is unambiguous in the sense that every input string admits exactly one accepting run.

▶ **Lemma 9.**

1. *A function is polyregular if and only if it is a composition consisting of:*
   a. *a (letter-to-letter) rational function; followed by*
   b. *a first-order polyregular function.*
2. *A function is an* MSO *string-to-string interpretation if and only if it is a composition consisting of:*
   a. *a (letter-to-letter) rational function; followed by*
   b. *a first-order string-to-string interpretation.*

The proof of Lemma 9 can be found in [6]. It is based on ideas from [7, 15, 3] and uses factorisation forests. With the lemma, we show that item 2 in Theorem 7 implies item 1, i.e. if first-order string-to-string interpretations are exactly the first-order polyregular functions, then MSO interpretations are exactly the polyregular functions:

$$\begin{aligned} \text{polyregular} &= && \text{by item 1 of Lemma 9} \\ (\text{first-order polyregular}) \circ \text{rational} &= && \text{by item 2 of Theorem 7} \\ (\text{first-order interpretations}) \circ \text{rational} &= && \text{by item 2 of Lemma 9} \\ \text{MSO interpretations} & && \end{aligned}$$

It remains to prove item 2 in Theorem 7, i.e. that first-order string-to-string interpretations are exactly the first-order polyregular functions. The right-to-left inclusion follows immediately from [3, Lemma 5.3], which says that a formula in first-order logic can define the reachability relation on program states in first-order for-programs. We are left with the left-to-right-inclusion:

$$\text{first-order string-to-string interpretations} \subseteq \text{first-order definable for-programs} \qquad (1)$$

The rest of the paper is devoted to showing the above inclusion. When simulating a first-order interpretation by a for-program, we will mainly be concerned with the universe of the output string (which is a set of k-tuples of positions in the input string) and its ordering. The labelling of the k-tuples can then be recovered using the for-program from Lemma 6.

[2] Closure under composition was proved for pebble transducers in [10, Theorem 11] and for the class of for-programs in [3, Section 8.1] as a step in proving equivalence with the other models of polyregular functions.

The main result is that every first-order definable linear ordering on tuples of positions can be implemented by a for-program. To be able to speak about this result, we introduce some notation for devices that produce lists of tuples of positions.

Enumerators. Let $k \in \mathbb{N}$. A *k-enumerator* over an alphabet Σ is a function of the following form:

- **Input.** A string $w \in \Sigma^*$;
- **Output.** A list of k-tuples of positions in w, that is nonrepeating[3].

We compare the following two ways of implementing k-enumerators:

1. A k-enumerator is called *definable* if there are two FO formulas: one with k variables, which says when a tuple is part of the output list, and one with $2k$ variables, which defines a total order on the tuples selected by the first formula.
2. A k-enumerator is called *programmable* if its output can be computed by a first-order for-program that instead of outputting letters uses instructions of the form `output (x1,...,xk)` where `x1`, ..., `xk` are position variables.

For definable k-enumerators, the order on tuples in the output list is given explicitly by the formula φ, while in programmable ones, the order is implicit from the order in which the output instructions are executed during the computation.

▶ **Example 10.** We present an enumerator based on Example 3. Consider the 2-enumerator which outputs all pairs of positions (x_1, x_2) with $x_2 \leq x_1$, listed in lexicographic order, where x_1 is ordered in increasing order and x_2 is ordered in decreasing order. Here is an example:

$$abbb \quad \mapsto \quad (1,1),(2,2),(2,1),(3,3),(3,2),(3,1),(4,4),(4,3),(4,2),(4,1)$$

This enumerator is definable, as witnessed by the formula $\varphi_{\leq}$ in Example 3. The formula $\varphi_{\leq}$ is quantifier-free, but in general, quantifiers are allowed. Here is a for-program that computes the same function:

```
for x1 in first..last
  for x2 in last..first
    if x2 ≤ x1 then
      output (x1,x2)
```

The following lemma is the main technical result of this paper.

▶ **Lemma 11.** *Every definable k-enumerator is also programmable.*

Our proof of Lemma 11 uses two fundamental ingredients. The first is by now standard: this is Simon's factorisation forest theorem [18], which roughly says that every string can be cut into pieces that are similar to each other. The second ingredient is new: the Domination Lemma, presented in Section 4.1, roughly says that if a string is cut into pieces that are similar to each other, then any first-order definable linear order on tuples of positions must observe an implicit stack discipline. These two results are combined in Section 4.2 to prove Lemma 11. Before we proceed with the proof of Lemma 11, we use it to complete the proof of Theorem 7.

[3] Every tuple appears at most once, but positions can appear in multiple tuples. We need this for the existence of the formulas stated in the following definitions.

Proof of Theorem 7, second part. The only part of Theorem 7 that has not been proved yet is that every first-order string-to-string interpretation is polyregular. Suppose that f is a k-dimensional first-order string-to-string interpretation. Consider the k-enumerator that inputs a string w and outputs the list of k-tuples of positions in w used to represent output positions of $f(w)$, in the appropriate order. Apply Lemma 11 to obtain a first-order for-program g which computes the same list. To compute the original function f, we use a for-program which behaves as g, except that instead of outputting a k-tuple of positions like g, it uses the program described in Lemma 6 as a subroutine to check what is the output letter that should be produced for this tuple, and outputs that letter. ◀

4.1 The Domination Lemma

In this section we present the Domination Lemma, which says that if $\prec$ is a first-order definable linear order on k-tuples of positions in a string, then there is an implicit stack discipline in the following sense. For every type (see below) t of tuples of positions there is a coordinate $d \in \{1, \ldots, k\}$ such that for the subset of k-tuples of positions formed by all of type t, the order $\prec$ is uniquely determined by the order of the d-th coordinates in the string.

We begin by explaining the notions of types. For $r \in \{0, 1, \ldots\}$, the *rank r type* of a structure $\mathfrak{A}$ with k distinguished positions $\bar{x} \coloneqq (x_1, \ldots, x_k)$ is defined to be the set of first-order formulas of quantifier rank at most r and with k free variables that are true in $\mathfrak{A}, \bar{x}$. The number k is the *arity* of the type. For arity 0, we talk about the rank r *type of the structure* $\mathfrak{A}$. If the structure $\mathfrak{A}$ is implicit from the context, then we talk about the rank r type of the tuple $\bar{x}$. For every finite vocabulary, there are finitely many types of given arity and rank. We write $\equiv_r$ for the equivalence relation on structures with distinguished elements of having the same rank r type. For a binary relation R, its *inverse* is the set $\{(v, u) \mid (u, v) \in R\}$. For $p \in \{1, -1\}$, define R^p to be either R or its inverse, depending on the value of p.

▶ **Lemma 12** (Domination Lemma). *For all $k, m, r \in \{1, 2, \ldots\}$, there is an $\omega \in \{1, 2, \ldots\}$ with the following property. Let $n \in \{1, 2, \ldots\}$, let $w_1, \ldots, w_n$ be strings over some alphabet Σ and let $\mathfrak{A}$ be the ordered structure of the concatenation $w_1 \cdots w_n$ extended with the* block order *defined by*

$$x \sqsubset y \qquad \text{if} \qquad x \text{ is a position in } w_i \text{ and } y \text{ is a position in } w_j \text{ for some } i < j.$$

Let $\prec$ be a linear order on k-tuples in $\mathfrak{A}$ defined by a first-order formula of quantifier rank r, and let t be a k-ary rank ω type over the vocabulary of $\mathfrak{A}$. If

$$w_i \equiv_\omega w_{i+1} \qquad \text{holds for all } i \in \{1, \ldots, n-1\} \text{ with at most } m \text{ exceptions,}$$

then there is a $d \in \{1, \ldots, k\}$, called the dominating coordinate*, and a $p \in \{-1, 1\}$, called the* polarity*, such that*

$$x_d \sqsubset^p y_d \quad \text{implies} \quad (x_1, \ldots, x_k) \prec (y_1, \ldots, y_k) \qquad \text{for all } \underbrace{x_1, \ldots, x_k}_{\text{of type } t}, \underbrace{y_1, \ldots, y_k}_{\text{of type } t} \text{ in } \mathfrak{A}.$$

The Domination Lemma is the technical heart of this paper. The full proof can be found in [6]. To explain some of the ideas that we use, we treat a special case in detail. In the Domination Lemma, the structure $\mathfrak{A}$ consists of blocks organised in a linear way. A very simple linear order – although infinite – is the natural one on the rational numbers; one reason for its simplicity is that quantifiers can be eliminated (see [13, Section 5.6.2]). Because of this, it is quite easy to prove a version of the Domination Lemma for the rational numbers and still its proof bears some similarity to the proof of the general case.

▶ **Lemma 13** (Rational Domination Lemma). *Let $\prec$ be a linear ordering on k-tuples of rational numbers defined by a quantifier-free (equivalently, first-order) formula using only the usual ordering $<$ on rational numbers. Then there is a coordinate $d \in \{1, \ldots, k\}$ and a polarity $p \in \{-1, 1\}$ such that*

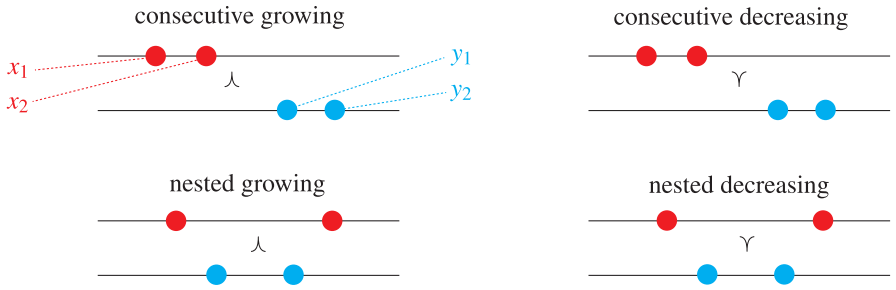
$$x_d <^p y_d \quad \textit{implies} \quad (x_1, \ldots, x_k) \prec (y_1, \ldots, y_k)$$

for all tuples of rational numbers satisfying $x_1 < \cdots < x_k$ and $y_1 < \cdots < y_k$.

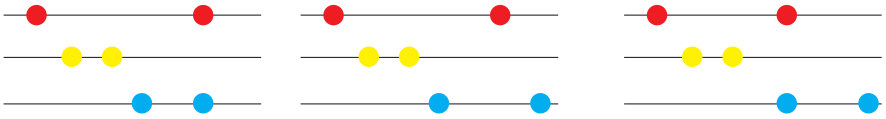
Proof. We first prove the statement for $k = 1$ and $k = 2$ and then we deduce the general case.

1. When $k = 1$, then the formula defining $\prec$ must be either $x < y$ or $x > y$.
2. For $k = 2$, we do a case analysis. Note that whether $\bar{x} \prec \bar{y}$ or $\bar{y} \prec \bar{x}$ holds depends only on the order relationship of the positions in $\bar{x}$ and $\bar{y}$ in the rational numbers and not on the precise values in $\bar{x}$ and $\bar{y}$.

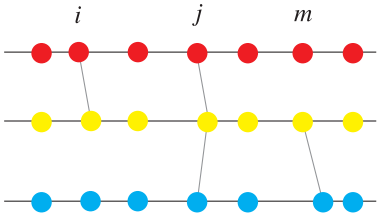
   The following picture shows the two possible relationships for two pairs $\bar{x}$ and $\bar{y}$ when they are "consecutive" and the two possible relationships when they are "nested":



   Suppose we are given a pair $\bar{x}$ and without loss of generality, assume the "consecutive growing" case for some second pair $\bar{y}$. We only show the proof for the case that there is a pair $\bar{y}'$ such that $\bar{x}$ and $\bar{y}$ are "nested growing" ("nested decreasing" works analogously). We prove that $d = 1$ is dominating for $\prec$ with polarity $p = 1$. Consider all three remaining configurations of pairs $\bar{x}$ and $\bar{y}$ with $x_1 < y_1$. In all cases, $\bar{x} \prec \bar{y}$ is proved by finding an intermediate pair (drawn in yellow), whose order with respect to $\bar{x}$ and $\bar{y}$ follows from the assumptions "consecutive/nested growing" (in the pictures below, we assume that lower lines represent bigger tuples in the ordering $\prec$):



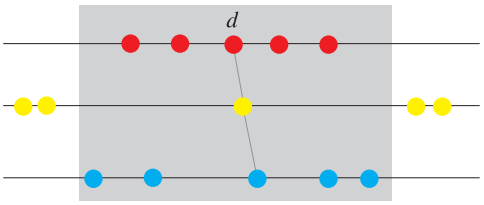
3. Consider the case $k > 2$. Fix a "growing" tuple of k rational numbers, i.e. a tuple $\bar{z}$ such that for $1 \leq i < j \leq k$, it holds that $z_1 \leq z_i < z_j \leq z_k$. Define $\prec_{ij}^{\bar{z}}$ to be the restriction of $\prec$ to tuples that agree with $\bar{z}$ on coordinates from $\{1, \ldots, k\} \setminus \{i, j\}$. Using the reasoning from the previous item, the ordering $\prec_{ij}^{\bar{z}}$ must admit some dominating coordinate $d \in \{i, j\}$ and one of the cases "growing" or "decreasing". This must hold for every choice of $\bar{z}$ and i, j. Furthermore, the dominating coordinate d depends only on i and j and not on $\bar{z}$, likewise for the choice of "growing" or "decreasing". Let us write $i \to j$ if j dominates, otherwise we write $j \to i$. The reasoning in the following picture shows that $\to$ is transitive, i.e. $i \to j$ and $j \to m$ implies $i \to m$:



Step 1. Move coordinate i to its final position, and move j so that the tuple grows

Step 2. Move coordinate j back to the initial, position and move m to its final position

Therefore, $\to$ is in fact a total order on $\{1, \ldots, k\}$. Let d be the maximum with respect to this order. The following picture explains why d is the dominating coordinate d from the statement of Lemma 12.



Step 1. For every $i \neq d$, use $i \to d$ to move coordinates $i \neq d$ outside the gray interval (the gray interval contains $\bar{x}$ and $\bar{y}$)

Step 2. Move coordinates $i \neq d$ to their final positions

Suppose without loss of generality that we are in the "growing" case for each pair of coordinates. Then we can first move all coordinates apart from d to positions smaller than $\min\{x_1, y_1\}$ or bigger than $\max\{x_k, y_k\}$ and then use the dominations $i \to d$ to move them, one by one, to their final positions (always increasing the d-th coordinate slightly to a value in the open interval (x_d, y_d)). ◀

4.2 Proof of Lemma 11

We now return to Lemma 11, i.e., we prove that every definable k-enumerator is also programmable. In the proof, we use the following version of the Factorisation Forest Theorem. We use the term *interval* for a connected set of positions in a string.

▶ **Theorem 14** (Factorisation Forest Theorem, aperiodic variant). *Let $h\colon \Sigma^+ \to S$ be a semigroup homomorphism, where S is finite and aperiodic. Then there exists a function which assigns to each string in Σ^+ a partition of the positions into intervals (so-called* blocks*) such that:*

1. *All blocks are nonempty, and for each string in Σ^+ of length at least 2, there are at least two blocks.*
2. *If a string has at least three blocks, then all of the blocks have the same value under h.*
3. *There exists $M \in \mathbb{N}$ such that all strings have height at most M, where the height of a string is defined as follows: letters have height 1, for other strings the height is the maximum of the heights of its blocks + 1.*
4. *There is a first-order formula φ such that for every string w, the positions satisfying $\varphi(x)$ are exactly the first positions of the blocks of w.*

Apart from the Factorisation Forest Theorem and the Domination Lemma, our proof uses the following straightforward result on combining outputs of two for-programs. As a convention, if ψ is a first-order formula with k free variables and f is a k-enumerator, then $f|\psi$ denotes the k-enumerator where the output list of f is filtered so that it contains only tuples satisfying ψ.

▶ **Lemma 15** (Merging Lemma). *Let f be a definable k-enumerator. Let Φ be a finite set of* FO *formulas ψ, each one with k free variables, such that every k-tuple of positions satisfies at least one formula from Φ. Then f is programmable if and only if every $f|\psi$ is programmable.*

We are now ready to prove Lemma 11. Let f be a definable k-enumerator. We need to describe a for-program that outputs the same list of tuples as f. Let r be the maximal quantifier rank of the first-order formulas used in the definition of f. Apply the Domination Lemma to k, $m \coloneqq 5k$, and r, yielding a constant ω. Define h to be the function which maps a string $w \in \Sigma^+$ to the rank ω type of the corresponding ordered model of w. Compositionality of first-order logic (see [16, Section 3.4]) on strings says that the image of h, the set of rank ω types of strings, is a finite aperiodic semigroup and h is a semigroup homomorphism. Apply the Factorisation Forest Theorem to h, yielding a function that partitions each string into blocks and an upper bound M on heights of strings. By abuse of notation, we lift notions about strings to intervals inside strings: the height of an interval X in a string w is defined to be the height (in the sense of item 3 in Theorem 14) of the infix of w induced by X. Likewise, we define the blocks of X as the blocks of the infix induced by X, viewed as intervals contained in X.

To show that f is also programmable, we use an induction over heights in factorisation forests. More precisely, we prove that for every $i \in \mathbb{N}$ there is a for-program which computes the following:

- **Input.** A string $w \in \Sigma^+$ with distinguished nonempty intervals $X_1, \dots, X_k$ that are pairwise equal or disjoint, and such that the sum of their heights (in the sense of Theorem 14) is at most i. Each interval is represented by its first and its last position.
- **Output.** The list $f(w)$ restricted to tuples in $X_1 \times \cdots \times X_k$.

By item 3 in Theorem 14, the for-program with parameter $i \coloneqq kM$ will work for every choice of pairwise equal or disjoint intervals, in particular when all of the intervals are the entire string. The induction base $i = k$ (where every interval has the height 1) is straightforward: each interval is a singleton, and the for-program simply checks if the unique tuple in $X_1 \times \cdots \times X_k$ belongs to the output of f by using the subroutines from Lemma 6. The rest of the proof is devoted to the induction step, more specifically, to producing the correct order of the tuples: whether a tuple belongs to the output or not can again be checked using the subroutines from Lemma 6.

Let $X_1, \dots, X_k$ be intervals in an input string w that are pairwise disjoint or equal. Define $\mathcal{X}$ to be the coarsest partition of the positions in the input string into intervals that satisfies $X_1, \dots, X_k \in \mathcal{X}$. This partition uses at most $2k + 1$ intervals. Consider a factorisation

$$w = w_1 \cdots w_n$$

where each w_j is a block of one of the elements of $\mathcal{X}$. Define $\mathfrak{A}$ as in the Domination Lemma, i.e. as the ordered structure of w extended with an extra order $\sqsubset$ that describes the partition into factors $w_1, \dots, w_n$. By item 4 of the Factorisation Forest Theorem, the order $\sqsubset$ can be defined by a first-order formula which uses the input string and the endpoints of the intervals $X_1, \dots, X_k$. It follows that for every k-ary rank ω type t over the vocabulary of $\mathfrak{A}$, there is a corresponding first-order formula that selects the k-tuples of positions in w that have type t in $\mathfrak{A}$. Since there are finitely many choices of t, it follows from the Merging Lemma that it is enough to show that for every t, there is a for-program which outputs the tuples of type t.

Let t be a k-ary rank ω type over the vocabulary of $\mathfrak{A}$. We show a for-program which outputs all tuples in

$$T \coloneqq \{\bar{x} \in X_1 \times \cdots \times X_k : \bar{x} \text{ has type } t \text{ and is in the output of } f(w)\}$$

according to their order given by $f(w)$, call this order $\prec$.

If an interval from $\mathcal{X}$ has more than two blocks, then, by item 2 of the Factorisation Forest Theorem, all of these blocks have the same image under h, i.e., the same rank ω type. Since there are at most $2k+1$ intervals, it follows that with at most $2(2k+1)-1 = 4k+1 < 5k$ exceptions, consecutive strings w_j and w_{j+1} have the same rank ω type. Hence, for the order $\prec$ defined by $f(w)$, the Domination Lemma yields $d \in \{1,\dots,k\}$ and $p \in \{-1,1\}$ such that

$$x_d \sqsubset^p y_d \quad \text{implies} \quad (x_1,\dots,x_k) \prec (y_1,\dots,y_k) \qquad \text{for all } \underbrace{x_1,\dots,x_k}_{\text{of type } t}, \underbrace{y_1,\dots,y_k}_{\text{of type } t} \text{ in } \mathfrak{A}.$$

This means that the tuples in T are $\prec$-ordered as $T_1 \prec^p T_2 \prec^p \cdots \prec^p T_s$, where s is the number of blocks in X_d and T_j consists of the tuples from T where the coordinate x_d is in the j-th block of X_d. Our for-program can simply loop over all the blocks of X_d – in increasing or decreasing order depending on the choice of p – because the endpoints of each block can be identified in first-order logic due to item 4 of the Factorisation Forest Theorem. In each iteration of the loop, the for-program outputs the tuples in the corresponding T_j using the following claim, thus completing the proof of the lemma.

▷ Claim 16. There is a for-program which inputs the i-th block of X_d, given by its endpoints, and outputs the tuples from T_j ordered according to $\prec$.

Proof of the claim. The general idea is to replace X_d with its j-th block (call this block X) and use the induction assumption. However, if there is some $j \neq d$ such that $X_j = X_d$, then replacing X_d with X would violate the assumption that the intervals are pairwise disjoint or equal (since $X \subsetneq X_j$). To overcome this issue, we use the following simple case disjunction. For each of the 3^k possible values of

$$v \in \{\text{positions before } X,\ X,\ \text{positions after } X\}^k$$

construct a for-program that outputs all tuples from $Y_1 \times \cdots \times Y_k$, where Y_j is the intersection of X_j with the j-th entry of v. Since each Y_j is a union of blocks of X_j, it is empty or its height is at most the height of X_j. Furthermore, if Y_d is nonempty, then it is X, which is a block of X_d, and therefore its height is strictly smaller than the height of X_d. It follows that the induction assumption can be applied to produce all tuples in $Y_1 \times \cdots \times Y_k$, for any given choice of v. These choices can be combined using the Merging Lemma. ◁

References

1. Rajeev Alur and Pavol Cerný. Streaming transducers for algorithmic verification of single-pass list-processing programs. In *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, Austin, TX, USA, January 26-28, 2011*, pages 599–610, 2011. doi:10.1145/1926385.1926454.
2. Rajeev Alur, Adam Freilich, and Mukund Raghothaman. Regular combinators for string transformations. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, page 9. ACM, 2014.
3. Mikołaj Bojańczyk. Polyregular Functions, 2018. arXiv:1810.08760.
4. Mikołaj Bojańczyk and Wojciech Czerwiński. Automata Toolbox. URL: https://www.mimuw.edu.pl/~bojan/upload/reduced-may-25.pdf.
5. Mikołaj Bojańczyk, Laure Daviaud, and Shankara Narayanan Krishna. Regular and First-Order List Functions. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 125–134, 2018. doi:10.1145/3209108.3209163.

6 Mikołaj Bojańczyk, Sandra Kiefer, and Nathan Lhote. String-to-string interpretations with polynomial-size output, 2019. arXiv:1905.13190.

7 Thomas Colcombet. A combinatorial theorem for trees. In *International Colloquium on Automata, Languages, and Programming*, pages 901–912. Springer, 2007.

8 Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic.* A Language-Theoretic Approach. Cambridge University Press, June 2012.

9 Vrunda Dave, Paul Gastin, and Shankara Narayanan Krishna. Regular Transducer Expressions for Regular Transformations. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 315–324, 2018. doi:10.1145/3209108.3209182.

10 Joost Engelfriet. Two-way pebble transducers for partial functions and their composition. *Acta Informatica*, 52(7-8):559–571, 2015.

11 Joost Engelfriet and Hendrik Jan Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *ACM Transactions on Computational Logic (TOCL)*, 2(2):216–254, 2001.

12 Noa Globerman and David Harel. Complexity Results for Two-Way and Multi-Pebble Automata and their Logics. *Theor. Comput. Sci.*, 169(2):161–184, 1996.

13 Erich Grädel, Phokion G. Kolaitis, Leonid Libkin, Maarten Marx, Joel Spencer, Moshe Y. Vardi, Yde Venema, and Scott Weinstein. *Finite Model Theory and Its Applications.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 2007. doi:10.1007/3-540-68804-8.

14 Wilfrid Hodges. *Model Theory.* Cambridge University Press, Cambridge, March 1993.

15 Wojciech Kazana and Luc Segoufin. Enumeration of monadic second-order queries on trees. *ACM Transactions on Computational Logic (TOCL)*, 14(4):1–12, 2013.

16 Leonid Libkin. *Elements of finite model theory.* Springer Science & Business Media, 2013.

17 Tova Milo, Dan Suciu, and Victor Vianu. Typechecking for XML transformers. *Journal of Computer and System Sciences*, 66(1):66–97, 2003.

18 Imre Simon. Factorization forests of finite height. *Theoretical Computer Science*, 72(1):65–94, 1990.

19 Wolfgang Thomas. Languages, Automata, and Logic. In *Handbook of Formal Languages, Volume 3: Beyond Words*, pages 389–455. Springer, 1997. doi:10.1007/978-3-642-59126-6_7.

A Kleene Theorem for Nominal Automata

Paul Brunet
University College London, UK
paul.brunet-zamansky.fr
paul@brunet-zamansky.fr

Alexandra Silva
University College London, UK
www.alexandrasilva.org
alexandra.silva@ucl.ac.uk

Abstract

Nominal automata are a widely studied class of automata designed to recognise languages over infinite alphabets. In this paper, we present a Kleene theorem for nominal automata by providing a syntax to denote regular nominal languages. We use regular expressions with explicit binders for creation and destruction of names and pinpoint an exact property of these expressions – namely memory-finiteness – identifying a subclass of expressions denoting exactly regular nominal languages.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects; Theory of computation → Formal languages and automata theory

Keywords and phrases Kleene Theorem, Nominal automata, Bracket Algebra

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.107

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version A full version of the paper is available at http://hal.inria.fr/hal-02112892.

Supplement Material A Coq library is available on GitHub.

Funding ERC Starting Grant ProFoundNet (grant code 679127)
Paul Brunet: EPSRC project EP/R006865/1
Alexandra Silva: Leverhulme Prize (PLP-2016-129)

1 Introduction

Languages over infinite alphabets have been studied in a variety of contexts: query-based languages [8], XML processing [19], URLs [1], process calculi [5], etc. Accordingly, a number of automata models have been introduced for these languages, either register-based, where the state space is finite but registers are available for storing data, or based on nominal sets, where the state space is infinite but can be represented finitely due to symmetries. The most general classes of such automata are Kaminski and Francez's finite-memory automata (FMA) [8], in the register-based style, and Bojańczyk, Klin and Lasota's nondeterministic orbit-finite automata (NOFA) [4], in the nominal style. These two kinds of automata have been shown to have the same expressivity [4], and equivalence is known to be undecidable [8, 16].

While automata are useful to process and compare languages, to specify languages it is often more natural to use regular expressions; this is for instance the standard way of denoting a path in an XML tree. To that effect, many classes of expressions have been proposed [9, 12, 11, 18, 14]. The expressions from [14] capture the full class of languages recognised by either FMA of NOFA, but having been developed for FMAs they are not straightforwardly suitable to describe NOFA languages. Some of the other formalisms are more natural in the context of nominal automata, but all fail to capture the full class, and instead coincide with some (usually decidable) sub-classes.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 107; pp. 107:1–107:13


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

We define in this paper a new class of regular expressions for data languages, originally motivated by applications to program verification, as part of larger framework called *bracket algebra*. These expressions feature explicit allocation $\langle_a$ and deallocation $_a\rangle$ binders, and may be used to generate nominal languages. We prove in this paper that they are in fact able to describe every language recognisable by a NOFA.

Let us illustrate our syntax on simple examples. To make this discussion simpler, we assume for now that our alphabet is an infinite set of names $\mathbb{A}$. The first notion we present is that of α-equivalence of words with binders. Here we choose to define α-equivalence as the smallest congruence stable by permutation of bound or fresh names. For instance the following pair of words is equivalent: $\langle_a\, a\langle_b\, b\,{}_a\rangle\langle_a\, a\,{}_b\rangle\langle_b\, b\,{}_a\rangle\,{}_b\rangle =_\alpha \langle_b\, b\langle_c\, c\,{}_b\rangle\langle_d\, d\,{}_c\rangle\langle_a\, a\,{}_d\rangle\,{}_a\rangle$. Indeed, we may derive this as follows (we underline the redex at each step):

$$\begin{aligned}\langle_a\, a\underline{\langle_b\, b\,{}_a\rangle\langle_a\, a\,{}_b\rangle}\langle_b\, b\,{}_a\rangle\,{}_b\rangle &=_\alpha \langle_a\, a\langle_c\, c\,{}_a\rangle\underline{\langle_a\, a\,{}_c\rangle\langle_b\, b\,{}_a\rangle}\,{}_b\rangle \\ &=_\alpha \underline{\langle_a\, a\langle_c\, c\,{}_a\rangle}\langle_d\, d\,{}_c\rangle\langle_b\, b\,{}_d\rangle\,{}_b\rangle =_\alpha \langle_b\, b\langle_c\, c\,{}_b\rangle\langle_d\, d\,{}_c\rangle\langle_a\, a\,{}_d\rangle\,{}_a\rangle.\end{aligned}$$

We then define well-formed words to be those without name capture, i.e. for every prefix $u\langle_a$, every $\langle_a$ in u must be matched with a corresponding $_a\rangle$. For instance $\langle_a\, a\langle_b\, b\,{}_b\rangle\,{}_a\rangle$ is well-formed, but $\langle_a\, a\langle_a\, a\,{}_a\rangle\,{}_a\rangle$ is not, even though the two are equivalent. Now, consider regular expressions over an alphabet composed of names from $\mathbb{A}$ and binders $\langle_a$ and $_a\rangle$. We associate to such an expression e a nominal language $(\!|e|\!)$ in several steps:

1) take the regular language $[\![e]\!]$ denoted by e;
2) compute its closure by α-equivalence $[\![e]\!]^\alpha$, adding every word that is equivalent to some word in the initial language;
3) restrict this language to its well-formed members;
4) erase the brackets.

Here are some examples:

L1 $:= (\!|\langle_a\, a\,{}_a\rangle|\!) = \mathbb{A}$: the set of all atoms;
L2 $:= (\!|\langle_a\, \langle_b\, ab\,{}_b\rangle\,{}_a\rangle|\!) = \{ab \mid a \neq b\}$: two letter words made of different letters;
L3 $:= (\!|\langle_a\, a\,{}_a\rangle^\star|\!) = \mathbb{A}^\star$: the set of all words;
L4 $:= (\!|\langle_a\, a\langle_a\, a\,{}_a\rangle^\star\,{}_a\rangle|\!) = \{a_1 \dots a_n \mid n > 0, \forall 1 < i, a_i \neq a_1\}$: the set of words such that the first letter is different from all others;
L5 $:= (\!|\langle_a\, \langle_a\, a\,{}_a\rangle^\star a\,{}_a\rangle|\!) = \{a_1 \dots a_n \mid n > 0, \forall i < n, a_i \neq a_n\}$: the set of words such that the last letter is different from all others;
L6 $:= (\!|\langle_a\, a\,(\langle_b\, b\,{}_a\rangle\langle_a\, a\,{}_b\rangle)^\star\,(1 + \langle_b\, b\,{}_b\rangle)\,{}_a\rangle|\!) = \{a_1a_2 \dots a_n \mid n > 0, \forall i, a_i \neq a_{i+1}\}$: the set of non-empty words such that two consecutive letters are different;
L7 $:= (\!|(\langle_a\, x)^\star\,(y\,{}_a\rangle)^\star|\!) = \{x\} \cup \{x^n y^m \mid n \leqslant m\}$.

As one can see, this technique allows for the definition of a large class of nominal languages. In fact this class is in some sense "too large" and contains languages that are not regular, like for instance L7. To get a Kleene theorem, we therefore introduce a tractability condition: we ask regular expressions to have a memory-finite language. Intuitively this means there should be a number N such that any prefix of a word in the language has less than N unmatched brackets. This condition is decidable by induction on expressions, and such expressions generate exactly the class of languages recognisable by NOFAs. The main result of the paper is an exact correspondence between memory-finite nominal languages and NOFA:

▶ **Theorem 1** (Kleene Theorem). *Let L be a nominal language. The following are equivalent:*
(i) *L is rational, that is $L = (\!|e|\!)$ for some memory-finite regular expression e.*
(ii) *L is regular nominal, that is recognisable by a NOFA.*

The paper is structured as follows. In Section 2, we define our notations, and recall some elements of nominal automata theory. We introduce in Section 3 words with explicit binders and define an α-equivalence relation for these words. To recognise this relation we construct in Section 4 a nominal transducer. We then present in Section 5 our syntax for regular expressions with binders, and prove in Section 6 our main result, a Kleene theorem for NOFA. We briefly discuss related work in Section 7. We omit some proofs in this paper, but a longer version is available on HAL.

This paper is part of a larger research program developing a framework to reason about programs with explicit resource (de)allocation. A companion paper describing the algebraic framework of *bracket algebra* and a hierarchy of nominal languages can be found online, as well as a Coq formalisation of the framework.

2 Preliminaries

The set of finite subsets of a set A is denoted by $\mathcal{P}_f(A)$. If A is finite, its cardinal is denoted by $\#A \in \mathbb{N}$. The set of words over an alphabet Σ is written $\Sigma^\star$. The empty word is denoted by ε, concatenation of words u and v is written uv, and $|u|$ is the length of word u. For $w \in \Sigma^\star$ and $x \in \Sigma$, $|w|_x$ is the number of occurrences of x in w. We write $\lfloor w \rfloor$ for the set of letters appearing in the word w, i.e. $\lfloor w \rfloor := \{x \in \Sigma \mid |w|_x > 0\}$. We denote the i^{th} letter of a word u by u_i, for $0 < i \leqslant |u|$. The set of prefixes of a language is defined as: $\mathbf{pref}(L) := \{u \in \Sigma^\star \mid \exists v : uv \in L\}$.

Given a set A, and $B \subseteq A$, the set $B^{\bowtie}$ of shuffles of B consists of the lists without repetitions of elements from B: $B^{\bowtie} := \{l \in B^\star \mid |l| = \#B \wedge (\forall 0 < i < j \leqslant |l|\,,\ l_i \neq l_j)\}$. Observe that if $w \in B^{\bowtie}$, then $\{a \mid \exists 0 < i \leqslant |w| : w_i = a\} = B$. We say that a list $l \in A^\star$ is duplication-free, written $l \in A^{(\star)}$ when $l \in \{a \mid \exists 0 < i \leqslant |l| : l_i = a\}^{\bowtie}$.

Rational expressions over an alphabet Σ are terms generated by the following grammar: $e, f \in \mathrm{Rat}\,\langle\Sigma\rangle ::= 0 \mid 1 \mid l \mid e + f \mid e \cdot f \mid e^\star$, where l ranges over the alphabet Σ. Such a term e denotes a language $[\![e]\!]$, defined in the usual way:

$$[\![0]\!] := \emptyset, \qquad [\![1]\!] := \{\varepsilon\}\,, \quad [\![e \cdot f]\!] := \{uv \mid u \in [\![e]\!] \wedge v \in [\![f]\!]\}\,,$$
$$[\![e + f]\!] := [\![e]\!] \cup [\![f]\!]\,, \quad [\![l]\!] := \{l\}\,, \qquad [\![e^\star]\!] := [\![e]\!]^\star = \{u_1 \dots u_n \mid n \in \mathbb{N} \wedge \forall i, u_i \in [\![e]\!]\}\,.$$

2.1 Nominal sets

We fix an infinite set $\mathbb{A}$ of atoms (also called names), and write $\mathfrak{S}_{\mathbb{A}}$ the set of finitely supported permutations over $\mathbb{A}$. These are bijections π such that there is a finite set $\bar{a} \subseteq \mathbb{A}$ such that $a \notin \bar{a} \Rightarrow \pi(a) = a$. In the following we let $a, b, \dots$ range over $\mathbb{A}$ and $\bar{a}, \bar{b}, \dots$ range over finite sets of atoms. The inverse of a permutation π is written π^{-1} . The permutation exchanging a and b, and leaving every other name unchanged, is written $(a\ b)$. We say that a permutation π fixes a finite set $\bar{a} \subseteq \mathbb{A}$, written $\pi \perp \bar{a}$, when $\forall a \in \bar{a},\ \pi(a) = a$.

A set X is called nominal if it can be equipped with two functions, respectively action $- \cdot - : \mathfrak{S}_{\mathbb{A}} \times X \to X$ and support $\mathbf{supp}(-) : X \to \mathcal{P}_f(\mathbb{A})$, satisfying $\forall x \in X,\ \forall \pi, \pi' \in \mathfrak{S}_{\mathbb{A}}$,:

$$\pi \perp \mathbf{supp}(x) \Rightarrow \pi \cdot x = x. \tag{$\dagger_1$}$$
$$\mathbf{supp}(\pi \cdot x) = \left\{a \in \mathbb{A} \;\middle|\; \pi^{-1}(a) \in \mathbf{supp}(x)\right\}. \tag{$\dagger_2$}$$
$$\pi \cdot (\pi' \cdot x) = (\pi \circ \pi') \cdot x. \tag{$\dagger_3$}$$

Intuitively, this means that we may replace a name by another in any element of X, and that each element of X only depends on a finite number of names. We say that a permutation π fixes a subset $Y \subseteq X$, also written $\pi \perp Y$ if $\forall y \in Y, \pi \cdot y = y$. This enables use to state $(\dagger_1)$ as $\pi \perp \mathbf{supp}\,(x) \Rightarrow \pi \perp x$. We say that the name a is fresh for x, and write $a \# x$, whenever $a \notin \mathbf{supp}\,(x)$. We will also use the notation $X \restriction_{\bar{a}}$ to mean $\{x \in X \mid \mathbf{supp}\,(x) \subseteq \bar{a}\}$.

▶ Remark 2. In Pitts' book [17] a nominal set is defined as a $\mathfrak{S}_{\mathbb{A}}$-action such that every element has some finite support. From conditions $(\dagger_1)$ and $(\dagger_3)$ we infer that X is a nominal set as in [17]. Furthermore, condition $(\dagger_2)$ enforces that $\mathbf{supp}\,(x)$ is the least finite set that supports x, so our notion of support coincides with the one introduced in [17]. For Coq implementation considerations, we chose to include the support function in the definition.

For the rest of this section, we fix a nominal set X. Given $x, y \in X$, we say that x and y are in the same orbit, written $x \sim_{\mathcal{O}} y$, if there exists $\pi \in \mathfrak{S}_{\mathbb{A}}$ such that $x = \pi \cdot y$. This is an equivalence relation, and its equivalence classes are called orbits. A subset $Y \subseteq X$ is called:

- strict if it has no symmetries, i.e. $(\dagger_1)$ holds as an equivalence: $\pi \perp \mathbf{supp}\,(y) \Leftrightarrow \pi \perp y$;
- equivariant if for every permutation $\pi \in \mathfrak{S}_{\mathbb{A}}$, we have $\pi \cdot Y = Y$, meaning

$$\forall \pi \in \mathfrak{S}_{\mathbb{A}}, \forall y \in X,\ y \in Y \Leftrightarrow \pi \cdot y \in Y;$$

- finitely supported if there is a finite $\bar{a} \subseteq \mathbb{A}$ such that $\pi \perp \bar{a}$ entails $\pi \cdot Y = Y$;
- orbit-finite if Y only intersects finitely many orbits;
- tractable if it is both orbit-finite and finitely supported.

▶ Remark 3. In Bojańczyk [2, 3] terminology, what we call tractable sets are simply called orbit-finite, even though these are sets that are both orbit-finite and finitely supported. We chose a different name to avoid confusion as in other papers orbit-finite sets are not necessarily finitely supported.

In the following, we will use the following results adapted from [3]:

▶ **Lemma 4** (Simple extension of Lemma 3.5 in [3]). *Every tractable set can be expressed as the image of a tractable set of words from $\mathbb{A}^\star$ by some equivariant function.*

▶ **Lemma 5** (Fact 3.6 in [3]). *Tractable sets are closed under finite unions and products, and under finitely supported subsets.*

2.2 Nominal automata

Let Σ be an orbit-finite nominal alphabet. A nominal automaton (NOFA) over Σ is a structure $\mathcal{A} = \langle Q, \Sigma, \Delta, I, F \rangle$ where Q is a tractable state space, $I, F \subseteq Q$ are finitely supported sets of respectively initial and final states, and $\Delta \subseteq Q \times \Sigma \times Q$ is a finitely supported transition relation. This definition corresponds to Bojańczyk, Klin, and Lasota's "orbit-finite automata" [4]. We define the automaton's path relation in the usual way, by saying that $p \xrightarrow{\varepsilon}_{\mathcal{A}} p$ and whenever $p \xrightarrow{w}_{\mathcal{A}} q'$ and $\langle q', x, q \rangle \in \Delta$ then we also have $p \xrightarrow{wx}_{\mathcal{A}} q$. Notice that since Δ is finitely supported, so is the path relation. The language recognised by such an automaton is defined as usual as the set of traces leading from an initial state to a final state:

$$\mathcal{L}_{\mathcal{A}} := \left\{ w \in \Sigma^\star \;\middle|\; \exists \langle q_i, q_f \rangle \in I \times F : q_i \xrightarrow{w}_{\mathcal{A}} q_f \right\}.$$

Nominal regular languages are those recognised by nominal automata.

▶ Remark 6. In the literature, the name "Nominal automaton" is sometimes used to refer to a different class of automata, where the tractability requirement is replaced by orbit-finite and equivariant. These two classes define the same languages: an equivariant automaton is a particular case of a tractable one, and any tractable automaton with support $\bar{a}$ might be seen as an equivariant automaton by replacing the set of atoms $\mathbb{A}$ with the set $\mathbb{A} \setminus \bar{a}$. However, we feel that our approach leads to more intuitive encoding of some natural languages. Consider for instance the language $a \cdot \mathbb{A}^\star$ of words over $\mathbb{A}$ starting with the letter $a \in \mathbb{A}$. This language is not equivariant, therefore to represent it with an equivariant automaton one needs to remove the name a from the set of names, considering instead the alphabet $\mathbb{A}$ as a nominal set over the set of names $\mathbb{A} \setminus \{a\}$. We feel this is a bit counter-intuitive. However, it may be represented by a simple tractable automaton with two states p and q, with p initial, q final, a transition $p \xrightarrow{a} q$, and transitions $q \xrightarrow{b} q$ for every name $b \in \mathbb{A}$.

We will later on rely on the following properties of nominal automata.

▶ **Lemma 7.** *Every nominal automaton is language equivalent to a nominal automaton whose state space is strict.*

▶ **Lemma 8.** *Nominal automata enjoy ε-elimination.*

A nominal automaton is called deterministic (DOFA) if it has a single initial state and its transition relation is a deterministic function, i.e. if we have two transitions $\langle p, x, q\rangle \in \Delta \wedge \langle p, x, q'\rangle \in \Delta$, then $q = q'$. Languages recognised by DOFA form a strict subclass of the regular nominal languages. E.g. the language over $\mathbb{A}$ of words with the last letter distinct from all others is regular nominal but cannot be recognised by a DOFA: intuitively to check for membership one needs to guess what will be the last letter before reading the word. There is also a significant complexity difference: equivalence of DOFA is decidable in polynomial time [15], the corresponding problem for NOFA is undecidable [16].

▶ **Lemma 9.** *Regular languages can be recognised by deterministic nominal automata.*

Proof. Regular languages can be recognised by deterministic finite state automata. Being finite, such automata are also tractable, thus deterministic nominal automata. ◀

2.3 Nominal transductions

We will make intensive use of transductions in this paper. A nominal transducer is a nominal automaton over an alphabet of the shape $(\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\})$. For a nominal transducer $\mathcal{T}$, we may define its path relation $-[-/-]\!\to_{\mathcal{T}}$ and the binary relation $\mathcal{R}_{\mathcal{T}}$ it recognises:

$$\frac{}{p \,-[\varepsilon/\varepsilon]\!\to_{\mathcal{T}} p} \qquad \frac{p \,-[w/w']\!\to_{\mathcal{T}} q' \quad \langle q', \langle x, x'\rangle, q\rangle \in \Delta}{p \,-[wx/w'x']\!\to_{\mathcal{T}} q}$$

$$\mathcal{R}_{\mathcal{T}} := \{\langle u, v\rangle \in \Sigma^\star \times \Gamma^\star \mid \exists \langle q_i, q_f\rangle \in I \times F : q_i \,-[u/v]\!\to_{\mathcal{T}} q_f\}\,.$$

A binary relation $R \subseteq \Sigma^\star \times \Gamma^\star$ is called a nominal transduction if it is recognised by some nominal transducer. For a transduction R, we will sometimes see R as either a function $\Sigma^\star \to \mathcal{P}(\Gamma^\star)$ or a function $\mathcal{P}(\Sigma^\star) \to \mathcal{P}(\Gamma^\star)$, writing:

$$u \in \Sigma^\star,\ R(u) := \{v \in \Gamma^\star \mid u\ R\ v\} \qquad L \subseteq \Sigma^\star,\ R(L) := \{v \in \Gamma^\star \mid \exists u \in L : u\ R\ v\}\,.$$

This should not introduce any ambiguity, thanks to typing considerations.

▶ **Lemma 10.** *Nominal regular languages are stable under nominal transductions.*

Proof. Let Σ, Δ be two tractable alphabets, and $\Sigma' := (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\})$. Consider a nominal automaton $\mathcal{A} = \langle Q_1, \Sigma, \Delta_1, I_1, F_1 \rangle$ and a nominal transducer $\mathcal{T} = \langle Q_2, \Sigma', \Delta_2, I_2, F_2 \rangle$. We want to show that the language $\mathcal{R}_{\mathcal{T}}(\mathcal{L}_{\mathcal{A}})$ is regular nominal, by building a nominal automaton $\mathcal{T}(\mathcal{A})$ with ε-transitions. Its states are in $Q_1 \times Q_2$, with initial and final states respectively $I_1 \times I_2$ and $F_1 \times F_2$. Its transition relation is given by:

$$\Delta := \{\langle \langle p_1, p_2 \rangle, x, \langle q_1, q_2 \rangle \rangle \mid \exists y : \langle p_1, y, q_1 \rangle \in \Delta_1 \wedge \langle p_2, \langle y, x \rangle, q_2 \rangle \in \Delta_2\}$$
$$\cup \{\langle \langle p_1, p_2 \rangle, x, \langle p_1, q_2 \rangle \rangle \mid \langle p_2, \langle \varepsilon, x \rangle, q_2 \rangle \in \Delta_2\}.$$ ◀

3 Words over an alphabet with binders

For the rest of the paper, we fix an orbit-finite nominal set $\mathbb{X}$ of variables, to represent our alphabet. We consider words built out of variables, left and right binders, respectively written $\langle_a$ and $_a\rangle$. These binders are meant to represent the creation and destruction of names.

We now introduce a notion of α-equivalence for these words. This relation will be a congruence stable under substitution of "local" names: for instance the words $\langle_a\ {}_a\rangle$ and $\langle_b\ {}_b\rangle$ are equivalent. The definitions in this section are straightforward adaptations from [7].

Formally, we define our alphabet by $\mathbb{\Sigma} := \mathbb{X} \cup \{\langle_a \mid a \in \mathbb{A}\} \cup \{{}_a\rangle \mid a \in \mathbb{A}\}$. This alphabet can be endowed with a nominal structure in the obvious way, by setting $\pi \cdot \langle_a = \langle_{\pi(a)}$, $\pi \cdot {}_a\rangle = {}_{\pi(a)}\rangle$, and $\mathbf{supp}(\langle_a) = \mathbf{supp}({}_a\rangle) = \{a\}$. In the following, a word with binders will be an element of $\mathbb{\Sigma}^\star$, that is a finite sequence of letters from the alphabet $\mathbb{\Sigma}$. Words with binders come with a natural nominal structure: the action is defined by applying the alphabet action letter by letter, and the support of a word is the union of the supports of its letters.

Before we define α-equivalence, we need to introduce the notion of *binding power* of a word with binders. The purpose of this notion is to keep track of the occurrences of each name along a word, and enable us to decide whether a particular name is local to the word, and more generally to get a precise account of the way the name is used in the word, from the point of view of the context. The binding monoid $\mathcal{B}$ is defined as the free monoid over the three element set $\{\mathbf{c}, \mathbf{f}, \mathbf{d}\}$, quotiented by the identities: $\mathbf{f} \cdot \mathbf{f} = \mathbf{f}$, $\mathbf{c} \cdot \mathbf{f} = \mathbf{c}$, $\mathbf{f} \cdot \mathbf{d} = \mathbf{d}$, and $\mathbf{c} \cdot \mathbf{d} = \varepsilon$. The letters $\mathbf{c}$, $\mathbf{f}$, $\mathbf{d}$ are meant to represent that a name might be *created*, *free* or *destroyed*. An important property of this monoid is the following, as noticed in [7]: every element of $\mathcal{B}$ can be uniquely represented in the form $\mathbf{d}^m\mathbf{f}^n\mathbf{c}^p$, with $\langle m, n, p \rangle \in \mathbb{N} \times \{0, 1\} \times \mathbb{N}$. We use this remark to define the *size*[1] of a binding element $b \in \mathcal{B}$ as $|\mathbf{d}^m\mathbf{f}^n\mathbf{c}^p| = m + p$.

The binding power of a letter $l \in \mathbb{\Sigma}$ with respect to a name $a \in \mathbb{A}$, written $\mathcal{F}_a(l)$, is computed as follows:

$$\mathcal{F}_a(\langle_b) := \begin{cases} \mathbf{c} & (a = b) \\ \varepsilon & (a \neq b) \end{cases} \qquad \mathcal{F}_a({}_b\rangle) := \begin{cases} \mathbf{d} & (a = b) \\ \varepsilon & (a \neq b) \end{cases} \qquad \mathcal{F}_a(x) := \begin{cases} \mathbf{f} & (a \in \mathbf{supp}(x)) \\ \varepsilon & (a \# x) \end{cases}$$

The function $\mathcal{F}$ may be extended to words naturally as a monoid homomorphism, by setting $\mathcal{F}_a(\varepsilon) = \varepsilon$ and $\mathcal{F}_a(lw) = \mathcal{F}_a(l) \cdot \mathcal{F}_a(w)$. If $\mathcal{F}_a(u) = \mathbf{d}^m\mathbf{f}^n\mathbf{c}^p$ with $n \in \{0, 1\}$, we define $d_a(u) := m$, $f_a(u) := n$, and $c_a(u) := p$. This is well defined thanks to the uniqueness of such representations. This function is equivariant, in the sense that $\mathcal{F}_{\pi(a)}(\pi \cdot u) = \mathcal{F}_a(u)$.

The weight of a word u is the sum of the sizes of its binding powers: $\|u\| := \sum_{a \in \mathbb{A}} |\mathcal{F}_a(u)|$. This sum is finite, since for every name a outside the finite set $\mathbf{supp}(u)$ we know that the binding power of u with respect to a is ε, so $|\mathcal{F}_a(u)| = 0$. The memory of a word u is the maximum weight of a prefix of u, i.e. $\mathbf{m}(u) := \max\{\|v\| \mid \exists w \in \mathbb{\Sigma}^\star : vw = u\}$.

[1] Since the size of a Boolean is constant, we do not count n in the size of $\mathbf{d}^m\mathbf{f}^n\mathbf{c}^p$. This simplifies a number of computations.

Table 1 Alpha-equivalence.

$$\frac{}{\varepsilon =_\alpha \varepsilon}\ (\alpha\varepsilon) \qquad \frac{u =_\alpha v \quad v =_\alpha w}{u =_\alpha w}\ (\alpha\mathbf{t}) \qquad \frac{w_1 =_\alpha w_2}{w_1 l =_\alpha w_2 l}\ (\alpha\mathbf{r}) \qquad \frac{w_1 =_\alpha w_2}{l w_1 =_\alpha l w_2}\ (\alpha\mathbf{l})$$

$$\frac{a \diamond u \quad b \#_\alpha u}{\langle_a u\, {}_a\rangle =_\alpha \langle_b (a\ b) \cdot u\, {}_b\rangle}\ (\alpha\alpha).$$

(a) Definition of Alpha-equivalence.

$$u =_\alpha v \Rightarrow v =_\alpha u \quad (1) \qquad u =_\alpha v \Rightarrow \forall \pi,\ \pi \cdot u =_\alpha \pi \cdot v \quad (4)$$
$$u =_\alpha v \wedge u' =_\alpha v' \Rightarrow uu' =_\alpha vv' \quad (2) \qquad u =_\alpha v \Rightarrow |u| = |v| \quad (5)$$
$$u =_\alpha v \Rightarrow \forall a,\ \mathcal{F}_a(u) = \mathcal{F}_a(v) \quad (3)$$

(b) Properties of Alpha-equivalence.

We use the binding power to define the following: a is balanced in the word w, written $a \diamond w$, if $\mathcal{F}_a(w) \in \{\mathbf{f}, \varepsilon\}$; a is α-fresh in w, written $a \#_\alpha w$, if $\mathcal{F}_a(w) = \varepsilon$; the α-support of w, written $\mathbf{supp}_\alpha(w)$, is the set of names a such that $\mathcal{F}_a(w) \neq \varepsilon$. Notice that $\mathbf{supp}_\alpha(w) \subseteq \mathbf{supp}(w)$. Therefore, we get that $\pi(a) \#_\alpha \pi \cdot u$ if and only if $a \#_\alpha u$, and similarly for $\pi(a) \in \mathbf{supp}_\alpha(\pi \cdot u)$ and $\pi(a) \diamond \pi \cdot u$.

We may now define the α-equivalence relation over words. It is the smallest congruence such that applying the transposition $(a\ b)$ to a word where a and b are α-fresh yields an equivalent word. We give the formal definition of $=_\alpha$ in Table 1a and list some of its properties in Table 1b. The propositions (1) and (2) state that $=_\alpha$ is symmetric and that concatenation is compatible with $=_\alpha$, which together with $(\alpha\varepsilon)$and $(\alpha\mathbf{t})$establishes $=_\alpha$ as a congruence, while (3), (4), and (5) are necessary preservation properties of $=_\alpha$. The proofs of these results follow a simple induction of proof trees.

Note that the deduction system we provided for $=_\alpha$ is not a priori equivalent to the informal description we gave before. However, the correspondence can be proved in the sense that the same relation is obtained if we replace rule $(\alpha\alpha)$ with the following rule:

$$\frac{a \#_\alpha u \quad b \#_\alpha u}{u =_\alpha (a\ b) \cdot u}\ (\alpha\alpha')$$

However, this proof is not straightforward: $(\alpha\alpha')$ obviously implies $(\alpha\alpha)$ (as the latter may be seen as an instance of the former), but the converse direction is more subtle. Unfortunately, this is the most interesting direction, as it is necessary to show that words quotiented by $=_\alpha$ form a nominal set, with the support function $\mathbf{supp}_\alpha()$. This property may however be established using the transducer presented in the next section.

We say that a word u is well-formed when for every decomposition $u = u_1 \langle_a u_2$, we have $c_a(u_1) = 0$. Intuitively, this means that there is no name capture for bound variables. The set of well-formed words is written $\mathcal{WF}$, and we define $\mathbf{wf}(u) := \{v \mid u =_\alpha v \wedge v \in \mathcal{WF}\}$.

4 A transducer for α-equivalence-checking

The problem that arises when trying to prove statements like $(\alpha\alpha)$is that α-equivalence is not preserved in the inductive calls: the property $ux =_\alpha vy$ does not entail $u =_\alpha v$. In this section we introduce a nominal transducer recognising the relation $=_\alpha$. The reachability

relation in this transducer will give us more powerful proof techniques, allowing us to perform proofs by induction. This transducer serves several purposes: it provides us with a decision procedure for $=_\alpha$, enables us to show that $(\alpha\alpha')$ is admissible, and will be used here as a bridge between nominal automata and rational expressions over $\mathbb{\Sigma}$.

4.1 Stacks

The states of this transducer will consist of lists of pairs of atoms, called stacks in the following. Before we define the transducer, we introduce some useful notations. Stacks are generated by the following grammar: $s \in \mathbb{S} ::= [] \mid s :: \langle a, b\rangle$, where a, b range over names. Hence $\mathbb{S}$ is isomorphic to $(\mathbb{A} \times \mathbb{A})^\star$. We will also use the notation $s :: t$ for the concatenation of the two stacks $s, t \in \mathbb{S}$. We write $\mathfrak{p}_1(s)$ for the word over $\mathbb{A}$ obtained by erasing the second components of every pair in s, and symmetrically $\mathfrak{p}_2(s)$ when we erase the first components. For instance $\mathfrak{p}_1([] :: \langle a, b\rangle :: \langle c, d\rangle) = ac$, and $\mathfrak{p}_2([] :: \langle a, b\rangle :: \langle c, d\rangle) = bd$.

Stacks can be endowed with a canonical nominal structure defined by:

$$\pi \cdot [] := [] \qquad \pi \cdot (s :: \langle a, b\rangle) := \pi \cdot s :: \langle \pi(a), \pi(b)\rangle$$
$$\mathbf{supp}([]) := \emptyset \qquad \mathbf{supp}(s :: \langle a, b\rangle) := \mathbf{supp}(s) \cup \{a, b\}.$$

Note that $\mathbf{supp}(s) = \mathbf{supp}(\mathfrak{p}_1(s)) \cup \mathbf{supp}(\mathfrak{p}_2(s)) = \lfloor \mathfrak{p}_1(s) \rfloor \cup \lfloor \mathfrak{p}_2(s) \rfloor$.

The pivotal notions for stacks are the validates predicate and the pop function. We say that a stack s validates the pair $\langle a, b\rangle$, written $s \models \langle a, b\rangle$, when either $a = b$ and $a \# s$, or s can be decomposed as $s = s' :: \langle a, b\rangle :: s''$ in such a way that $a \notin \lfloor \mathfrak{p}_1(s'') \rfloor$ and $b \notin \lfloor \mathfrak{p}_2(s'') \rfloor$. When s validates $\langle a, b\rangle$, we may pop the pair from s, yielding the stack $s \ominus \langle a, b\rangle$ defined by:

$$\frac{a \notin \mathbf{supp}(s)}{s \ominus \langle a, a\rangle := s} \qquad \frac{a \notin \lfloor \mathfrak{p}_1(s') \rfloor \quad b \notin \lfloor \mathfrak{p}_2(s') \rfloor}{(s :: \langle a, b\rangle :: s') \ominus \langle a, b\rangle := s :: s'}.$$

4.2 Equivalence transducer

We now define the equivalence transducer $\mathcal{T}_\alpha$, recognising $=_\alpha$. Strictly speaking, this will not be a nominal transducer, as we will discuss later on. Its state space is $\mathbb{S}$, with initial state $[]$, and the set of accepting states $\mathbb{S}^{acc}$ consists of all stacks s containing only reflexive pairs, i.e. such that $\mathfrak{p}_1(s) = \mathfrak{p}_2(s)$. The transition relation $-[-/-]\to_{\mathcal{T}_\alpha}$ is defined by:

$$\begin{array}{rcl} & & s \,-[\langle_a / \langle_b]\to_{\mathcal{T}_\alpha}\, s :: \langle a, b\rangle \\ s \models \langle a, b\rangle & \Rightarrow & s \,-[\,_a\rangle / \,_b\rangle]\to_{\mathcal{T}_\alpha}\, s \ominus \langle a, b\rangle \\ \forall a \in \mathbf{supp}(x),\ s \models \langle a, \pi(a)\rangle & \Rightarrow & s \,-[x/\pi \cdot x]\to_{\mathcal{T}_\alpha}\, s \end{array}$$

Note that this relation is functional, in the sense that for every triple $\langle s, l, l'\rangle \in \mathbb{S} \times \mathbb{\Sigma} \times \mathbb{\Sigma}$ there exists at most one stack s' such that $s \,-[l/l']\to_{\mathcal{T}_\alpha}\, s'$. This transducer over an infinite state space is equivariant, as one can easily check that $s \,-[u/v]\to_{\mathcal{T}_\alpha}\, s'$ entails $\pi \cdot s \,-[\pi \cdot u/\pi \cdot v]\to_{\mathcal{T}_\alpha}\, \pi \cdot s'$. However, it is not orbit finite. This seems to be unavoidable since there are infinitely many α-equivalence classes (in particular, words of different length cannot be equivalent).

▶ **Theorem 11.** *The relation $\mathcal{R}_{\mathcal{T}_\alpha}$ is exactly $=_\alpha$.*

The full proof has been done in Coq. The following technical lemma allows one to relate the binding power of a word with the stack contents:

▶ **Lemma 12.** *Whenever* $s \xrightarrow{-[u/v]\to_{\mathcal{T}_\alpha}} s'$ *the following identities hold:*

$$|\mathfrak{p}_1(s')|_a = (|\mathfrak{p}_1(s)|_a \dot{-} d_a(u)) + c_a(u) \qquad |\mathfrak{p}_2(s')|_a = (|\mathfrak{p}_2(s)|_a \dot{-} d_a(v)) + c_a(v).$$

(Where $\dot{-}$ *is the truncated subtraction.)*

This lemma has the following corollaries:

▶ **Corollary 13.** *If* $[] \xrightarrow{-[u/v]\to_{\mathcal{T}_\alpha}} s \xrightarrow{-[u'/v']\to_{\mathcal{T}_\alpha}} s'$ *then* $|s| \leqslant \mathbf{m}(uu')$.

Proof. By Lemma 12, and since $[] \xrightarrow{-[u/v]\to_{\mathcal{T}_\alpha}} s$, we have $|s| = \sum_a c_a(u) \leqslant \|u\|$. Since $\|u\| \leqslant \mathbf{m}(uu')$, the result follows. ◀

▶ **Corollary 14.** *For any words* u, v *of length* n*, the following are equivalent:*

(i) $u =_\alpha v$ *and* $v \in \mathcal{WF}$*;*

(ii) *there are stacks* $s_0 \dots s_n$ *such that* $s_0 = []$*,* $s_n \in \mathbb{S}^{acc}$*, for every index* $0 \leqslant i < n$ *we have* $s_i \xrightarrow{-[u_{i+1}/v_{i+1}]\to_{\mathcal{T}_\alpha}} s_{i+1}$*, and for any index* $0 \leqslant i \leqslant n$ *and name* a *we have* $|\mathfrak{p}_2(s_i)|_a \leqslant 1$.

These results allow us to show that the following are nominal transductions:

$$=_\alpha^{\leqslant n} := \{\langle u, v\rangle \mid u =_\alpha v \wedge \mathbf{m}(u) \leqslant n\}$$
$$\mathbf{wf}^n := \{\langle u, v\rangle \mid u =_\alpha v \wedge \mathbf{m}(u) \leqslant n \wedge v \in \mathcal{WF}\}.$$

▶ **Theorem 15.** *For any* $n \in \mathbb{N}$*, both* $=_\alpha^{\leqslant n}$ *and* $\mathbf{wf}^n$ *are nominal transductions.*

Proof. Thanks to Corollary 13, we know that $=_\alpha^{\leqslant n}$ is recognised by $\mathcal{T}_\alpha$ restricted to states $\mathbb{S}^{\leqslant n}$, made up of stacks of length less that n. This is a tractable set, by Lemma 5. Combined with Corollary 14, this proves that $\mathbf{wf}^n$ is recognised by $\mathcal{T}_\alpha$ restricted to stacks such that $|s| \leqslant n$ and $\forall a, |\mathfrak{p}_2(s)|_a \leqslant 1$. This set of stacks being an equivariant subset of $\mathbb{S}^{\leqslant n}$, by Lemma 5 it is also tractable. ◀

5 Memory-finite rational languages

In this section we consider regular languages over $\mathbb{\Sigma}$, i.e. languages $[\![e]\!]$ for some $e \in \mathrm{Rat}\langle\mathbb{\Sigma}\rangle$. We may lift α-equivalence to languages by first defining the α-closure of a language L as:

$$L^\alpha := \{u \in \mathbb{\Sigma}^\star \mid \exists v \in L,\ u =_\alpha v\}.$$

Now we say that two languages are equivalent if their α-closures are equal.

We lift the support function from $\mathbb{\Sigma}$ to $\mathrm{Rat}\langle\mathbb{\Sigma}\rangle$ in the canonical way: for letters in $\mathbb{\Sigma}$ we use the $\mathbf{supp}(-)$ function from the nominal structure of the alphabet, the support of 0 and 1 is the empty set, the support of $e^\star$ is that of e and the support of both $e + f$ and $e \cdot f$ is $\mathbf{supp}(e) \cup \mathbf{supp}(f)$. This definition is an over approximation of the pointwise lifting of the support function on words: indeed $\bigcup_{u \in [\![e]\!]} \mathbf{supp}(u) \subseteq \mathbf{supp}(e)$. Note that $\mathbf{supp}(e)$ is always finite, and supports $[\![e]\!]$ in the sense that whenever $\pi \perp \mathbf{supp}(e)$, we have $\pi \cdot [\![e]\!] = [\![e]\!]$.

A language $L \subseteq \mathbb{\Sigma}^\star$ is called memory-finite if there exists a bound N such that $\forall u \in L, \mathbf{m}(u) \leqslant N$. A rational expression is memory-finite if its language is memory-finite.

▶ **Lemma 16.** *For any rational expression* e*, the following are equivalent:*

(i) e *is memory-finite;*

(ii) *the set* $\{\mathcal{F}_a(u) \mid u \in [\![e]\!],\ a \in \mathbb{A}\}$ *is finite;*

(iii) $\forall u \in [\![e]\!], \mathbf{m}(u) \leqslant 2 \times |e|$.

(Where $|e|$ *is the number of occurrences of letters in* e*.)*

This lemma was proved in Coq. The following result is of independent interest:

▶ **Theorem 17.** *If e is memory-finite, then $[\![e]\!]^\alpha$ is recognisable by DOFA.*

Proof. Let N be the memory of $[\![e]\!]$. By definition, this means that $[\![e]\!]^\alpha$ is equal to the language $=_\alpha^{\leqslant N}([\![e]\!])$. However, the automaton built by applying the construction from Lemma 10 does not yield a deterministic automaton, even if the input automaton is deterministic. Fortunately, in the present case we can determinise the resulting automaton. To do so, we will rely on the following technical result about $\mathcal{T}_\alpha$, which was established using Coq: for every word $u \in \mathbb{\Sigma}^\star$, there is a word $tr(u) \in \mathbb{A}^\star$ such that for any stack s and word v:

$$[] -[u/v]\to_{\mathcal{T}} s \Rightarrow \mathfrak{p}_1(s) = tr(u) \qquad [] -[v/u]\to_{\mathcal{T}} s \Rightarrow \mathfrak{p}_2(s) = tr(u).$$

Notice that this implies that $\mathbf{supp}(tr(u)) \subseteq \mathbf{supp}(u)$: indeed since $u =_\alpha u$ there is a stack s such that $[] -[u/u]\to_{\mathcal{T}_\alpha} s$, so $tr(u) = \mathfrak{p}_1(s)$, and according to Lemma 12 whenever $a \in \mathfrak{p}_1(s)$ we have $c_a(u) \neq 0$ which implies $a \in \mathbf{supp}(u)$.

Let $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$ be some deterministic finite-state automaton for $[\![e]\!]$, with $\Sigma \in \mathcal{P}_f(\mathbb{\Sigma})$. We write $\bar{a}$ for the finite set of names mentioned in the finite alphabet Σ: $\bar{a} := \bigcup_{l\in\Sigma} \mathbf{supp}(l) \subseteq \mathbf{supp}(e)$. Notice that this means that $\pi \perp \bar{a} \Rightarrow \pi \perp \Sigma^\star$. Without loss of generality, we assume that δ is a partial function $Q \times \Sigma \to Q$ and that $\mathcal{A}$ has no sink-state: for any state $q \in Q$, there exists a word $u \in \Sigma^\star$ such that $\delta(q,u) \in F$. If we look back at the proof of Lemma 10, we see that the states in the automaton we get for $=_\alpha^{\leqslant N}(\mathcal{A})$ are pairs of a state from Q and a stack from $\mathbb{S}^{\leqslant N} := \left(\mathbb{A}^2\right)^{\leqslant N}$. Now, let us do the standard powerset construction on this automaton: we get an automaton $\mathcal{A}' := \langle Q', \mathbb{\Sigma}, \delta', q_0', F'\rangle$ where:

$$Q' = \mathcal{P}\left(Q \times \mathbb{S}^{\leqslant N}\right); \qquad q_0' = \{\langle q_0, []\rangle\}; \qquad F' = \{\bar{q} \in Q' \mid \bar{q} \cap (F \times \mathbb{S}^{acc}) \neq \emptyset\};$$
$$\delta'(\bar{q}, l) = \{\langle q', s'\rangle \mid \exists \langle q, s\rangle \in \bar{q}, \exists l' \in \Sigma : q' = \delta(q,l) \wedge s -[l'/l]\to_{\mathcal{T}_\alpha} s'\}.$$

Unfortunately, the state space Q' is not tractable, since it is not orbit-finite. However, as we will now prove, the subset of reachable states is tractable. Therefore if we restrict $\mathcal{A}'$ to its reachable part we get a language-equivalent DOFA. A state $\bar{q} \in Q'$ is reachable if there exists a word v such that $\delta'(q_0', v) = \bar{q}$. By unfolding the definitions, we can see that $\bar{q}$ is reachable by the word v when the following equivalence is satisfied:

$$\forall q, s : \langle q, s\rangle \in \bar{q} \Leftrightarrow \exists u : q = \delta(q_0, u) \wedge [] -[u/v]\to_{\mathcal{T}_\alpha} s.$$

This implies that $\forall \langle q, s\rangle \in \bar{q}$, $\mathfrak{p}_2(s) = tr(v)$, and $\mathfrak{p}_1(s) = tr(u)$ for some $u \in \mathbf{pref}([\![e]\!])$. This second condition tells us that $\mathfrak{p}_1(s) \in \bar{a}^{\leqslant N}$ which is a finite set. Hence the set of reachable states is contained (modulo isomorphism) in the set: $\mathcal{Q} := \mathcal{P}\left(Q \times \bar{a}^{\leqslant N}\right) \times \mathbb{A}^{\leqslant N}$. This set being the product of a finite set with a tractable one, it is tractable. Notice that the set of reachable states is supported by the finite set $\bar{a}$: indeed if $\pi \perp \bar{a}$, then we already know that $\pi \perp \Sigma^\star$ so if $\bar{q}$ is reachable by the word v, then $\pi \cdot \bar{q}$ is reachable by $\pi \cdot v$ since:

$$\begin{aligned}\langle q, s\rangle \in \pi \cdot \bar{q} \Leftrightarrow \left\langle q, \pi^{-1} \cdot s\right\rangle \in \bar{q} &\Leftrightarrow \exists u : q = \delta(q_0, u) \wedge [] -[u/v]\to_{\mathcal{T}_\alpha} \pi^{-1} \cdot s \\ &\Leftrightarrow \exists u : q = \delta(q_0, u) \wedge \pi \cdot [] -[\pi \cdot u/\pi \cdot v]\to_{\mathcal{T}_\alpha} s \\ &\Leftrightarrow \exists u : q = \delta(q_0, u) \wedge [] -[u/\pi \cdot v]\to_{\mathcal{T}_\alpha} s.\end{aligned}$$

We conclude that the set of reachable states is tractable by applying Lemma 5, which tells us that a finitely supported subset of a tractable set is tractable. ◀

We may use expressions over $\mathbb{\Sigma}$ to generate languages over $\mathbb{X}$ as follows: the language generated by a term $e \in \mathrm{Rat}\langle\mathbb{\Sigma}\rangle$, written $(\!|e|\!)$, is the set of words obtained by erasing the brackets from the well-formed words from $[\![e]\!]^\alpha$. In other words, if we denote by η the monoid homomorphism defined by $\eta(\langle_a) = \eta(\,_a\rangle) = \varepsilon$ and $\eta(x) = x$, we have $(\!|e|\!) := \eta(\mathbf{wf}([\![e]\!]))$.

6 Kleene Theorem

In this section, we show that regular nominal languages over $\mathbb{X}$ are exactly those generated by memory-finite rational expressions. To that end, we call a language L rational if there is some memory-finite expression e such that $L = (\!|e|\!)$. One direction is immediate:

▶ **Lemma 18.** *For any memory-finite expression e, $(\!|e|\!)$ is regular nominal.*

Proof. Since e is memory-finite, according to Lemma 16, every word in e has memory less than $2 \times |e|$. Therefore, $\mathbf{wf}\left([\![e]\!]\right) = \mathbf{wf}^{2\times|e|}\left([\![e]\!]\right)$. By the classic Kleene theorem $[\![e]\!]$ is regular and thanks to Theorem 15 we know that $\mathbf{wf}^{2\times|e|}$ is a nominal transduction. Since we may also see easily that η is a nominal transduction, the statement follows from Lemma 10. ◀

We now show that nominal regular languages are rational. We fix a nominal automaton $\mathcal{A} = \langle Q, \mathbb{X}, \Delta, I, F\rangle$, and assume without loss of generality that its state space is strict, equivariant and orbit-finite. We also fix a finite set $\bar{\mathbf{a}}_0 \subseteq \mathbb{A}$ that supports I, F and Δ. As a first step, we will find a finite sub-automaton of $\mathcal{A}$ that is "large enough" to describe the language of $\mathcal{A}$. We do this by picking a finite set $\bar{\mathbf{a}} \subseteq \mathbb{A}$ such that:

$$\forall \alpha \in I \cup F \cup \Delta, \exists \beta \in I \cup F \cup \Delta : \mathbf{supp}\left(\beta\right) \subseteq \bar{\mathbf{a}} \wedge \exists \pi : \pi \perp \bar{\mathbf{a}}_0 \wedge \pi \cdot \beta = \alpha.$$

Such a set always exists: we just need to pick a representative per orbit, and take the union of their supports. As a shorthand, we write $\mathfrak{S}_0$ for the set of permutations over $\mathbb{A} \setminus \bar{\mathbf{a}}_0$, i.e. the permutations $\pi \in \mathfrak{S}_{\mathbb{A}}$ such that π fixes $\bar{\mathbf{a}}_0$. We then define the finite automaton $\mathcal{A} \restriction_{\bar{\mathbf{a}}} := \langle Q \restriction_{\bar{\mathbf{a}}}, \mathbb{X} \restriction_{\bar{\mathbf{a}}}, \Delta \restriction_{\bar{\mathbf{a}}}, I \restriction_{\bar{\mathbf{a}}}, F \restriction_{\bar{\mathbf{a}}}\rangle$. We can relate the runs of $\mathcal{A} \restriction_{\bar{\mathbf{a}}}$ to those in $\mathcal{A}$ as follows.

▶ **Lemma 19.** *For any letters $(x_i)_{1,\dots,n}$ and any states $(q_i)_{0,\dots,n}$, t.f.a.e.:*

(i) *there is a run $p_0 \xrightarrow{x_1}_{\mathcal{A}} p_1 \dots \xrightarrow{x_n}_{\mathcal{A}} p_n$*

(ii) *there is a run $q_0 \xrightarrow{y_1}_{\mathcal{A}\restriction_{\bar{\mathbf{a}}}} q_1 \cdots \xrightarrow{y_n}_{\mathcal{A}\restriction_{\bar{\mathbf{a}}}} q_n$ and a sequence $(\pi_i)_{0,\dots,n}$ from $\mathfrak{S}_0$ such that $\pi_0 \cdot q_0 = p_0$ and $\forall i > 0$ we have $\pi_i \cdot \langle q_{i-1}, y_i, q_i\rangle = \langle p_{i-1}, x_i, p_i\rangle$.*

We now define a finite automaton $\mathcal{A}'$ over the alphabet $\mathbb{\Sigma} \restriction_{\bar{\mathbf{a}}}^{\star}$. The state space of this automaton will be $Q' := Q \restriction_{\bar{\mathbf{a}}} \cup \{q_0, q_f\}$, with q_0 and q_f fresh states, respectively the initial and final states. We build its transitions as follows:

1. we have $q_0 \xrightarrow{\langle a_1 \cdots \langle a_n}$ $q \in \Delta'$ for any $q \in I \restriction_{\bar{\mathbf{a}}}$, and any word $a_1 \dots a_n \in \left(\bar{\mathbf{a}}_0 \cup \mathbf{supp}\left(q\right)\right)^{\bowtie}$;
2. we have $q \xrightarrow{a_1 \rangle \cdots a_n \rangle}$ $q_f \in \Delta'$ for any $q \in F \restriction_{\bar{\mathbf{a}}}$, and any word $a_1 \dots a_n \in \left(\mathbf{supp}\left(q\right) \setminus \bar{\mathbf{a}}_0\right)^{\bowtie}$;
3. we have $p \xrightarrow{\langle a_1 \cdots \langle a_n\, x\, b_1 \rangle \cdots b_m \rangle}$ $q \in \Delta'$ for every transition $p \xrightarrow{x}_{\mathcal{A}\restriction_{\bar{\mathbf{a}}}} q$ and any pair of words:

$$a_1 \dots a_n \in \left(\left(\mathbf{supp}\left(q\right) \cup \mathbf{supp}\left(x\right)\right) \setminus \left(\mathbf{supp}\left(p\right) \cup \bar{\mathbf{a}}_0\right)\right)^{\bowtie}$$
$$b_1 \dots b_m \in \left(\left(\mathbf{supp}\left(p\right) \cup \mathbf{supp}\left(x\right)\right) \setminus \left(\mathbf{supp}\left(q\right) \cup \bar{\mathbf{a}}_0\right)\right)^{\bowtie}.$$

Since we have only a finite number of transitions, we know that this automaton may be transformed into a finite state automaton over $\mathbb{\Sigma} \restriction_{\bar{\mathbf{a}}}$, therefore thanks to Kleene's theorem there is a rational expression $e \in \mathrm{Rat}\,\langle \mathbb{\Sigma} \rangle$ such that $[\![e]\!] = \mathcal{L}_{\mathcal{A}'}$. We now need to check that e is memory-finite and that $(\!|e|\!) = \mathcal{L}_{\mathcal{A}}$. For the first property, we show the following lemma:

▶ **Lemma 20.** *For every run $q_0 \xrightarrow{w}_{\mathcal{A}'} q \in Q \restriction_{\bar{\mathbf{a}}}$, the word $w \in \mathcal{WF}$, $\mathbf{m}\left(w\right) \leqslant \#\bar{\mathbf{a}}$ and either $a \in \mathbf{supp}\left(q\right) \cup \bar{\mathbf{a}}_0$ and $\mathcal{F}_a\left(w\right) = \mathbf{c}$, or $a \notin \mathbf{supp}\left(q\right) \cup \bar{\mathbf{a}}_0$ and $\mathcal{F}_a\left(w\right) = \varepsilon$.*

This entails that $[\![e]\!] \subseteq \mathcal{WF}$ and $\mathbf{m}\left(e\right) \leqslant \#\bar{\mathbf{a}}$. Lemma 20 will also serve in the next proof.

▶ **Lemma 21.** *For any $w \in \mathbb{\Sigma}^\star$, the word w belongs to $\mathbf{wf}\left(\mathcal{L}_{\mathcal{A}'}\right)$ if and only if there is a sequence of permutations $\pi_0 \dots \pi_{n+1} \in \mathfrak{S}_0$ and a run $q_0 \xrightarrow{u_0}_{\mathcal{A}'} q_1 \xrightarrow{u_1}_{\mathcal{A}'} \cdots \xrightarrow{u_n}_{\mathcal{A}'} q_{n+1} \xrightarrow{u_{n+1}}_{\mathcal{A}'} q_f$ such that $w = (\pi_0 \cdot u_0) \dots (\pi_{n+1} \cdot u_{n+1})$ and $\forall 0 < i \leqslant n,\ \pi_{i-1} \cdot q_i = \pi_i \cdot q_i$.*

From Lemmas 19 and 21 it is not hard to see that our construction is correct, thus proving that every regular nominal language is rational.

▶ **Theorem 1** (Kleene Theorem)**.** *Let L be a nominal language. The following are equivalent:*

(i) *L is rational, that is $L = (\!|e|\!)$ for some memory-finite regular expression e.*

(ii) *L is regular nominal, that is recognisable by a NOFA.*

7 Related work

Schröder at al.'s *regular bar-expressions* [18] enjoy a Kleene-like theorem. Regular bar-expressions add an operator $|_a$ to the alphabet, intuitively writing an a on the right-hand side of the bar, and hiding it from the left-hand side. These expressions are equipped with two semantics, called "local" and "global" freshness. Under "local" freshness, the class of automata represented by these expressions is a strict subset of the class of nominal automata, where no name may be guessed (i.e. for every transition $p \xrightarrow{x} q$ we have $\mathbf{supp}\,(q) \subseteq \mathbf{supp}\,(p) \cup \mathbf{supp}\,(x)$), and where a policy of "name dropping" is enforced: a name may be in the support of a state only if it will appear later. For instance, this precludes recognising the languages L_2 and L_5 from the introduction. Under "global" freshness however the situation is more contrasted. With this semantics, the expressive power of bar-expressions is incomparable with that of memory-finite expressions. Indeed, they can denote the language of words where all the letters are different by $|a^\star$, but cannot denote $L_3 := (\!|\langle_a\, a\, {}_a\rangle^\star|\!) = \mathbb{A}^\star$. However if we drop the memory-finite requirement, one can translate bar-expressions into regular expressions over $\mathbb{\Sigma}$ by replacing every occurrence of $|a$ with $\langle_a\, a$ and suffixing the expression with $\left(\sum_{a \in \mathbf{supp}(e)} {}_a\rangle\right)^\star$. For instance the term $|a^\star$ is sent to the expression $(\langle_a\, a)^\star\ {}_a\rangle^\star$. In this case, our well-formed predicate corresponds to the *clean* predicate used to define the global freshness semantics, and this transformation preserves languages. This means that unrestricted expressions with brackets are strictly more expressive than bar-expressions.

In a study of Nominal Kleene Algebra [11, 10, 6], *NKA expressions* were introduced, and half a Kleene theorem for NOFA was proved. These expressions feature a unary $\nu_a(e)$ operator to make a name a local to an expression e. These expressions do not allow the interleaving of scopes, thus failing to capture languages such as 5 from the introduction.

Kurz et al. [13] considered regular expressions with binders. However, their framework only accounts for well nested brackets, thus not covering many of the languages we consider. They present a Kleene theorem for history-dependent automata that incorporates a bound on the nesting depth of binding, rejecting words that exceed this depth, which is the analogue restriction at the automaton level of our memory-finiteness property at the language level. It is unclear whether HD-automata could be generalised to accommodate interleaving of scopes.

On the other hand Libkin and Vrgoč's *regular expressions with memory* [14] enjoy a full Kleene theorem with register automata. Since register automata and nominal automata are equi-expressive, this means that regular expressions with memory are as expressive as our memory-finite expressions. They are however quite different in style. The point of view they choose is that of data words: they assume a finite alphabet Σ and an infinite set of data values $\mathcal{D}$, and consider languages over the alphabet $\Sigma \times \mathcal{D}$, i.e. each letter carries a data value. The key feature of their syntax is to use annotation on letters. They fix a number of variables $x_1 \dots x_k$, and use regular expressions over an alphabet made of elements of the shape $a[c]{\downarrow}I$ where a is a letter from Σ, I is a subset of the variables, and c is a boolean

formula that may use atomic predicates $x_i^{=}$ and $x_i^{\neq}$. These expressions are then interpreted as ternary relations, linking two k-tuples of data values with data words. In effect, this amounts to simulating the run of a register automaton where the k-tuples of data values represent the content of the registers.

References

1 Michal Bielecki, Jan Hidders, Jan Paredaens, Jerzy Tyszkiewicz, and Jan Van den Bussche. Navigating with a Browser. In *ICALP*, pages 764–775, 2002. doi:10.1007/3-540-45465-9_65.

2 Mikołaj Bojańczyk. Nominal Monoids. *Theory of Computing Systems*, 53(2):194–222, 2013. doi:10.1007/s00224-013-9464-1.

3 Mikołaj Bojańczyk. Slightly Infinite Sets. A draft of a book, 2017. URL: https://www.mimuw.edu.pl/~bojan/paper/atom-book.

4 Mikołaj Bojańczyk, Bartek Klin, and Sławomir Lasota. Automata theory in nominal sets. *Logical Methods in Computer Science*, 10(3):1–44, 2014. doi:10.2168/LMCS-10(3:4)2014.

5 Benedikt Bollig, Peter Habermehl, Martin Leucker, and Benjamin Monmege. A Robust Class of Data Languages and an Application to Learning. *Logical Methods in Computer Science*, 10, 2014. doi:10.2168/LMCS-10(4:19)2014.

6 Paul Brunet and Damien Pous. A Formal Exploration of Nominal Kleene Algebra. In *MFCS*, 2016. doi:10.4230/LIPIcs.MFCS.2016.22.

7 Jamie Gabbay, Dan R. Ghica, and Daniela Petrişan. Leaving the Nest: Nominal Techniques for Variables with Interleaving Scopes. In *CSL*, volume 41, 2015. doi:10.4230/LIPIcs.CSL.2015.374.

8 Michael Kaminski and Nissim Francez. Finite-memory automata. *Theoretical Computer Science*, 134(2):329–363, 1994. doi:10.1016/0304-3975(94)90242-9.

9 Michael Kaminski and Tony Tan. Regular Expressions for Languages over Infinite Alphabets. In *Computing and Combinatorics*, 2004. doi:10.1007/978-3-540-27798-9_20.

10 Dexter Kozen, Konstantinos Mamouras, and Alexandra Silva. Completeness and Incompleteness in Nominal Kleene Algebra. In *RAMiCS*, 2015. doi:10.1007/978-3-319-24704-5_4.

11 Dexter Kozen, Konstantinos Mamouras, Alexandra Silva, and Daniela Petrişan. Nominal Kleene Coalgebra. In *ICALP*, volume 9135, pages 290–302, 2015. doi:10.1007/978-3-662-47666-6.

12 Alexander Kurz, Tomoyuki Suzuki, and Emilio Tuosto. A Characterisation of Languages on Infinite Alphabets with Nominal Regular Expressions. In *TCS*, pages 193–208, 2012.

13 Alexander Kurz, Tomoyuki Suzuki, and Emilio Tuosto. On Nominal Regular Languages with Binders. In *FoSSaCS*, pages 255–269, 2012.

14 Leonid Libkin, Tony Tan, and Domagoj Vrgoč. Regular Expressions for Data Scientists. *Journal of Computer and System Sciences*, 81(7):1278–1287, 2015. doi:10.1016/j.jcss.2015.03.005.

15 Andrzej S Murawski, Steven J Ramsay, and Nikos Tzevelekos. Polynomial-Time Equivalence Testing for Deterministic Fresh-Register Automata. In *MFCS*, 2018. doi:10.4230/LIPIcs.MFCS.2018.72.

16 Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings over infinite alphabets. *ACM Transactions on Computational Logic*, 5(3):403–435, 2004. doi:10.1145/1013560.1013562.

17 Andrew M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*. Cambridge University Press, New York, NY, USA, 2013.

18 Lutz Schröder, Dexter Kozen, Stefan Milius, and Thorsten Wißmann. Nominal Automata with Name Binding. In *FoSSaCS*, pages 124–142, 2017. doi:10.1007/978-3-662-54458-7_8.

19 Thomas Schwentick. Automata for XML – A survey. *Journal of Computer and System Sciences*, 73(3):289–315, 2007. doi:/10.1016/j.jcss.2006.10.003.

Completeness of Graphical Languages for Mixed States Quantum Mechanics

Titouan Carette
Université de Lorraine, CNRS, Inria, LORIA, F 54000 Nancy, France
https://members.loria.fr/TCarette/accueil/
titouan.carette@loria.fr

Emmanuel Jeandel
Université de Lorraine, CNRS, Inria, LORIA, F 54000 Nancy, France
https://members.loria.fr/EJeandel/
emmanuel.jeandel@loria.fr

Simon Perdrix
Université de Lorraine, CNRS, Inria, LORIA, F 54000 Nancy, France
https://members.loria.fr/SPerdrix/
simon.perdrix@loria.fr

Renaud Vilmart
Université de Lorraine, CNRS, Inria, LORIA, F 54000 Nancy, France
https://members.loria.fr/RVilmart/
renaud.vilmart@loria.fr

Abstract

There exist several graphical languages for quantum information processing, like quantum circuits, ZX-Calculus, ZW-Calculus, etc. Each of these languages forms a †-symmetric monoidal category (†-SMC) and comes with an interpretation functor to the †-SMC of (finite dimension) Hilbert spaces. In the recent years, one of the main achievements of the categorical approach to quantum mechanics has been to provide several equational theories for most of these graphical languages, making them complete for various fragments of pure quantum mechanics.

We address the question of the extension of these languages beyond pure quantum mechanics, in order to reason on mixed states and general quantum operations, i.e. completely positive maps. Intuitively, such an extension relies on the axiomatisation of a *discard* map which allows one to get rid of a quantum system, operation which is not allowed in pure quantum mechanics.

We introduce a new construction, the *discard construction*, which transforms any †-symmetric monoidal category into a symmetric monoidal category equipped with a discard map. Roughly speaking this construction consists in making any isometry causal.

Using this construction we provide an extension for several graphical languages that we prove to be complete for general quantum operations. However this construction fails for some fringe cases like the Clifford+T quantum mechanics, as the category does not have enough isometries.

2012 ACM Subject Classification Mathematics of computing; Theory of computation → Quantum computation theory; Theory of computation → Logic

Keywords and phrases Quantum Computing, Quantum Categorical Mechanics, Category Theory, Mixed States, Completely Positive Maps

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.108

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version A full version of the paper is available at https://arxiv.org/abs/1902.07143.

Funding This work is funded by ANR-17-CE25-0009 SoftQPro and PIA-GDN/Quantex.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 108; pp. 108:1–108:15

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

1 Introduction

Graphical languages that speak of quantum information can be formalised through the notion of symmetric monoidal categories. Hence, it has a nice graphical representation using string diagrams [41]. Qubits are represented by wires, and morphisms by graphical elements where some wires go in, and some others go out, just as in quantum circuits (which is actually a particular case of symmetric monoidal category), and where these graphical elements can be composed either in sequence (usual composition) or in parallel (tensor product). They usually come with an additional structure, a contravariant functor called dagger.

Examples of graphical languages for quantum mechanics and quantum computing are the quantum circuits and the ZX-Calculus [11]. Some variants of the ZX-calculus have been introduced more recently like the ZW-calculus [26] and the ZH-calculus [6]. All these languages are defined using generators (elementary gates) and come with an interpretation functor which associates to any diagram a pure quantum evolution, i.e. a morphism in the category of Hilbert spaces. Given a graphical language, there are generally several ways to represent a quantum evolution, thus a graphical language is also equipped with an equational theory which allows to transform a diagram into another equivalent diagram. A fundamental property, generally hard to prove, is the completeness of the language: given two diagrams representing the same quantum evolution, one can be turned into the other using only the transformation rules in the theory.

The languages considered have usually been built so as to be able to represent any pure quantum evolution, i.e. any perfectly isolated quantum system which hence does not interact with the environment. In this case, the language is called universal for pure quantum mechanics. The hardness of the completeness problem, as well as constraints given by the complexity to physically achieve some gates, focused the research on some restrictions of the languages. On the one hand, finite presentations for the quantum circuits were shown to be complete for some restrictions – namely Clifford [42], one-qubit Clifford+T [37], two-qubit Clifford+T [43], CNot-dihedral [1] –, however none of these restrictions is universal, nor approximately universal. Regarding the ZX-calculus, completeness results exist for non-universal restrictions of the ZX-Calculus [3, 4, 16, 24], but also for the many-qubit Clifford+T ZX-Calculus [31], which was the first completeness result for an approximately universal fragment of the language. Then complete theories have been introduced for the universal ZX-Calculus [28, 33, 32, 44] and ZW-Calculus [27, 28]. The completeness of the graphical languages for pure quantum mechanics is one of the main achievements of the categorical approach to quantum mechanics, and is the cornerstone for the application of this formalism in many areas of quantum information processing. The ZX-Calculus already proved to be useful for quantum information processing [14] (e.g. measurement-based quantum computing [18, 23, 29], quantum codes [17, 9, 20, 22], circuit optimisation [21], foundations [5, 19] ...). Moreover the ZX-calculus can be concretely used through two softwares: Quantomatic [36] and PyZX [34].

The existence of complete graphical languages beyond pure quantum mechanics for more general, not necessarily pure, quantum evolutions is an open question that we address in the present paper.

While pure quantum evolutions correspond to linear maps over Hilbert spaces, probability distributions over quantum states as well as some quantum evolutions like discarding a quantum system can be represented, following the von Neumann approach, by means of density matrices and completely positive maps. The category of completely positive maps has been already studied [39], and in particular the connections between the pure and the van Neumann approaches is a central question in categorical quantum mechanics. Selinger

introduced a construction called CPM to turn a category for pure quantum mechanics into a category for density matrices and completely positive maps [40]. Another approach to relate pure quantum mechanics to the general one is the notion of environment structure [10, 12, 15]. The notion of *purification* is central in the definition of environment structure. The CPM-construction and the environment structure approaches have been proved to be equivalent [12].

In terms of graphical languages, the environment structure approach cannot be used in a straightforward way to extend a graphical language beyond pure quantum mechanics. Roughly speaking the environment structure approach provides second order axioms which associates with any equation on arbitrary (non necessarily pure) evolutions an equivalent equation on pure evolutions. Such a second order axiom cannot be easily handled by a equational theory on diagrams. Regarding the CPM-construction, the main property which has been exploited in [14] is that CPM($\mathbf{C}$) is essentially a subcategory of $\mathbf{C}$, thus one can use a graphical language which has been designed for $\mathbf{C}$ in order to represent morphisms in CPM($\mathbf{C}$): Given a complete graphical language for $\mathbf{C}$, we can use a subset of the pure diagrams to represent the evolutions in CPM($\mathbf{C}$). The main caveat of this approach is that this subset is not necessarily closed under the equational theory on pure diagrams, and as a consequence does not provide a complete graphical language for CPM($\mathbf{C}$).

Our contributions. In [30] was shown that the category **CPTPM** of completely positive trace-preserving maps is the universal monoidal category with a terminal unit and a functor from the category of isometries. We build upon this result by introducing a new construction, the *discard construction*, which transforms any †-symmetric monoidal category into a symmetric monoidal category equipped with a discard map. Roughly speaking this construction consists in making any isometry causal. Indeed, in quantum mechanics, the isometries (linear maps U such $U^\dagger \circ U = I$) are known to be causal, i.e. applying U and then discarding the subsystem on which it has been applied is equivalent to discarding the subsystem straightaway. Specifically, the discard construction proceeds as follows: first the discard is added to the subcategory of isometries, making the unit of the tensor a terminal object in this sub-category , as pointed out in [30]. Then the discard construction is obtained as the pushout of the resulting category and the initial one.

We show that the discard construction does not always produce an environment structure for the original category, and thus is not equivalent to the CPM construction. We show that a necessary and sufficient condition for the two constructions to be equivalent is that the initial category has enough isometries. We show that most of the categories usually used in the context of the categorical quantum mechanics, like **FHilb** and **Stab**, do have enough isometries, however **Clifford+T** does not.

Finally, we show that the discard construction provides a simple recipe to extend graphical languages beyond pure quantum mechanics. We provide an extension for several graphical languages that we prove to be complete for general quantum operations.

Structure of the paper. In section 2, we review some categorical notions used in categorical quantum mechanics. Section 3 is dedicated to the definition of the discard construction and the relation with the CPM construction. Finally, in section 4 we use the discard construction to extend the ZX-calculus to make it complete for general (not necessarily pure) quantum evolutions. The construction is also applied to other graphical languages.

2 Background

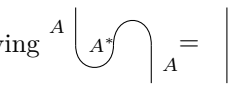
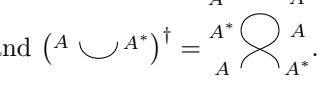
2.1 Dagger symmetric monoidal categories

To avoid any size issue, all our categories are small, the homset of a category $\mathbf{C}$ will be denoted $\mathbf{C}[A, B]$. For simplicity, all the monoidal categories considered in the following will be *strict*. Recall that a *strict symmetric monoidal category* (SMC) $\mathbf{C}$ is a category together with a tensor product bifunctor $\otimes : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$, a unit object I such that $A \otimes I = I \otimes A = A$ and $A \otimes (B \otimes C) = (A \otimes B) \otimes C$, and a symmetry natural isomorphism: $\sigma_{A,B} : A \otimes B \to B \otimes A$ satisfying $\sigma_{A,I} = 1_A$, $\sigma_{A,B\otimes C} = (1_B \otimes \sigma_{A,C}) \circ (\sigma_{A,B} \otimes 1_C)$, and $\sigma_{A,B} \circ \sigma_{B,A} = 1_{B\otimes A}$. A *prop* is an SMC whose set of objects is freely spanned by one object. There is an associated notion of strict symmetric monoidal functor $F : \mathbf{C} \to \mathbf{D}$ which preserves unit, tensors and symmetries. We will use *string diagram* notations for SMC where morphisms are described as boxes and

$g \circ f :=$  $f \otimes g :=$  $1_A :=$ $1_I :=$  $\sigma_{A,B} :=$ 

A †-SMC $\mathbf{C}$, is an SMC with an i.o.o. (identity on object) involutive and contravariant SMC-functor $(.)^\dagger : \mathbf{C} \to \mathbf{C}$. That is, every morphism $f : A \to B$ has a dagger $f^\dagger : B \to A$ such that $f^{\dagger\dagger} = f$, moreover the dagger respects the symmetries $\sigma^\dagger_{A,B} = \sigma_{B,A}$. The dagger is a central notion in categorical quantum computing and can be used to define specific properties of morphisms:

▶ **Definition 1.** *$f : A \to B$ is an* isometry *if $f^\dagger \circ f = 1_A$, i.e.* 

In this paper most of the categories considered are furthermore compact closed: A dagger compact category (†-CC) is a †-SMC where every object A has a dual object A^* such that for all objects A, there are two morphisms $^A\smile^{A^*} : A \otimes A^* \to I$ and $_{A^*}\frown_A : I \to A^* \otimes A$ satisfying  and  .

2.2 Examples

We will consider two kinds of SMCs in this paper: the categories of quantum evolutions and the graphical languages.

Quantum evolutions. Pure quantum evolutions correspond the category of Hilbert spaces. We will consider several of its subcategories: **FHilb** is the category of finite dimensional Hilbert spaces whose objects are $\mathbb{C}^n$ and morphisms are linear maps. Its tensor is the usual tensor product of vector spaces and its dagger is the adjoint with respect to the usual scalar product. It is the mathematical model for pure quantum mechanics. In quantum information processing, quantum data is usually carried by qubits, hence **Qubit** is the full subcategory of **FHilb** with objects of the form $\mathbb{C}^{2^n}$. **Stab** is the sub-category of **Qubit** which is finitely generated by the Clifford operators: H, S, CNot, the state $|0\rangle$, the projector $\langle 0|$, and the scalar 2 where:

$$\mathrm{H} = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \quad \mathrm{S} = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix} \quad \mathrm{CNot} = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix} \quad |0\rangle = \begin{pmatrix}1\\ 0\end{pmatrix} \quad \langle 0| = \begin{pmatrix}1 & 0\end{pmatrix}$$

Those are amongst the most commonly used gates in quantum computation (see [38] for details). **Clifford+T** is the same as **Stab** but with the additional generator $T = \left(\begin{smallmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{smallmatrix}\right)$, the morphisms of **Clifford+T** are exactly the matrices with entries in the ring $\mathbb{Z}[i, \frac{1}{\sqrt{2}}]$ [31]. Contrary to **Stab**, **Clifford+T** is *approximately universal* in the sense that $\forall n, m \in \mathbb{N}$, $\forall f \in \mathbf{Qubit}[\mathbb{C}^{2^n}, \mathbb{C}^{2^m}]$ and $\forall \epsilon > 0$, there exists $g \in \mathbf{Clifford+T}[\mathbb{C}^{2^n}, \mathbb{C}^{2^m}]$ such that $||f - g|| < \epsilon$. **FHilb**, **Qubit**, **Clifford+T**, and **Stab** are all †-CC. Notice that **Qubit**, **Clifford+T**, and **Stab** are props, but **FHilb** is not.

Probability distributions over pure quantum states as well as some quantum evolutions like discarding a quantum system are not *pure* but can be represented, following the von Neumann approach, by means of density matrices and completely positive maps. Let **CPM** be the category of completely positive maps of finite dimension whose objects are $\mathbb{C}^n$ and $\mathbf{CPM}[\mathbb{C}^n, \mathbb{C}^m] = \{U : \mathbb{C}^{n\times n} \to \mathbb{C}^{m\times m} \mid U \text{ is a completely positive linear map}\}$. Similarly to the pure case, one can define various subcategories of **CPM**. Notice that it can be achieved by the CPM construction described in the next section.

Graphical languages. The second kind of categories we are considering in this paper are graphical languages. They are props which come with interpretation functors defining their semantics. A prop is in fact the equivalent of Lawvere theories for symmetric monoidal theories. They can be presented by generators and relations as one would do for usual theories, see [45] and [7] for a detailed discussion.

▶ **Definition 2.** *A* graphical language $\mathcal{G}$ *is a prop presented by a set of* generators Σ *and a set of* equations E *together with a function* $[\![.]\!] : \Sigma \to \hom(S)$ *called the* interpretation *of* $\mathcal{G}$ *in* S*.* $\mathcal{G}$ *is said to be* sound *if* $[\![.]\!]$ *defines an interpretation functor* $[\![.]\!] : \mathcal{G} \to S$*, and* universal *(resp.* complete*) when this functor is surjective (resp. faithful).*

The ZX-, ZW- and ZH-calculi or the quantum circuits are examples of such categories with semantics in **Qubit**.

2.3 Environment structures and CPM-construction

Connecting the Hilbert approach – for pure quantum mechanics – and the von Neumann approach – for open systems – is a central question in categorical quantum mechanics. Selinger pointed out that any †-CC for pure quantum mechanics can be turned into a category for density matrices and completely positive maps via the CPM construction [40]:

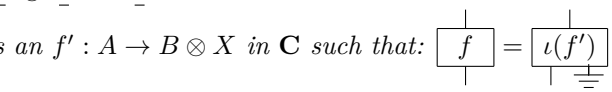
▶ **Definition 3.** *Given a* †*-CC* $\mathbf{C}$*, let* $\mathrm{CPM}(\mathbf{C})$ *be the* †*-CC with the same objects as* $\mathbf{C}$ *such that* $\mathrm{CPM}(\mathbf{C})[A, B] = \left\{ \begin{array}{c} A \\ \boxed{f} \\ B \quad C \end{array} \; \begin{array}{c} A^* \\ \boxed{f^*} \\ C^* \quad B^* \end{array} , f \in \mathbf{C}[A, B \otimes C] \right\}$*, where* $\begin{array}{c} A^* \\ \boxed{g^*} \\ B^* \end{array} := \begin{array}{c} \quad B \\ \boxed{g^\dagger} \\ B^* \quad A \end{array} A^*$.

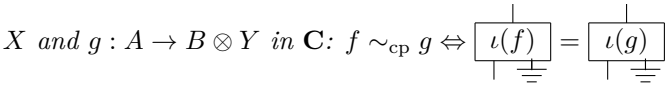
Applying it to **FHilb** one obtains the category **CPM** of completely positives maps. The CPM construction can also be applied to **Qubit**, **Clifford+T**, and **Stab**. Notice that the CPM-construction has been then extended to the non necessarily compact categories [12].

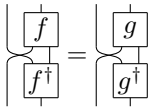
Another approach to relate pure quantum mechanics to the general one is the notion of environment structure [10, 12, 15]. The notion of *purification* is central in the definition of environment structure. Intuitively, it means that (1) there is a discard morphism for every object; (2) any morphism can be purified, i.e. decomposed into a pure morphism followed by a discarding map, and (3) this purification is essentially unique. More formally:

▶ **Definition 4.** *An environment structure for a †-CC* $\mathbf{C}$ *is a CC* $\overline{\mathbf{C}}$ *with the same objects as* $\mathbf{C}$*, an i.o.o SMC-functor* $\iota : \mathbf{C} \to \overline{\mathbf{C}}$ *and for each object* A *a morphism* $\perp_A : A \to I$ *such that:*

(1) $\perp_I = 1_I$*, and for all* $A, B : \mathbf{C}$, $\perp_A \otimes \perp_B = \perp_{A\otimes B}$.

(2) *For all* $f : A \to B$ *in* $\overline{\mathbf{C}}$*, there is an* $f' : A \to B \otimes X$ *in* $\mathbf{C}$ *such that:* 

(3) *For any* $f : A \to B \otimes X$ *and* $g : A \to B \otimes Y$ *in* $\mathbf{C}$*:* $f \sim_{\text{cp}} g \Leftrightarrow$ 

where the relation $\sim_{\text{cp}}$ *is defined as:* $f \sim_{\text{cp}} g \Leftrightarrow$ 

Notice that $\sim_{\text{cp}}$ is technically not a relation on morphisms but on tuples (A, B, X, f) with $f \in \mathbf{C}[A, B \otimes X]$: $(A, B, X, f) \sim_{\text{cp}} (C, D, Y, g)$ if $A = C, B = D$ and f, g satisfy the graphical condition represented above. By abuse of notation, we write $f \sim_{\text{cp}} g$, as the other components of the tuple will be usually obvious from context. We will do the same for our relation $\sim_{\text{iso}}$ below.

CPM is actually an environment structure for the category **FHilb**, and more generally for any †-CC $\mathbf{C}$, CPM($\mathbf{C}$) is an environment structure for $\mathbf{C}$ and conversely any environment structure for $\mathbf{C}$ is equivalent to CPM($\mathbf{C}$) [12]. Actually one can notice that CPM($\mathbf{C}$)$[A, B]$ is nothing but the set of equivalent classes of $\sim_{\text{cp}}$.

The notion of environment structures has also been generalised to the non compact case [12]. We chose here to focus on the compact case.

3 The Discard Construction

We introduce a new construction, the *discard construction*, which consists in adding a discard map for every object of a †-SMC, and thus intuitively transforming a category for pure quantum mechanics into a category for general quantum evolutions.

Causality is a central notion in quantum mechanics which has been axiomatised using a discard map as follows [35]: $f : A \to B$ is *causal* if and only if  $= \perp$. Amongst the pure quantum evolutions, the isometries are causal evolutions. The discard construction essentially consists in making any isometry causal. Thus, whereas the CPM construction relies on completely positive maps and the environment structures on the concept of purification, the discard construction relies on causality.

3.1 Definition

We introduce the new construction in three steps. First, given a †-SMC, one can consider its subcategory of isometries:

▶ **Definition 5.** *Given a †-SMC* $\mathbf{C}$*,* $\mathbf{C}_{\text{iso}}$ *is the subcategory with the same objects as* $\mathbf{C}$ *and isometries as morphisms, i.e. for all* $A, B : \mathbf{C}$*,* $\mathbf{C}_{\text{iso}}[A, B] = \{f : \mathbf{C}[A, B], f^\dagger \circ f = 1_A\}$.

Notice that $\mathbf{C}_{\text{iso}}$ is an SMC but usually not a †-SMC. Any †-SMC-functor $F : \mathbf{C} \to \mathbf{D}$ between two †-SMC can be restricted to their subcategories of isometries leading to an SMC-functor $F_{\text{iso}} : \mathbf{C}_{\text{iso}} \to \mathbf{D}_{\text{iso}}$. Thus there is a restriction functor iso : †-**SMC** → **SMC**. Remark that this functor preserves fullness and faithfulness. One always has an inclusion i.o.o. faithful SMC-functor: $i_{\text{iso}} : \mathbf{C}_{\text{iso}} \to \mathbf{C}$.

In quantum mechanics, isometries are causal evolutions, i.e. applying an isometry and then discarding all outputs is equivalent to discarding the inputs straight away. As pointed out in [30], adding discard maps to the category of isometries would make I a terminal object. Such a category is said to be *affine symmetric monoidal category* (ASMC). We define the affine completion of an SMC:

▶ **Definition 6.** *Given an SMC* $\mathbf{C}$*, we define* $\mathbf{C}^!$ *as* $\mathbf{C}$ *with an additional morphism* $!_A : A \to I$ *for each object* $A : \mathbf{C}$*. We denote the inclusion functor* $i^! : \mathbf{C} \to \mathbf{C}^!$ *which is strict monoidal and i.o.o. We further impose that* $1_I = !_I$*, and that for all* $f : \mathbf{C}[A, B]$*,* $!_B \circ i^!(f) = !_A$*. This makes* I *a terminal object in* $\mathbf{C}^!$*, and thus* $\mathbf{C}^!$ *is an ASMC.*

Notice by the way that $!_A \otimes !_B = 1_I \circ (!_A \otimes !_B) = !_I \circ (!_A \otimes !_B) = !_{A\otimes B}$. Again given a functor $F : \mathbf{C} \to \mathbf{D}$, one can define a functor $F^! : \mathbf{C}^! \to \mathbf{D}^!$ by $F^!(!_A) = !_{i^!(F(A))}$ and $F^!(f) = i^!(F(f))$ for the other morphisms. In [30], Huot and Staton show that **CPTPM**, the category of completely positive trace preserving maps, is equivalent to $\mathbf{FHilb}^!_{\text{iso}}$, thus giving a caracterisation of it via a universal property. We extend this idea to non-trace preserving maps by proceeding to a local affine completion of the subcategory of isometries.

We define the category $\mathbf{C}^{\fallingdotseq}$ as the pushout of $\mathbf{C}$ and $\mathbf{C}^!_{\text{iso}}$:

▶ **Definition 7.** *Given a* †*-SMC* $\mathbf{C}$*,* $\mathbf{C}^{\fallingdotseq}$ *is defined as the pushout in the category of symmetric monoidal categories:*

$$\begin{array}{ccc} \mathbf{C}_{\text{iso}} & \xrightarrow{i_{\text{iso}}} & \mathbf{C} \\ {\scriptstyle i^!}\downarrow & & \downarrow{\scriptstyle \iota_{\mathbf{C}}} \\ \mathbf{C}^!_{\text{iso}} & \xrightarrow[\iota_{\mathbf{C}^!_{\text{iso}}}]{} & \mathbf{C}^{\fallingdotseq} \end{array}$$

The existence of this pushout follows from the fact that the forgetful functor from strict symmetric monoidal categories to categories **StrictSymMonCat** → **Cat** preserves coequalizers, and from [8, Theorem 9.3.9]. As all our functors are i.o.o., we can also describe it simply combinatorially. The objects of $\mathbf{C}^{\fallingdotseq}$ are the same as $\mathbf{C}$. Its morphisms are equivalence classes generated by formal composition and tensoring of morphisms in $\mathbf{C}^!_{\text{iso}}$ and $\mathbf{C}$. The equivalence relation is generated by the equations of both categories augmented with equations $i^!(f) = i_{\text{iso}}(f)$ for all f in $\mathbf{C}_{\text{iso}}$. The functors $\iota_{\mathbf{C}}$ and $\iota_{\mathbf{C}^!_{\text{iso}}}$ are the natural ways to embed $\mathbf{C}$ and $\mathbf{C}^!_{\text{iso}}$. We will see those formal compositions as string diagrams whose components are morphisms of $\mathbf{C}$ and $\mathbf{C}^!_{\text{iso}}$ wired to each others. Two diagrams represent the same morphism if we can rewrite one into the other applying the equations of both categories and $i^!(f) = i_{\text{iso}}(f)$ for all f in $\mathbf{C}_{\text{iso}}$. This forms a well defined SMC.

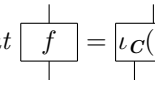
Since the only morphisms in $\mathbf{C}_{\text{iso}}$ which are not identified with the morphisms of $\mathbf{C}$ are those that contain $!_A$, we can see $\mathbf{C}^{\fallingdotseq}$ as $\mathbf{C}$ augmented with discard maps which delete isometries.

▶ **Definition 8.** *The discard map on an object* A *is defined in* $\mathbf{C}^{\fallingdotseq}$ *by* $\overset{A}{\perp\!\!\!\fallingdotseq} := \iota_{\mathbf{C}^!_{\text{iso}}}(!_A)$.

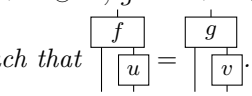
Notice, that for any isometry $f : A \to B$ in $\mathbf{C}^{\fallingdotseq}$, $\boxed{f}$ $= \perp\!\!\!\fallingdotseq$, thus any isometry is causal.

3.2 Relation to environment structures and CPM

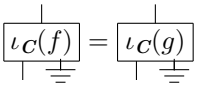
In order to compare the $\mathbf{C}^{\fallingdotseq}$ construction with environment structures and the CPM construction we need to study in details the purification process in $\mathbf{C}^{\fallingdotseq}$. First notice that any morphism of $\mathbf{C}^{\fallingdotseq}$ admits a purification:

▶ **Lemma 9.** *Let* $\mathbf{C}$ *be a* †*-SMC. For all* $f : \mathbf{C}^{\neq}[A, B]$*, there is an* $X : \mathbf{C}$ *and an* $f' : \mathbf{C}[A, B \otimes X]$ *such that*  $= \iota_{\mathbf{C}}(f')$.

The purification needs not be unique, however it satisfies an essential uniqueness condition. To state it we define the relation $\sim_{\text{iso}}$.:

▶ **Definition 10.** *Let* $\mathbf{C}$ *be a* †*-SMC, and two morphisms* $f : A \to B \otimes X$*,* $g : A \to B \otimes Y$*.* $f \sim_{\text{iso}} g$ *if there are two isometries* $u : X \to Z$ *and* $v : Y \to Z$*, such that*  .

Notice that the relation $\sim_{\text{iso}}$ is not transitive, thus we consider $\sim_{\text{iso}}^{+}$ its transitive closure to make it an equivalence relation. It is easy to show that if $f \sim_{\text{iso}}^{+} g$ then f and g purify the same morphism of $\mathbf{C}^{\neq}$. The converse is also true:

▶ **Lemma 11.** *For all* $f : A \to B \otimes X$ *and* $g : A \to B \otimes Y$*:* $f \sim_{\text{iso}}^{+} g \Leftrightarrow$ 

So the purification is unique up to $\sim_{\text{iso}}^{+}$. Lemma 11 also gives an alternative definition of $\mathbf{C}^{\neq}$ which relates more easily to the CPM construction. It is the same construction as CPM with $\sim_{\text{cp}}$ replaced by $\sim_{\text{iso}}^{+}$. In other words $\mathbf{C}^{\neq}[A, B]$ is the set of equivalent classes of $\sim_{\text{iso}}^{+}$.

As we have introduced a new discard construction, a natural question is whether $\mathbf{C}^{\neq}$ is an environment structure for $\mathbf{C}$. To be an environment structure, three conditions are required. The first two are satisfied: $\mathbf{C}^{\neq}$ has a discard morphism for every object, and every morphism can be purified. The third one is the uniqueness of the purification: according to the definition of the environment structures, f and g purify the same morphism if and only if $f \sim_{\text{cp}} g$ whereas according to Lemma 11, f and g purify the same morphism if and only if $f \sim_{\text{iso}}^{+} g$. As a consequence $\mathbf{C}^{\neq}$ is an environment structure for $\mathbf{C}$ if and only if $\sim_{\text{cp}} = \sim_{\text{iso}}^{+}$. It turns out that one of the inclusions is always true:

▶ **Lemma 12.** *For any* †*-SMC category* $\mathbf{C}$*, we have* $\sim_{\text{iso}}^{+} \subseteq \sim_{\text{cp}}$.

As a consequence, if $\sim_{\text{cp}} \neq \sim_{\text{iso}}^{+}$, it means that there are some morphisms f, g that are equal in $\sim_{\text{cp}}$ but cannot be proved equal in $\sim_{\text{iso}}^{+}$. Intuitively it means the category has not enough isometries to prove those terms equal, which leads to the following definition:

▶ **Definition 13.** *A* †*-SMC category* $\mathbf{C}$ *has* enough isometries *if the equivalences relations* $\sim_{\text{cp}}$ *and* $\sim_{\text{iso}}^{+}$ *of* $\mathbf{C}$ *are equal.*

▶ **Lemma 14.** *Given a* †*-SMC* $\mathbf{C}$*, the following properties are equivalent:*

1. $\mathbf{C}$ *has enough isometries;*
2. $\mathbf{C}^{\neq}$ *is an environment structure for* $\mathbf{C}$*;*
3. $\mathbf{C}^{\neq} \simeq \text{CPM}(\mathbf{C})$.

Notice that if $\mathbf{C}$ has enough isometries, the discard construction provides a definition of $\text{CPM}(\mathbf{C})$ via a universal property. This gives a more direct way to build the environment, avoiding to deal with the equivalence classes of the CPM construction.

▶ Remark 15. Let's focus for a moment on the category $\text{Causal}\,\text{CPM}(\mathbf{C})$ of causal maps, that is the subcategory of maps cancelled by the discards in $\text{CPM}(\mathbf{C})$. We have that: $\sim_{\text{cp}} \subseteq \sim_{\text{iso}}^{+} \Rightarrow \mathbf{C}_{\text{iso}}^{!} \simeq \text{Causal}\,\text{CPM}(\mathbf{C})$. In fact by Lemma 14, $\text{CPM}(\mathbf{C}) \simeq \mathbf{C}^{\neq}$, and then the subcategory $\text{Causal}\,\text{CPM}(\mathbf{C})$ is equivalent to the subcategory of maps cancelled by the discards in $\mathbf{C}^{\neq}$ which is equivalent to $\mathbf{C}_{\text{iso}}^{!}$. $\text{Causal}\,\text{CPM}(\mathbf{FHilb})$ being exactly $\mathbf{CPTPM}$, we have recovered the result of [30].

3.3 Examples

We consider the usual subcategories of **FHilb** used for pure quantum mechnanics and show in each case whether the discard construction produces an environment structure or not. First of all, thanks to the Stinespring dilation theorem, $\mathbf{FHilb}^{\doteqdot}$ is not only an environment structure for **FHilb**, but the relation $\sim_{\text{iso}}$ is also transitive in this case:

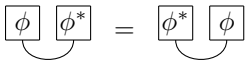
▶ **Proposition 16.** $\mathbf{FHilb}^{\doteqdot}$ *is an environment structure for* **FHilb**. *Furthermore* $\sim_{\text{iso}}^{+}=\sim_{\text{iso}}$.

When dealing with graphical languages we will be more interested in the full subcategory **Qubit** of **FHilb**:

▶ **Proposition 17.** $\mathbf{Qubit}^{\doteqdot}$ *is an environment structure for* **Qubit**.

Notice that in general, the property of having enough isometries does not transfer to full subcategories: If **D** is a full subcategory of **C**, we might have $f \sim_{\text{iso}}^{+} g$ on **C** but $f \not\sim_{\text{iso}}^{+} g$ on **D**. This could happen for two reasons: First the chain of intermediate morphisms that prove that $f \sim_{\text{iso}}^{+} g$ might live outside of **D**. Second, the isometries that "prove" that $f \sim_{\text{iso}}^{+} g$ on **C** might have codomain outside of **D**.

If our category is not a full subcategory, then everything falls apart, and finding conditions that guarantee that $\mathbf{C}^{\doteqdot}$ is an environment structure for **C** is not easy.

For subcategories of **Qubit**, necessary conditions can be given. This category has the peculiarity that $\cdot^*$ is the identity on objects and that $f^{**} = f$ for all morphisms ($\cdot^*$ maps a matrix to its conjugate matrix). In particular, for any state $\phi : I \to I \otimes X$, we have $\phi^* \sim_{\text{cp}} \phi$. Indeed  .

So a necessary condition for a subcategory of **Qubit** to behave nicely is that for all states ϕ, we have $\phi^* \sim_{\text{iso}}^{+} \phi$. This is the case in **Stab**: Given a stabilizer state ϕ, there always exists a stabilizable unitary U s.t. $U\phi = \phi^*$. In fact:

▶ **Proposition 18.** $\mathbf{Stab}^{\doteqdot}$ *is an environment structure for* **Stab**.

The main idea of the proof is to use the map/state duality, and structural results about bipartite stabilizer states [2].

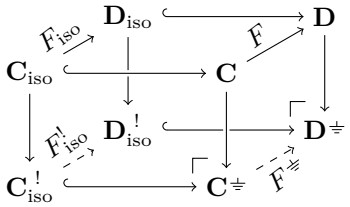
No such unitary exists in general in **Clifford+T**: For almost all states ϕ, there is no unitary U (and actually no morphism at all) s.t. $U\phi = \phi^*$. **Clifford+T** therefore has not got enough isometries:

▶ **Proposition 19.** $(\mathbf{Clifford+T})^{\doteqdot}$ *is not an environment structure for* **Clifford+T**. *More precisely, there exists a state* ϕ *s.t.* $\phi \sim_{\text{cp}} \phi^*$ *but* $\phi \not\sim_{\text{iso}}^{+} \phi^*$. *One can take for example* $\phi = 1 + 2i$ *(in this case* ϕ *is a state with no input and outputs, hence a scalar).*

Note that for all categories above, we have $\sim_{\text{iso}}^{+}=\sim_{\text{iso}}$. That it holds in **Qubit** and **FHilb** is a consequence of the Witt extension theorem: Every isometry $f : A \to B$ is equal to a unitary $g : B \to B$ precomposed with a canonical embedding from A to B. It it well known in **Stab** and it is true in **Clifford** + **T** by [25, Lemma 5].

4 Application to the ZX-Calculus and other graphical languages

We now focus on the behavior of interpretation functors with respect to the discard construction. The discard construction defines a functor $(\_)^{\doteqdot} : \dagger\text{-}\mathbf{SMC} \to \mathbf{SMC}$. Indeed, given a †-SMC functor F, F_{iso} and $F_{\text{iso}}^{!}$ uniquely define a functor $F^{\doteqdot}$ by pushout.



The following lemma and theorem are the main tools to apply the discard construction to graphical languages:

▶ **Lemma 20.** *If F is faithful and if $F_{\text{iso}} : \mathbf{C}_{\text{iso}} \to \mathbf{D}_{\text{iso}}$ is surjective, then $F(f) \sim^+_{\text{iso}} F(g) \Rightarrow f \sim^+_{\text{iso}} g$.*

▶ **Theorem 21.** *Let $\mathbf{C}$ and $\mathbf{D}$ be two †-SMCs and $F : \mathbf{C} \to \mathbf{D}$ a †-SMC-functor. If F is faithful and if $F_{\text{iso}} : \mathbf{C}_{\text{iso}} \to \mathbf{D}_{\text{iso}}$ is surjective, then $F^{\doteq} : \mathbf{C}^{\doteq} \to \mathbf{D}^{\doteq}$ is faithful. If furthermore F is surjective then $F^{\doteq}$ is surjective and faithful.*

Notice that the hypothesis on F_{iso} is very strong, as it makes it an isomorphism: We want it to be surjective as we do not want to lose even one isometry. In particular we do not know if the theorem still applies if F is merely an equivalence of categories.

Reformulating for graphical languages this gives:

▶ **Corollary 22** (of Theorem 21)**.** *Given a †-CC $\mathbf{C}$ with enough isometries, if $\mathcal{G}$ is a †-CC universal complete graphical language for $\mathbf{C}$ then $\mathcal{G}^{\doteq}$ is a universal complete language for* CPM($\mathbf{C}$)*.*

This provides a general recipe. We start with a universal complete graphical language $\mathcal{G}$. We build $\mathcal{G}^{\doteq}$, by Theorem 21, $[\![.]\!]^{\doteq} : \mathcal{G}^{\doteq} \to \mathbf{C}^{\doteq}$ is full and faithful. Furthermore $\mathbf{C}^{\doteq} \simeq \text{CPM}(\mathbf{C})$. $\mathcal{G}^{\doteq}$ as a prop can be presented by adding one new generator $\frac{\perp}{=}$ to the signature Σ and one equation for each isometry of $\mathcal{G}$. In general, if one is provided with a spanning set of the isometries, the number of equations can be drastically reduced. We just need one equation for each element of this set. We then obtain a universal complete graphical language.

We will now briefly review the ZX-calculus and some of its twin languages. They are all universal and complete for subcategories of **Qubit**. Each time we will apply the recipe with a well chosen spanning set and provide the additional axioms involving $\frac{\perp}{=}$. We will not discuss minimality, i.e. if adding these new axioms can help to simplify others.

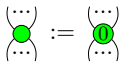
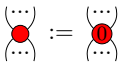
4.1 The ZX-calculus

The ZX-Calculus was introduced in [11] by Coecke and Duncan for pure quantum evolutions. It is a †-compact prop generated by:

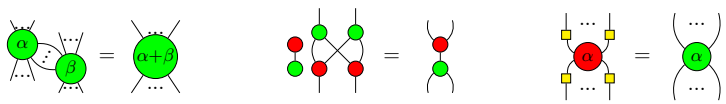
$R_Z^{(n,m)}(\alpha) : n \to m ::$  $\qquad R_X^{(n,m)}(\alpha) : n \to m ::$  $\qquad H : 1 \to 1 ::$

and the two compositions: spacial $(. \otimes .)$ and sequential $(. \circ .)$. The symmetric and compact structure are provided by $\sigma : 2 \to 2 ::$  , $\epsilon : 2 \to 0 ::$  and $\eta : 0 \to 2 ::$.

To simplify, the red and green nodes are represented empty when holding a 0 angle:

 $\qquad$ and $\qquad$ 

The language is universal [11]. So far, it has two complete axiomatisations [28, 33]. Some of the main axioms are:



ZX-diagrams represent quantum evolutions, so there exists a functor $[\![.]\!] : \mathrm{ZX} \to \mathbf{Qubit}$, called the *standard interpretation*, which associates to any diagram $D : n \to m$ a linear map $[\![D]\!] : \mathbb{C}^{2^n} \to \mathbb{C}^{2^m}$ inductively defined as follows:

$[\![.]\!]$

$$[\![D_1 \otimes D_2]\!] := [\![D_1]\!] \otimes [\![D_2]\!] \qquad [\![D_2 \circ D_1]\!] := [\![D_2]\!] \circ [\![D_1]\!]$$

$$[\![\text{⊓}]\!] := (1) \qquad [\![\text{|}]\!] := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad [\![\text{⏚}]\!] := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

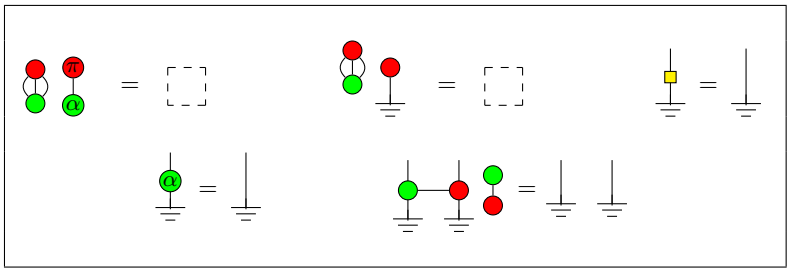
$$[\![\cup]\!] := (1 \ 0 \ 0 \ 1) \qquad [\![\times]\!] := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad [\![\cap]\!] := \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$[\![\alpha]\!] := (1 + e^{i\alpha}) \qquad \left[\!\!\left[\begin{smallmatrix} n \\ \alpha \\ m \end{smallmatrix} \right]\!\!\right] := 2^m \begin{pmatrix} \overbrace{\begin{matrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{matrix}}^{2^n} \\ \begin{matrix} \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & e^{i\alpha} \end{matrix} \end{pmatrix} \quad (n+m>0)$$

For any $n, m \geq 0$ and $\alpha \in \mathbb{R}$: $\left[\!\!\left[\begin{smallmatrix} n \\ \alpha \\ m \end{smallmatrix} \right]\!\!\right] = [\![\text{⏚}]\!]^{\otimes m} \circ \left[\!\!\left[\begin{smallmatrix} n \\ \alpha \\ m \end{smallmatrix} \right]\!\!\right] \circ [\![\text{⏚}]\!]^{\otimes n}$
$\big($where $M^{\otimes 0} = (1)$ and $M^{\otimes k} = M \otimes M^{\otimes k-1}$ for $k \in \mathbb{N}^*\big)$.

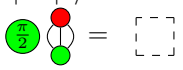
Theorem 21 provides a recipe for transforming the language for mixed states and CPMs. The resulting language $\mathrm{ZX}^{\doteq}$ can be seen as a prop with the generators of the ZX-Calculus, augmented with ⏚ and with the axiomatisation enriched with $\{⏚ \circ D = ⏚ \mid D^\dagger \circ D = I\}$. We actually do not need an infinite axiomatisation. Indeed, the set of isometries of the ZX-Calculus can be finitely generated.

Using ($e^{i\alpha}$, $|0\rangle$, H, $R_Z(\alpha)$, CNot) as spanning set of the isometries [38], we obtain only five axioms:



4.2 The $\frac{\pi}{2}$ fragment of ZX-calculus

The $\mathrm{ZX}_{\frac{\pi}{2}}$ is obtained from ZX by restricting phases α to $\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$. It is universal and complete for **Stab** [3] with the adequate axiomatisation. Moreover according to Lemma 18 $\mathbf{Stab}^{\doteq}$ is an environment structure for **Stab**.

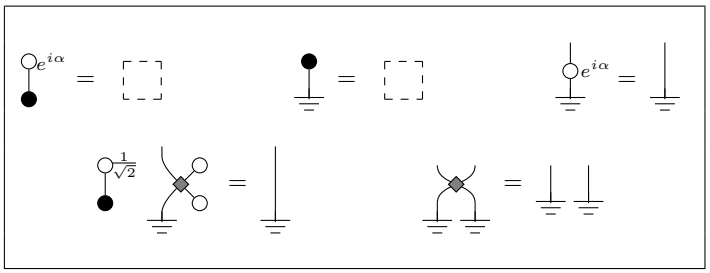
The set ($e^{i\frac{\pi}{4}}$, $|0\rangle$, H, $R_Z(\alpha)$, CNot), with α restricted to multiples of $\frac{\pi}{2}$, is a spanning set of isometries in **Stab** (notice that $e^{i\frac{\pi}{4}} = 2\langle 0|HSH|0\rangle\langle 0|H|0\rangle$ is in **Stab**), so adding the same set of equations than in $\mathrm{ZX}^{\doteq}$ with additional rule  will provide a complete axiomatisation for $\mathrm{ZX}^{\doteq}_{\frac{\pi}{2}}$.

4.3 The Clifford+T fragment of ZX-calculus

Restricting ZX to angles multiples of $\pi/4$, we obtain a language which is known to be universal and complete for **Clifford+T** [31]. However, as shown by Lemma 19, the semantic category **Clifford+T** does not have enough isometries. The discard construction is strictly coarser than CPM for this fragment. So we leave open the complete axiomatisation of quantum operations for this fragment.

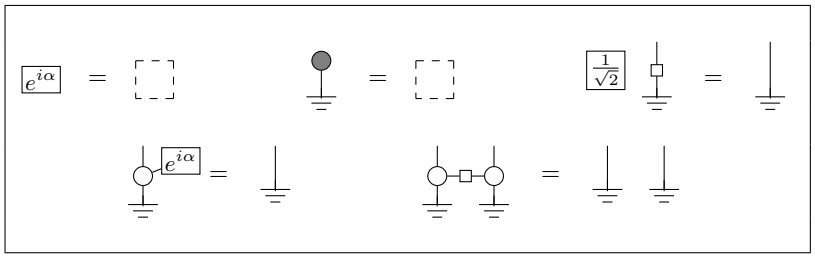
4.4 The ZW-calculus

The ZW-Calculus was introduced in [26], deriving from the GHZ/W-Calculus [13], where the main two generators are two non-equivalent ways to entangle three qubits, the so-called GHZ and W states. The language was made complete for pure quantum mechanics in [28]. Since CNot is hard to express in this calculus, we choose another set of universal diagrams, more suited to ZW, namely ($e^{i\alpha}$, $|1\rangle$, $R_Z(\alpha)$, H, CZ $\circ$ SWAP). The resulting rules for $\mathrm{ZW}^{\doteq}$ are:



4.5 The ZH-Calculus

The ZH-Calculus was introduced and proved to be complete in [6]. The point of this language is to easily represent hypergraph-states, a generalisation of graph-states, a useful resource for quantum computing. This language has been specifically designed to easily represent the multi-controlled Z (which constitute the hyperedges in the hypergraph-states). So in particular, CZ and $R_Z(\alpha)$ are easily representable. Up to a scalar, H is also easily doable, and $[\![X^{(0,1)}]\!] = |0\rangle$. Hence, choosing ($e^{i\alpha}$, $|0\rangle$, H, $R_Z(\alpha)$, CZ) as spanning set, we only need the axioms:



References

1 Matthew Amy, Jianxin Chen, and Neil J. Ross. A Finite Presentation of CNOT-Dihedral Operators. In Bob Coecke and Aleks Kissinger, editors, *Proceedings 14th International Conference on Quantum Physics and Logic, Nijmegen, The Netherlands, 3-7 July 2017*, volume 266 of *Electronic Proceedings in Theoretical Computer Science*, pages 84–97. Open Publishing Association, 2018. doi:10.4204/EPTCS.266.5.

2 Koenraad M. R. Audenaert and Martin B. Plenio. Entanglement on Mixed Stabilizer States: Normal Forms and Reduction Procedures. *New Journal of Physics*, 7:170–170, August 2005. doi:10.1088/1367-2630/7/1/170.

3 Miriam Backens. The ZX-Calculus is Complete for Stabilizer Quantum Mechanics. *New Journal of Physics*, 16(9):093021, September 2014. doi:10.1088/1367-2630/16/9/093021.

4 Miriam Backens. The ZX-Calculus is Complete for the Single-Qubit Clifford+T group. *Electronic Proceedings in Theoretical Computer Science*, 172:293–303, December 2014. doi:10.4204/eptcs.172.21.

5 Miriam Backens and Ali Nabi Duman. A Complete Graphical Calculus for Spekkens' Toy Bit Theory. *Foundations of Physics*, pages 1–34, 2014. doi:10.1007/s10701-015-9957-7.

6 Miriam Backens and Aleks Kissinger. ZH: A Complete Graphical Calculus for Quantum Computations Involving Classical Non-linearity. In Peter Selinger and Giulio Chiribella, editors, *Proceedings of the 15th International Conference on Quantum Physics and Logic, Halifax, Canada, 3-7th June 2018*, volume 287 of *Electronic Proceedings in Theoretical Computer Science*, pages 23–42. Open Publishing Association, 2019. doi:10.4204/EPTCS.287.2.

7 John C. Baez, Brandon Coya, and Franciscus Rebro. Props in Network Theory. In *Theory and Applications of Categories*, volume 33 (25), pages 727–783, July 2017. arXiv:1707.08321.

8 Michael Barr and Charles Wells. *Toposes, Triples and Theories*. Springer-Verlag, New York, 1985. URL: http://www.tac.mta.ca/tac/reprints/articles/12/tr12abs.html.

9 Nicholas Chancellor, Aleks Kissinger, Joschka Roffe, Stefan Zohren, and Dominic Horsman. Graphical Structures for Design and Verification of Quantum Error Correction. last revised Jan. 2018, 2016. arXiv:1611.08012.

10 Bob Coecke. Axiomatic Description of Mixed States from Selinger's CPM-Construction. *Electronic Notes in Theoretical Computer Science*, 210:3–13, 2008. Proceedings of the 4th International Workshop on Quantum Programming Languages (QPL 2006). doi:10.1016/j.entcs.2008.04.014.

11 Bob Coecke and Ross Duncan. Interacting Quantum Observables: Categorical Algebra and Diagrammatics. *New Journal of Physics*, 13(4):043016, April 2011. doi:10.1088/1367-2630/13/4/043016.

12 Bob Coecke and Chris Heunen. Pictures of Complete Positivity in Arbitrary Dimension. *Information and Computation*, 250:50–58, 2016.

13 Bob Coecke and Aleks Kissinger. The Compositional Structure of Multipartite Quantum Entanglement. In *Automata, Languages and Programming*, pages 297–308. Springer Berlin Heidelberg, 2010. doi:10.1007/978-3-642-14162-1_25.

14 Bob Coecke and Aleks Kissinger. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, 2017. doi:10.1017/9781316219317.

15 Bob Coecke and Simon Perdrix. Environment and Classical Channels in Categorical Quantum Mechanics. *Logical Methods in Computer Science*, Volume 8, Issue 4, November 2012. doi:10.2168/LMCS-8(4:14)2012.

16 Bob Coecke and Quanlong Wang. ZX-rules for 2-qubit Clifford+T quantum circuits, 2018.

17 Niel de Beaudrap and Dominic Horsman. The ZX-Calculus is a Language for Surface Code Lattice Surgery. *CoRR*, abs/1704.08670, 2017. arXiv:1704.08670.

18 Ross Duncan. A Graphical Approach to Measurement-Based Quantum Computing. In *Quantum Physics and Linguistics*, pages 50–89. Oxford University Press, February 2013. doi:10.1093/acprof:oso/9780199646296.003.0003.

19 Ross Duncan and Kevin Dunne. Interacting Frobenius Algebras Are Hopf. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS 2016, pages 535–544, New York, NY, USA, 2016. ACM. doi:10.1145/2933575.2934550.

20 Ross Duncan and Liam Garvie. Verifying the Smallest Interesting Colour Code with Quantomatic. In Bob Coecke and Aleks Kissinger, editors, *Proceedings 14th International Conference on Quantum Physics and Logic, Nijmegen, The Netherlands, 3-7 July 2017*, volume 266 of *Electronic Proceedings in Theoretical Computer Science*, pages 147–163. Open Publishing Association, 2018. doi:10.4204/EPTCS.266.10.

21 Ross Duncan, Aleks Kissinger, Simon Perdrix, and John van de Wetering. Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus, 2019.

22 Ross Duncan and Maxime Lucas. Verifying the Steane code with Quantomatic. *Electronic Proceedings in Theoretical Computer Science*, 171:33–49, December 2014. doi:10.4204/eptcs.171.4.

23 Ross Duncan and Simon Perdrix. Rewriting Measurement-Based Quantum Computations with Generalised Flow. *Lecture Notes in Computer Science*, 6199:285–296, 2010. doi:10.1007/978-3-642-14162-1_24.

24 Ross Duncan and Simon Perdrix. Pivoting Makes the ZX-Calculus Complete for Real Stabilizers. In *QPL 2013*, Electronic Proceedings in Theoretical Computer Science, pages 50–62, 2013. doi:10.4204/EPTCS.171.5.

25 Brett Giles and Peter Selinger. Exact Synthesis of Multiqubit Clifford+T Circuits. *Phys. Rev. A*, 87:032332, March 2013. doi:10.1103/PhysRevA.87.032332.

26 Amar Hadzihasanovic. A Diagrammatic Axiomatisation for Qubit Entanglement. In *2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 573–584, July 2015. doi:10.1109/LICS.2015.59.

27 Amar Hadzihasanovic. *The Algebra of Entanglement and the Geometry of Composition*. PhD thesis, University of Oxford, 2017. arXiv:1709.08086.

28 Amar Hadzihasanovic, Kang Feng Ng, and Quanlong Wang. Two Complete Axiomatisations of Pure-state Qubit Quantum Computing. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, pages 502–511, New York, NY, USA, 2018. ACM. doi:10.1145/3209108.3209128.

29 Clare Horsman. Quantum Picturalism for Topological Cluster-State Computing. *New Journal of Physics*, 13(9):095011, September 2011. doi:10.1088/1367-2630/13/9/095011.

30 Mathieu Huot and Sam Staton. Universal Properties in Quantum Theory. In Peter Selinger and Giulio Chiribella, editors, *Proceedings of the 15th International Conference on Quantum Physics and Logic, Halifax, Canada, 3-7th June 2018*, volume 287 of *Electronic Proceedings in Theoretical Computer Science*, pages 213–223. Open Publishing Association, 2019. doi:10.4204/EPTCS.287.12.

31 Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. A Complete Axiomatisation of the ZX-Calculus for Clifford+T Quantum Mechanics. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, pages 559–568, New York, NY, USA, 2018. ACM. doi:10.1145/3209108.3209131.

32 Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. A Generic Normal Form for ZX-Diagrams and Application to the Rational Angle Completeness, 2018.

33 Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. Diagrammatic Reasoning Beyond Clifford+T Quantum Mechanics. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, pages 569–578, New York, NY, USA, 2018. ACM. doi:10.1145/3209108.3209139.

34 A. Kissinger and John van de Wetering. PyZX, 2018. URL: https://github.com/Quantomatic/pyzx.

35 Aleks Kissinger and Sander Uijlen. A categorical semantics for causal structure. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12. IEEE, 2017.

36 Aleks Kissinger and Vladimir Zamdzhiev. Quantomatic: A Proof Assistant for Diagrammatic Reasoning. In Amy P. Felty and Aart Middeldorp, editors, *Automated Deduction - CADE-25*, pages 326–336, Cham, 2015. Springer International Publishing. doi:10.1007/978-3-319-21401-6_22.

37 Ken Matsumoto and Kazuyuki Amano. Representation of Quantum Circuits with Clifford and $\pi/8$ Gates, June 2008.

38 Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. doi:10.1017/CBO9780511976667.

39 Peter Selinger. Towards a Quantum Programming Language. *Mathematical. Structures in Comp. Sci.*, 14(4):527–586, August 2004. doi:10.1017/S0960129504004256.

40 Peter Selinger. Dagger Compact Closed Categories and Completely Positive Maps. *Electronic Notes in Theoretical Computer Science*, 170:139–163, March 2007. doi:10.1016/j.entcs.2006.12.018.

41 Peter Selinger. A Survey of Graphical Languages for Monoidal Categories. In *New structures for physics*, pages 289–355. Springer, 2010.

42 Peter Selinger. Generators and Relations for n-qubit Clifford Operators. *Logical Methods in Computer Science*, Volume 11, Issue 2, June 2015. doi:10.2168/LMCS-11(2:10)2015.

43 Peter Selinger and Xiaoning Bian. Relations for Clifford+T Operators on Two Qubits, 2015. URL: https://www.mathstat.dal.ca/~xbian/talks/.

44 Renaud Vilmart. A Near-Optimal Axiomatisation of ZX-Calculus for Pure Qubit Quantum Mechanics, 2018.

45 Fabio Zanasi. *Interacting Hopf Algebras – the theory of linear systems*. PhD thesis, Université de Lyon, 2015. URL: http://www.zanasi.com/fabio/#/publications.html.

Graph and String Parameters: Connections Between Pathwidth, Cutwidth and the Locality Number

Katrin Casel
Hasso Plattner Institute, University of Potsdam, Germany
Katrin.Casel@hpi.de

Joel D. Day
Department of Computer Science, Loughborough University, UK
J.Day@lboro.ac.uk

Pamela Fleischmann
Department of Computer Science, Kiel University, Germany
fpa@informatik.uni-kiel.de

Tomasz Kociumaka
Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel
Institute of Informatics, University of Warsaw, Poland
kociumaka@mimuw.edu.pl

Florin Manea
Department of Computer Science, Kiel University, Germany
flm@informatik.uni-kiel.de

Markus L. Schmid
Trier University, Germany
mlschmid@mlschmid.de

Abstract

We investigate the locality number, a recently introduced structural parameter for strings (with applications in pattern matching with variables), and its connection to two important graph-parameters, cutwidth and pathwidth. These connections allow us to show that computing the locality number is NP-hard but fixed-parameter tractable (when the locality number or the alphabet size is treated as a parameter), and can be approximated with ratio $O(\sqrt{\log \mathsf{opt}} \log n)$. As a by-product, we also relate cutwidth via the locality number to pathwidth, which is of independent interest, since it improves the best currently known approximation algorithm for cutwidth. In addition to these main results, we also consider the possibility of greedy-based approximation algorithms for the locality number.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Mathematics of computing → Combinatorics on words; Theory of computation → Approximation algorithms analysis

Keywords and phrases Graph and String Parameters, NP-Completeness, Approximation Algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.109

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version A full version of the paper is available at `https://arxiv.org/abs/1902.10983`.

Funding *Tomasz Kociumaka*: Supported by ISF grants no. 824/17 and 1278/16 and by an ERC grant MPM under the EU's Horizon 2020 Research and Innovation Programme (grant no. 683064). *Florin Manea*: Supported by the DFG grant MA 5725/2-1.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 109; pp. 109:1–109:16

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

1 Introduction

Graphs, on the one hand, and strings, on the other, are two different types of data objects and they have certain particularities. Graphs seem to be more popular in fields like classical and parameterised algorithms and complexity (due to the fact that many natural graph problems are intractable), while fields like formal languages, pattern matching, verification or compression are more concerned with strings. Moreover, both the field of graph algorithms as well as string algorithms are well established and provide rich toolboxes of algorithmic techniques, but they differ in that the former is tailored to computationally hard problems (e.g., the approach of treewidth and related parameters), while the latter focuses on providing efficient data-structures for near-linear-time algorithms. Nevertheless, it is sometimes possible to bridge this divide, i.e., by "flattening" a graph into a sequential form, or by "inflating" a string into a graph, to make use of respective algorithmic techniques otherwise not applicable. This paradigm shift may provide the necessary leverage for new algorithmic approaches.

In this paper, we are concerned with certain structural parameters (and the problems of computing them) for graphs and strings: the *cutwidth* $\mathsf{cw}(G)$ of a graph G (i.e., the maximum number of "stacked" edges if the vertices of a graph are drawn on a straight line), the *pathwidth* $\mathsf{pw}(G)$ of a graph G (i.e., the minimum width of a tree decomposition the tree structure of which is a path), and the *locality number* $\mathsf{loc}(\alpha)$ of a string α (explained in more detail in the next paragraph). By Cutwidth, Pathwidth and Loc, we denote the corresponding decision problems and with the prefix Min, we refer to the minimisation variants. The two former graph-parameters are very classical. Pathwidth is a simple (yet still hard to compute) subvariant of treewidth, which measures how much a graph resembles a path. The problems Pathwidth and MinPathwidth are intensively studied (in terms of exact, parameterised and approximation algorithms) and have numerous applications (see the surveys and textbook [10, 34, 8]). Cutwidth is the best-known example of a whole class of so-called *graph layout problems* (see the survey [17, 39] for detailed information), which are studied since the 1970s and were originally motivated by questions of circuit layouts.

The locality number is rather new and we shall discuss it in more detail. A word is k-local if there exists an order of its symbols such that, if we *mark* the symbols in the respective order (which is called a *marking sequence*), at each stage there are at most k contiguous blocks of marked symbols in the word. This k is called the *marking number* of that marking sequence. The *locality number* of a word is the smallest k for which that word is k-local, or, in other words, the minimum marking number over all marking sequences. For example, the marking sequence $\sigma = (\mathtt{x}, \mathtt{y}, \mathtt{z})$ marks $\alpha = \mathtt{xyxyzxz}$ as follows (marked blocks are illustrated by overlines): $\mathtt{xyxyzxz}$, $\overline{\mathtt{x}}\mathtt{y}\overline{\mathtt{x}}\mathtt{yz}\overline{\mathtt{xz}}$, $\overline{\mathtt{xyxy}}\mathtt{z}\overline{\mathtt{xz}}$, $\overline{\mathtt{xyxyzxz}}$; thus, the marking number of σ is 3. In fact, all marking sequences for α have a marking number of 3, except $(\mathtt{y}, \mathtt{x}, \mathtt{z})$, for which it is 2: $\mathtt{x}\overline{\mathtt{y}}\mathtt{x}\overline{\mathtt{y}}\mathtt{zxz}$, $\overline{\mathtt{xyxy}}\mathtt{z}\overline{\mathtt{xz}}$, $\overline{\mathtt{xyxyzxz}}$. Thus, the locality number of α, denoted by $\mathsf{loc}(\alpha)$, is 2.

The locality number has applications in pattern matching with variables [14]. A *pattern* is a word that consists of *terminal symbols* (e.g., $\mathtt{a}, \mathtt{b}, \mathtt{c}$), treated as constants, and *variables* (e.g., $x_1, x_2, x_3, \ldots$). A pattern is mapped to a word by substituting the variables by strings of terminals. For example, $x_1x_1\mathtt{bab}x_2x_2$ can be mapped to $\mathtt{acacbabcc}$ by the substitution $(x_1 \to \mathtt{ac}, x_2 \to \mathtt{c})$. Deciding whether a given pattern matches (i.e., can be mapped to) a given word is one of the most important problems that arise in the study of patterns with variables (note that the concept of patterns with variables arises in several different domains like combinatorics on words (word equations [30], unavoidable patterns [36]), pattern matching [1], language theory [2], learning theory [2, 19, 38, 42, 31, 22], database theory [7], as well as in practice, e.g., extended regular expressions with backreferences [26, 27, 44, 28], used in

programming languages like Perl, Java, Python, etc.). Unfortunately, the *matching problem* is NP-complete [2] in general (it is also NP-complete for strongly restricted variants [23, 21] and also intractable in the parameterised setting [24]). As demonstrated in [43], for the matching problem a paradigm shift as sketched in the first paragraph above yields a very promising algorithmic approach. More precisely, any class of patterns with bounded treewidth (for suitable graph representations) can be matched in polynomial-time. However, computing (and therefore algorithmically exploiting) the treewidth of a pattern is difficult (see the discussion in [21, 43]), which motivates more direct string-parameters that bound the treewidth and are simple to compute (virtually all known structural parameters that lead to tractability [14, 21, 43, 45] are of this kind (the efficiently matchable classes investigated in [15] are one of the rare exceptions)). This also establishes an interesting connection between ad-hoc string parameters and the more general (and much better studied) graph parameter treewidth. The locality number is a simple parameter directly defined on strings, it bounds the treewidth and the corresponding marking sequences can be seen as instructions for a dynamic programming algorithm. However, compared to other "tractability-parameters", it seems to cover best the treewidth of a string, but whether it can be efficiently computed is unclear.

In this paper, we investigate the problem of computing the locality number and, by doing so, we establish an interesting connection to the graph parameters cutwidth and pathwidth with algorithmic implications for approximating cutwidth. In the following, we first discuss related results in more detail and then outline our respective contributions.

Known Results and Open Questions. For Loc, only exact exponential-time algorithms are known and whether it can be solved in polynomial-time, or whether it is at least fixed-parameter tractable is mentioned as open problems in [14]. Approximation algorithms have not yet been considered. Addressing these questions is the main purpose of this paper.

Pathwidth and Cutwidth are NP-complete, but fixed-parameter tractable with respect to parameter $\mathsf{pw}(G)$ or $\mathsf{cw}(G)$, respectively (even with "linear" fpt-time $g(k)\,\mathrm{O}(n)$ [9, 11, 47]). With respect to approximation, their minimisation variants have received a lot of attention, mainly because they yield (like many other graph parameters) general algorithmic approaches for numerous graph problems, i.e., a good linear arrangement or path-decomposition can often be used for a dynamic programming (or even divide and conquer) algorithm. More generally speaking, pathwidth and cutwidth are related to the more fundamental concepts of small balanced vertex or edge separators for graphs (i.e., a small set of vertices (or edges, respectively) that, if removed, divides the graph into two parts of roughly the same size. More precisely, $\mathsf{pw}(G)$ and $\mathsf{cw}(G)$ are upper bounds for the smallest balanced *vertex* separator of G and the smallest balanced *edge* separator of G, respectively (see [20] for further details and explanations of the algorithmic relevance of balanced separators). The best known approximation algorithms for MinPathwidth and MinCutwidth (with approximations ratios of $\mathrm{O}(\sqrt{\log(\mathsf{opt})}\log(n))$ and $\mathrm{O}(\log^2(n))$, respectively) follow from approximations of vertex separators (see [20]) and edge separators (see [35]), respectively.

Our Contributions. There are two natural approaches to represent a word α over alphabet Σ as a graph $G_\alpha = (V_\alpha, E_\alpha)$: (1) $V_\alpha = \{1, 2, \ldots, |\alpha|\}$ and the edges are somehow used to represent the actual symbols, or (2) $V_\alpha = \Sigma$ and the edges are somehow used to represent the positions of α. We present a reduction of type (2) such that $|E_\alpha| = \mathrm{O}(|\alpha|)$ and $\mathsf{cw}(G_\alpha) = 2\,\mathsf{loc}(\alpha)$, and a reduction of type (1) such that $|E_\alpha| = \mathrm{O}(|\alpha|^2)$ and $\mathsf{loc}(\alpha) \leq \mathsf{pw}(G_\alpha) \leq 2\,\mathsf{loc}(\alpha)$. Since these reductions are parameterised reductions and also allow transferring approximation

results, we conclude that Loc is fixed-parameter tractable if parameterised by $|\Sigma|$ or by the locality number (answering the respective open problem from [14]), and also that there is a polynomial-time $O(\sqrt{\log(\mathsf{opt})}\log(n))$-approximation algorithm for MinLoc.

In addition, we also show a way to represent an arbitrary multi-graph $G = (V, E)$ by a word α_G over alphabet V, of length $|E|$ and with $\mathsf{cw}(G) = \mathsf{loc}(\alpha)$. This describes a Turing-reduction from Cutwidth to Loc which also allows to transfer approximation results between the minimisation variants. As a result, we can conclude that Loc is NP-complete (which solves the other open problem from [14]). Finally, by plugging together the reductions from MinCutwidth to MinLoc and from MinLoc to MinPathwidth, we obtain a reduction which transfers approximation results from MinPathwidth to MinCutwidth, which yields an $O(\sqrt{\log(\mathsf{opt})}\log(n))$-approximation algorithm for MinCutwidth. This improves, to our knowledge for the first time since 1999, the best approximation for Cutwidth from [35].

To our knowledge, this connection between cutwidth and pathwidth has not yet been reported in the literature so far. This is rather surprising since Cutwidth and Pathwidth have been jointly investigated in the context of exact and approximation algorithms, especially in terms of balanced vertex and edge separators. More precisely, the approximation of pathwidth and cutwidth follows from the approximation of vertex and edge separators, respectively, and the approximation of vertex separators usually relies on edge separators: the edge separator approximation from [35] can be used as a black-box for vertex separator approximation, and the best vertex separator algorithm from [20] uses a technique for computing edge separators from [4] as component. Our improvement, on the other hand, is achieved by going in the opposite direction: we use pathwidth approximation (following from [20]) in order to improve the currently best cutwidth approximation (from [35]). This might be why the reduction from cutwidth to pathwidth has been overlooked in the literature. Another reason might be that this relation is less obvious on the graph level and becomes more apparent if linked via the string parameter of locality, as in our considerations. Nevertheless, since pathwidth and cutwidth are such crucial parameters for graph algorithms, we also translate our locality based reduction into one from graphs to graphs directly.

2 Preliminaries

Basic Definitions. The set of strings (or words) over an alphabet X is denoted by X^*, by $|\alpha|$ we denote the length of a word α, $\mathsf{alph}(\alpha)$ is the smallest alphabet X with $\alpha \in X^*$. A string β is called a *factor* of α if $\alpha = \alpha'\beta\alpha''$; if $\alpha' = \varepsilon$ or $\alpha'' = \varepsilon$, where ε is the empty string, β is a *prefix* or a *suffix*, respectively. For a position j, $1 \leq j \leq |\alpha|$, we refer to the symbol at position j of α by the expression $\alpha[j]$, and $\alpha[j..j'] = \alpha[j]\alpha[j+1]\dots\alpha[j']$, $1 \leq j \leq j' \leq |\alpha|$. For a word α and $x \in \mathsf{alph}(\alpha)$, let $\mathsf{ps}_x(\alpha) = \{i \mid 1 \leq i \leq |\alpha|, \alpha[i] = x\}$ be the set of all positions where x occurs in α. For a word α, let $\alpha^0 = \varepsilon$ and $\alpha^{i+1} = \alpha\alpha^i$ for $i \geq 0$.

Let α be a word and let $X = \mathsf{alph}(\alpha) = \{x_1, x_2, \dots, x_n\}$. A *marking sequence* is an enumeration, or ordering on the letters, and hence may be represented either as an ordered list of the letters or, equivalently, as a bijection $\sigma : \{1, 2, \dots, |X|\} \to X$. Given a word α and a marking sequence σ, the *marking number* $\pi_\sigma(\alpha)$ (of σ with respect to α) is the maximum number of marked blocks obtained while marking α according to σ. We say that α is k-local if and only if, for some marking sequence σ, we have $\pi_\sigma(\alpha) \leq k$, and the smallest k such that α is k-local is the *locality number* of α, denoted by $\mathsf{loc}(\alpha)$. A marking sequence σ with $\pi_\sigma(\alpha) = \mathsf{loc}(\alpha)$ is *optimal* (for α). For a marking sequence $\sigma = (x_{\sigma(1)}, x_{\sigma(2)}, \dots, x_{\sigma(m)})$ and a word α, by *stage i of σ* we denote the word α with exactly positions $\bigcup_{j=1}^{i} \mathsf{ps}_{x_{\sigma(j)}}(\alpha)$ marked.

For a word α, the *condensed form of α*, denoted by $\mathsf{cond}(\alpha)$, is obtained by replacing every maximal factor x^k with $x \in \mathsf{alph}(\alpha)$ by x. For example, $\mathsf{cond}(x_1x_1x_2x_2x_2x_1x_2x_2) = x_1x_2x_1x_2$. A word α is *condensed* if $\alpha = \mathsf{cond}(\alpha)$.

▶ Remark 1. For a word α, we have $\mathsf{loc}(\mathsf{cond}(\alpha)) = \mathsf{loc}(\alpha)$ [14]. Hence, by computing $\mathsf{cond}(\alpha)$ in time $O(|\alpha|)$, algorithms for computing the locality number (and the respective marking sequences) for *condensed* words extend to algorithms for general words.

Examples and Word Combinatorial Considerations. The structure of 1-local and 2-local words is characterised in [14]. The simplest 1-local words are repetitions x^k for some $k \geq 0$. Furthermore, if α is 1-local, then $y^\ell \alpha y^r$ is 1-local, where $y \notin \mathsf{alph}(\alpha), \ell, r \geq 0$. Marking sequences for 1-local words can be obtained by going from the "inner-most" letters to the "outer-most" ones. The English words *radar*, *refer*, *blender*, and *rotator* are all 1-local.

Generally, in order to have a high locality number, a word needs to contain many alternating occurrences of (at least) two letters. For instance, $(x_1x_2)^n$ is n-local. In general, one can show that if $\mathsf{loc}(w) = k$, then $\mathsf{loc}(w^i) \in \{ik - i + 1, ik\}$.

The well-known *Zimin words* [36] also have high locality numbers compared to their lengths. These words are important in the domain of avoidability, as it was shown that a terminal-free pattern is unavoidable (i.e., it occurs in every infinite word over a large enough finite alphabet) if and only if it occurs in a Zimin word. The Zimin words Z_i, for $i \in \mathbb{N}$, are inductively defined by $Z_1 = x_1$ and $Z_{i+1} = Z_i x_{i+1} Z_i$. Clearly, $|Z_i| = 2^i - 1$ for all $i \in \mathbb{N}$. Regarding the locality of Z_i, note that marking x_2 leads to 2^{i-2} marked blocks; further, marking x_1 first and then the remaining symbols in an arbitrary order only extends or joins marked blocks. Thus, we obtain a sequence with marking number 2^{i-2}. In fact, it can be shown that $\mathsf{loc}(Z_i) = \frac{|Z_i|+1}{4} = 2^{i-2}$ for $i \in \mathbb{N}_{\geq 2}$. Notice that both Zimin words and 1-local words have an obvious palindromic structure. However, in the Zimin words, the letters occur multiple times, but not in large blocks, while in 1-local words there are at most 2 blocks of each letter. One can show that if w is a palindrome, with $w = uau^R$ or $w = uu^R$, and $\mathsf{loc}(u) = k$, then $\mathsf{loc}(w) \in \{2k - 1, 2k, 2k + 1\}$ (u^R denotes the reversal of u).

The number of occurrences of a letter alone is not always a good indicator of the locality of a word. The German word *Einzelelement* (a basic component of a construction) has 5 occurrences of e, but is only 3-local, as witnessed by marking sequence (*l,m,e,i,n,z,t*). Nevertheless, a repetitive structure often leads to high locality. The Finnish word *tutustuttu* (perfect passive of *tutustua* – to meet) is nearly a repetition and 4-local, while *pneumonoultramicroscopicsilicovolcanoconiosis* is an (English) 8-local word, and *lentokonesuihkuturbiinimoottoriapumekaanikkoaliupseerioppilas* is a 10-local (Finnish) word.

Complexity and Approximation. We briefly summarise the fundamentals of parameterised complexity [25, 18] and approximation [5].

A *parameterised problem* is a decision problem with instances (x, k), where x is the actual input and $k \in \mathbb{N}$ is the *parameter*. A parameterised problem P is *fixed-parameter tractable* if there is an *fpt-algorithm* for it, i.e., one that solves P on input (x, k) in time $f(k) \cdot p(|x|)$ for a recursive function f and a polynomial p. We use the $O^*(\cdot)$ notation which hides multiplicative factors polynomial in $|x|$.

A minimisation problem P is a triple (I, S, m), where I is the *set of instances*, S is a function that maps instances $x \in I$ to the *set of feasible solutions* for x, and m is the *objective function* that maps pairs (x, y) with $x \in I$ and $y \in S(x)$ to a positive rational number. For every $x \in I$, we denote $m^*(x) = \min\{m(x, y) \colon y \in S(x)\}$. For $x \in I$ and $y \in S(x)$, the value $R(x, y) = \frac{m(x,y)}{m^*(x)}$ is the *performance ratio* of y with respect to x. An algorithm $\mathcal{A}$ is an *approximation algorithm* for P with ratio $r : \mathbb{N} \to \mathbb{Q}$ (or an r-approximation algorithm, for short) if, for every $x \in I$, $\mathcal{A}(x) = y \in S(x)$, and $R(x, y) \leq r(|x|)$. We also let r be of the form $\mathbb{Q} \times \mathbb{N} \to \mathbb{Q}$ when the ratio r depends on $m^*(x)$ and $|x|$; in this case, we write $r(\mathsf{opt}, |x|)$. We further assume that the function r is monotonically non-decreasing. Unless stated otherwise, all approximation algorithms run in polynomial time with respect to $|x|$.

Pathwidth, Cutwidth and Problem Definitions. Let $G = (V, E)$ be a (multi)graph with the vertices $V = \{v_1, \ldots, v_n\}$. A *cut* of G is a partition (V_1, V_2) of V into two disjoint subsets V_1, V_2, $V_1 \cup V_2 = V$; the (multi)set of edges $\mathcal{C}(V_1, V_2) = \{\{x, y\} \in E \mid x \in V_1, y \in V_2\}$ is called the cut-set or the (multi)set of edges crossing the cut, while V_1 and V_2 are called the sides of the cut. The *size* of this cut is the number of crossing edges, i.e., $|\mathcal{C}(V_1, V_2)|$. A *linear arrangement* of the (multi)graph G is a sequence $(v_{j_1}, v_{j_2}, \ldots, v_{j_n})$, where $(j_1, j_2, \ldots, j_n)$ is a permutation of $(1, 2, \ldots, n)$. For a linear arrangement $L = (v_{j_1}, v_{j_2}, \ldots, v_{j_n})$, let $L(i) = \{v_{j_1}, v_{j_2}, \ldots, v_{j_i}\}$. For every i, $1 \leq i < n$, we consider the cut $(L(i), V \setminus L(i))$ of G, and denote the cut-set $\mathcal{C}_L(i) = \mathcal{C}(L(i), V \setminus L(i))$ (for technical reasons, we also set $\mathcal{C}_L(0) = \mathcal{C}_L(n) = \emptyset$). We define the *cutwidth* of L by $\mathsf{cw}(L) = \max\{|\mathcal{C}_L(i)| \mid 0 \leq i \leq n\}$. Finally, the cutwidth of G is the minimum over all cutwidths of linear arrangements of G, i.e., $\mathsf{cw}(G) = \min\{\mathsf{cw}(L) \mid L \text{ is a linear arrangement for } G\}$.

A path decomposition (see [11]) of a connected graph $G = (V, E)$ is a tree decomposition whose underlying tree is a path, i.e., a sequence $Q = (B_0, B_1, \ldots, B_m)$ (of *bags*) with $B_i \subseteq V$, $0 \leq i \leq m$, satisfying the following two properties:

- *Cover property*: for every $\{u, v\} \in E$, there is an index i, $0 \leq i \leq m$, with $\{u, v\} \subseteq B_i$.
- *Connectivity property*: for every $v \in V$, there exist indices i_v and j_v, $0 \leq i_v \leq j_v \leq m$, such that $\{j \mid v \in B_j\} = \{i \mid i_v \leq i \leq j_v\}$. In other words, the bags that contain v occur on consecutive positions in $(B_0, \ldots, B_m)$.

The *width* of a path decomposition Q is $\mathsf{w}(Q) = \max\{|B_i| \mid 0 \leq i \leq m\} - 1$, and the *pathwidth* of a graph G is $\mathsf{pw}(G) = \min\{\mathsf{w}(Q) \mid Q \text{ is a path decomposition of } G\}$. A path decomposition is *nice* if $B_0 = B_m = \emptyset$ and, for every i, $1 \leq i \leq m$, either $B_i = B_{i-1} \cup \{v\}$ or $B_i = B_{i-1} \setminus \{v\}$, for some $v \in V$.

It is convenient to treat a path decomposition Q as a scheme marking the vertices of the graph based on the order in which the bags occur in the bag sequence. More precisely, all vertices are initially marked as `open`. Then we process the bags one by one, as they occur in Q. When we process the first bag that contains a vertex v, then v becomes `active`. When we process the last bag that contains v, it becomes `closed`. The connectivity property enforces that vertices that are `closed` cannot be marked as `active` again, while the cover property enforces that adjacent vertices must be both `active` at some point. The width is the maximum number of vertices which are marked `active` at the same time minus one. If the path decomposition is nice, then whenever a bag is processed as described above, we change the marking of exactly one vertex.

We next formally define the computational problems of computing the parameters defined above. By Loc, Cutwidth and Pathwidth, we denote the problems to check for a given word α or graph G and integer $k \in \mathbb{N}$, whether $\mathsf{loc}(\alpha) \leq k$, $\mathsf{cw}(G) \leq k$, and $\mathsf{pw}(G) \leq k$, respectively. Note that since we can assume that $k \leq |\alpha|$ and $k \leq |G|$, whether k is given in binary or unary has no impact on the complexity. With the prefix Min, we refer to the minimisation variants. More precisely, MinLoc $= (I, S, m)$, where I is the set of words, $S(\alpha)$ is the set of all marking sequences for α and $m(\alpha, \sigma) = \pi_\sigma(\alpha)$ (note that $m^*(\alpha) = \mathsf{loc}(\alpha)$); MinCutwidth $= (I, S, m)$, where I are all multigraphs, $S(G)$ is the set of linear arrangements of G, and $m(G, L) = \mathsf{cw}(L)$ (note that $m^*(G) = \mathsf{cw}(G)$); finally, MinPathwidth $= (I, S, m)$, where I are all graphs, $S(G)$ is the set of path decompositions of G, and $m(G, Q) = \mathsf{w}(Q)$ (note that $m^*(G) = \mathsf{pw}(G)$).

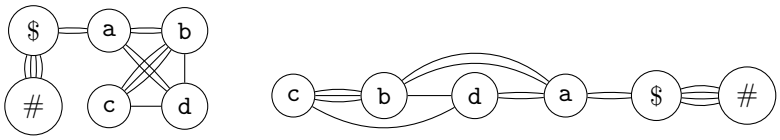


Figure 1 The graph $H_{\alpha,k}$ for $\alpha = \mathtt{abcbcdbada}$ and $k = 2$; an optimal linear arrangement of $H_{\alpha,k}$ with cutwidth 4 induces the optimal marking sequence $(\mathtt{c}, \mathtt{b}, \mathtt{d}, \mathtt{a})$ for α with marking number 2.

3 Locality and Cutwidth

In this section, we introduce polynomial-time reductions from Loc to Cutwidth and vice versa. The established close relationship between these two problems lets us derive several complexity-theoretic and algorithmic results for Loc. We also discuss approximation-preserving properties of our reductions.

First, we show a reduction from Loc to Cutwidth. For a word α and an integer $k \in \mathbb{N}$, we build a multigraph $H_{\alpha,k} = (V, E)$ whose set of nodes $V = \mathsf{alph}(\alpha) \cup \{\$, \#\}$ consists of symbols occurring in α and two additional characters $\$, \# \notin \mathsf{alph}(\alpha)$. The multiset of edges E contains an edge between nodes $x, y \in \mathsf{alph}(\alpha)$ for each occurrence of the factors xy and yx in α, as well as $2k$ edges between \$ and #, one edge between \$ and the first letter of α, and one edge between \$ and the last letter of α. An example is given in Figure 1.

▶ **Lemma 2.** *The graph $H_{\alpha,k}$ satisfies $\mathsf{cw}(H_{\alpha,k}) = 2k$ if and only if $\mathsf{loc}(\alpha) \leq k$.*

Proof. Suppose firstly that α is k-local, and let $\sigma = (x_1, x_2, \ldots, x_n)$ be an optimal marking sequence of α. Consider the linear arrangement $L = (x_1, x_2, \ldots, x_n, \$, \#)$. Clearly, $|\mathcal{C}(\{x_1, x_2, \ldots, x_n, \$\}, \{\#\})| = 2k$ and $|\mathcal{C}(\{x_1, x_2, \ldots, x_n\}, \{\$, \#\})| = 2$. Now consider a cut $(K_1, K_2) = (\{x_1, x_2, \ldots, x_i\}, \{x_{i+1}, \ldots, x_n, \$, \#\})$ for $1 \leq i < n$. Every edge $e \in \mathcal{C}(K_1, K_2)$ is of the form $\{x_j, x_h\}$ with $j \leq i < h$, or of the form $\{\alpha[1], \$\}$ or $\{\$, \alpha[|\alpha|]\}$. Consequently, every edge $e \in \mathcal{C}(K_1, K_2)$ corresponds to a unique factor $x_j x_h$ or $x_h x_j$ of α with $j \leq i < h$ and, after exactly the symbols $x_1, x_2, \ldots, x_i$ are marked, x_j is marked and x_h is not, or to a unique factor $\alpha[1]$ or $\alpha[|\alpha|]$ and, after exactly the symbols $x_1, x_2, \ldots, x_i$ are marked, $\alpha[1]$ or $\alpha[|\alpha|]$ is marked. Since there can be at most k marked blocks in α after marking the symbols $x_1, \ldots, x_i$, there are at most $2k$ such factors, which means that $|\mathcal{C}(K_1, K_2)| \leq 2k$. Thus $\mathsf{cw}(H_{\alpha,k}) \leq 2k$. Note that any linear arrangement must at some point separate the nodes \$ and #, meaning $\mathsf{cw}(H_{\alpha,k}) \geq 2k$, so we get that $\mathsf{cw}(H_{\alpha,k}) = 2k$.

Now suppose that the cutwidth of $H_{\alpha,k}$ is $2k$ and let L be an optimal linear arrangement witnessing this fact. Firstly, we note that L must either start with # followed by \$ (i.e., have the form $(\#, \$, \ldots)$) or end with # preceded by \$ (i.e., have the form $(\ldots, \$, \#)$. Otherwise, since $H_{\alpha,k}$ is connected, every cut separating \$ and # would be of size strictly greater than $2k$. Because a linear ordering and its mirror image have the same cutwidth, we may assume that the optimal linear arrangement has the form $L = (x_{\tau(1)}, x_{\tau(2)}, \ldots, x_{\tau(n)}, \$, \#)$ for some permutation τ of $\{1, \ldots, n\}$. Let σ be the marking sequence $(x_{\tau(1)}, x_{\tau(2)}, \ldots, x_{\tau(n)})$ of α induced by τ. Suppose, for contradiction, that for some i, with $1 \leq i < n$, after marking $x_{\tau(1)}, \ldots, x_{\tau(i)}$, we have $k' > k$ marked blocks. Furthermore, let $K_1 = \{x_{\tau(1)}, \ldots, x_{\tau(i)}\}$ and $K_2 = \{x_{\tau(i+1)}, \ldots, x_{\tau(n)}, \$, \#\}$. For every marked block $\alpha[s..t]$ that is not a prefix or a suffix of α, we have $\alpha[s], \alpha[t] \in K_1$ and $\alpha[s-1], \alpha[t+1] \in K_2$ and therefore $\{\alpha[s-1], \alpha[s]\}, \{\alpha[t], \alpha[t+1]\} \in \mathcal{C}(K_1, K_2)$. Moreover, for a marked prefix $\alpha[1..s]$, we have $\alpha[1], \alpha[s] \in K_1$ and $\$, \alpha[s+1] \in K_2$ and therefore $\{\alpha[1], \$\}, \{\alpha[s], \alpha[s+1]\} \in \mathcal{C}(K_1, K_2)$. Analogously, the existence of a marked suffix $\alpha[t..|\alpha|]$

leads to $\{\alpha[|\alpha|], \$\}, \{\alpha[t-1], \alpha[t]\} \in \mathcal{C}(K_1, K_2)$. Consequently, for each marked block, we have two unique edges in $\mathcal{C}(K_1, K_2)$, which implies $|\mathcal{C}(K_1, K_2)| \geq 2k' > 2k$. This contradicts the assumption that L is a witness that $H_{\alpha,k}$ has cutwidth $2k$. Thus, α must be k-local. ◀

In the following, we briefly discuss the complexity of this reduction. Suppose we are given a word α and an integer $k \leq |\alpha|$. It is usual in string algorithmics to assume that α is over an integer alphabet, i.e., $\mathsf{alph}(\alpha) \subseteq \{1, \ldots, |\alpha|\}$. In this framework, the multigraph $H_{\alpha,k}$ can be constructed in $\mathrm{O}(|\alpha|)$ time (e.g., represented as a list of vertices and a list of edges).

▶ **Lemma 3.** *If there is an $r(\mathsf{opt}, h)$-approximation algorithm for* MinCutwidth *running in $\mathrm{O}(f(h))$ time for an input multigraph with h edges, then there is an $(r(2\,\mathsf{opt}, |\alpha|) + \frac{1}{\mathsf{opt}})$-approximation algorithm for* MinLoc *running in $\mathrm{O}(f(|\alpha|) + |\alpha|)$ time on an input word α.*

Proof. As already indicated in the proof of Lemma 2, for $k = \mathsf{loc}(\alpha)$, every linear arrangement for $H_{\alpha,k}$ naturally translates to a marking sequence for α. However, in an approximate linear arrangement, the vertices # and \$ do not have to be at the first (or last) position. Still, the marking sequence corresponding to the linear arrangement L can have not more than $\frac{\mathsf{cw}(L)}{2}+1$ blocks, since only suffix and prefix can be marked blocks which correspond to only one instead of two edges in a cut in $H_{\alpha,k}$. This observation remains valid if we do not include the extra vertices # and \$ in $H_{\alpha,k}$ in the reduction. Let H_α be the graph obtained from $H_{\alpha,k}$ (for some k) by removing the extra vertices # and \$ (observe that this also removes the dependence on k). Removing vertices only decreases the cutwidth, so Lemma 2 implies that $\mathsf{cw}(H_\alpha) \leq 2m^*(\alpha)$. Let α be an instance of MinLoc and $\mathcal{A}$ an $r(\mathsf{opt}, h)$-approximation for MinCutwidth on multigraphs. The approximation algorithm $\mathcal{A}$ run on H_α returns a linear arrangement $L = \mathcal{A}(H_\alpha)$ with $\mathsf{cw}(L) \leq r(\mathsf{opt}, h)\,\mathsf{cw}(H_\alpha)$. Let σ be the marking sequence corresponding to L, then $R(\alpha, \sigma) = \frac{\pi_\sigma(\alpha)}{m^*(\alpha)} \leq \frac{2}{\mathsf{cw}(H_\alpha)}(\frac{\mathsf{cw}(L)}{2}+1) = \frac{\mathsf{cw}(L)}{\mathsf{cw}(H_\alpha)} + \frac{1}{m^*(\alpha)} = R(H_\alpha, L) + \frac{1}{m^*(\alpha)}$. The performance ratio $R(H_\alpha, L)$ is at most $r(\mathsf{opt}, h)$, where $h = |\alpha|$ is the number of edges in H_α. For the optimum value $k = m^*(\alpha)$, the cutwidth of $H_{\alpha,k}$ is at least $2k-2$ and σ has performance ratio at most $r(2\,\mathsf{opt}, |\alpha|)$ (with respect to the optimum value k for MinLoc). The approximation procedure builds the graph H_α in $\mathrm{O}(|\Sigma|)$, runs $\mathcal{A}$ on H_α in $\mathrm{O}(f(|\alpha|))$ and translates the linear arrangement into a marking sequence σ in $\mathrm{O}(|\Sigma|)$. This gives an $(r(2\,\mathsf{opt}, |\alpha|) + \frac{1}{\mathsf{opt}})$-approximation for MinLoc running time in $\mathrm{O}(f(|\alpha|) + |\alpha|)$ time. ◀

For a reduction from Cutwidth to Loc, let $H = (V, E)$ be a connected multigraph, where V is the set of nodes and E the multiset of edges (for technical reasons, we assume $|V| \geq 2$). Let $H' = (V, E')$ be the multigraph obtained by duplicating every edge in H. As such, each node in H' has even degree, so there exists an Eulerian cycle C (i.e., a cycle visiting each edge exactly once) in H', and, moreover, $\mathsf{cw}(H') = 2\,\mathsf{cw}(H)$. For each edge $e \in E'$, let α_e be the word over V that corresponds to an arbitrary traversal of the Eulerian path P obtained from C by deleting e; see Figure 2 for an example.

▶ **Lemma 4.** *For any edge e in E', the word α_e satisfies $\mathsf{cw}(H) \leq \mathsf{loc}(\alpha_e) \leq \mathsf{cw}(H)+1$. Moreover, there is a vertex $v \in V$ such that $\mathsf{loc}(\alpha_e) = \mathsf{cw}(H)$ for every edge e incident to v.*

Consequences. In the following, we overview a series of complexity-theoretic and algorithmic consequences of the reductions provided above. We first discuss negative results and note that we can close one of the main problems left open in [14].

▶ **Theorem 5.** *The* Loc *problem is NP-complete.*

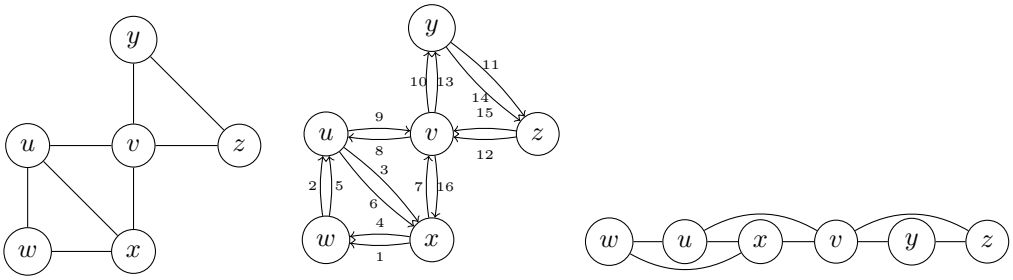


Figure 2 A graph H and its multigraph H' obtained by doubling the edges; the edge labels describe a Eulerian cycle that starts and ends in x. Deleting the edge (v, x) in this cycle yields the word $\alpha_{(v,x)} = xwuxwuxvuvyzvyzv$, which has an optimal marking sequence (w, u, x, v, y, z) with marking number 3, and, thus, induces an optimal linear arrangement of H with cutwidth 3.

Theorem 5 follows from the Turing reduction from CUTWIDTH to LOC, but it can also be proved using a polynomial-time one-to-many reduction from the well known NP-complete problem CLIQUE. This alternative approach is more technically involved but has the merit of emphasising how the combinatorial properties of the locality number can be used to construct computationally hard instances of LOC. Moreover, by the word-combinatorial observations about locality made in Section 2, it is clear that LOC is NP-complete also for words with special structure, e.g., palindromes and repetitions.

With respect to approximation, it is known that, assuming the Small Set Expansion Conjecture (denoted SSE; see [40]), there exists no constant-ratio approximation for MINCUTWIDTH (see [48]). Consequently, approximating MINLOC within any constant factor is also SSE-hard. In particular, we point out that stronger inapproximability results for MINCUTWIDTH are not known. Positive approximation results for MINLOC will be discussed in Section 4.

On certain graph classes, the SSE conjecture is equivalent to the Unique Games Conjecture [32] (see [40, 41]), which, at its turn, was used to show that many approximation algorithms are tight [33] and is considered a major conjecture in inapproximability. However, some works seem to provide evidence that could lead to a refutation of SSE; see [3, 6, 29]. In this context, we show in Section 4 a series of unconditional results which state that multiple natural greedy strategies do not provide low-ratio approximations of MINLOC.

As formally stated next, Lemma 2 extends algorithmic results for computing cutwidth to determining the locality number (we formulate this result so that it also covers fpt-algorithms with respect to the standard parameters $\mathsf{cw}(G)$ and $\mathsf{loc}(\alpha)$). Note that the maximum degree in a multigraph G is bounded from above by $2\,\mathsf{cw}(G)$, so the number of nodes n and the number of edges h satisfy $h \leq n \cdot \mathsf{cw}(G)$. Hence, we state the complexity in terms of n and $\mathsf{cw}(G)$ rather than with respect to h, which is the actual input size.

▶ **Lemma 6.** *If* MINCUTWIDTH *(resp.* CUTWIDTH*) can be solved in* $\mathrm{O}(f(\mathsf{cw}(G), n))$ *time for a multigraph* G *with* n *vertices, then the* MINLOC *(resp.,* LOC*) problem can be solved in* $\mathrm{O}(f(2\,\mathsf{loc}(\alpha), |\Sigma| + 2) + |\alpha|)$ *time for a word* α *over an alphabet* Σ*.*

In particular, we can draw the following corollaries using Lemma 6 and known results from the literature. Due to the algorithms of [12], which also work for multigraphs[1], MINLOC can be solved in $\mathrm{O}^*(2^{|\Sigma|})$ time and space, or in $\mathrm{O}^*(4^{|\Sigma|})$ time and polynomial space. In

[1] These algorithms actually support weighted graphs without any major modification and in the same complexity. In this setting, parallel edges connecting two vertices are replaced by a single "super-edge" whose weight is the number of parallel edges.

particular, this also implies that Loc is fixed-parameter tractable with respect to the alphabet size. Moreover, the fpt-algorithm from [47] directly implies that MinLoc is fixed-parameter tractable for parameter $\mathsf{loc}(\alpha)$ with linear fpt-running-time $g(\mathsf{loc}(\alpha))\,\mathrm{O}(n)$. Since Cutwidth is NP-complete already for graphs with maximum degree 3 (see [37]), we also derive a stronger statement compared to Theorem 5: Loc is NP-complete even if every symbol has at most 3 occurrences; if every symbol has at most 2 occurrences, the complexity of Loc is open, while the case where every symbol has only one occurrence is trivial. If, on the other hand, the symbols have many occurrences in comparison to $|\alpha|$, i.e., $|\Sigma| = \mathrm{O}(\log(|\alpha|))$, then Loc can be solved in polynomial time, e.g., using the $\mathrm{O}^*(2^{|\Sigma|})$-time algorithm mentioned above.

4 Locality and Pathwidth

In this section, we consider the approximability of the minimisation problem MinLoc. Since a marking sequence is just a linear arrangement of the symbols of the input word, this problem seems to be well tailored to greedy algorithms: until all symbols are marked, we choose an unmarked symbol according to some greedy strategy and mark it. There are two aspects that motivate the investigation of such approaches. Firstly, ruling out simple strategies is a natural initial step in the search for approximation algorithms for a new problem. Secondly, due to the results of Section 3, the obvious greedy approaches for computing the locality number may also provide a new angle to approximating the cutwidth of a graph, i.e., some greedy strategies may only become apparent in the locality number point of view and hard to see in the graph formulation of the problem. Given the fact that, as formally stated later as Theorem 10, approximating the cutwidth via approximation of the locality number does, in fact, improve the best currently known cutwidth approximation ratio, this seems to be a rather important aspect.

Unfortunately, we can formally show that many natural candidates for greedy strategies fail to yield promising approximation algorithms (and are therefore also not helpful for cutwidth approximation). We just briefly mention these negative results. The four considered basic strategies are the following: (1) prefer symbols with few occurrences, (2) symbols with many occurrences, (3) symbols leading to fewer blocks after marking, (4) symbols with earlier leftmost occurrence. All these strategies fail in a sense that there are arbitrarily long (condensed) words α with constant locality numbers for which these strategies yield marking sequences with marking numbers $\Omega(|\alpha|)$.

A more promising approach is to choose among symbols that extend at least one already marked block (except when marking the first symbol). We denote this strategy by BlockExt and marking sequences that can be obtained by it are called BlockExt-marking sequences. Intuitively, marking a symbol that has only isolated occurrences, and therefore will increase the current number of marked blocks by the number of its occurrences, seems a bad choice. This raises a general question of whether every word has a BlockExt-marking sequence that is also optimal for this word. We answer this question negatively: all BlockExt-marking sequences for words like $x_1yx_2yx_3y\dots x_{2k}y$ achieve a marking number of at least $2k-1$, while first marking $x_2, x_3, \dots, x_{k+1}$ in this order (which all have only isolated occurrences), then y, and then the rest of the symbols in some order, yields at most k marked blocks. However, this only shows a lower bound of roughly 2 for the approximation ratio of algorithms based on BlockExt, so BlockExt might still be a promising candidate. However, in order to devise a BlockExt-based approximation algorithm, we still face the problem of deciding which of the extending symbols should be chosen; trying out all of them is obviously too costly. Unfortunately, if we handle this decision by one of the basic strategies (1)–(4) from above,

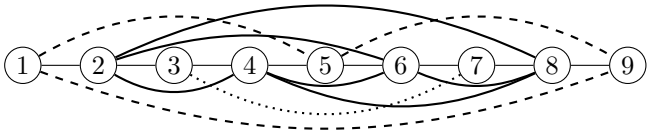


Figure 3 The graph G_α for $\alpha = \texttt{cabacabac}$; the three cliques are drawn with different edge-types.

e.g., choosing among all extending symbols one that leads to fewer new blocks, we again end up with poor approximation ratios. More precisely, we can again find arbitrarily long words α with constant locality numbers for which these algorithms yield marking numbers $\Omega(|\alpha|)$. Moreover, this is also true if we choose among all extending symbols one that has a maximum number of extending occurrences or one that maximises the ratio $\frac{\#\text{extending occ.}}{\#\text{occ.}}$.

While we obviously have not investigated *all* reasonable greedy strategies, we consider our negative results as sufficient evidence that a worthwhile approximation algorithm for computing the locality number most likely does not follow from such simple greedy strategies.

In the following, we adopt a more sophisticated approach of approximating the locality number: we devise a reduction to the problem of computing the pathwidth of a graph. To this end, we first have to describe how a (condensed) word can be represented as a graph: For a condensed word α, the graph $G_\alpha = (V_\alpha, E_\alpha)$ is defined by $V_\alpha = \{1, 2, \ldots, |\alpha|\}$ and $E_\alpha = \{\{i, i+1\} \mid 1 \leq i \leq |\alpha|-1\} \cup \{\{i, j\} \mid \{i, j\} \subseteq \mathsf{ps}_x(\alpha) \text{ for some } x \in \mathsf{alph}(\alpha)\}$. Intuitively, G_α is obtained by interpreting every position of α as a vertex, connecting neighbouring positions by edges, and turning every set $\mathsf{ps}_x(\alpha)$, $x \in \mathsf{alph}(\alpha)$, into a clique (see Figure 3).

We use G_α as a unique graph representation for condensed words and whenever we talk about a path decomposition for α, we actually refer to a path decomposition of G_α and, since G_α has the positions of α as its vertices, the marking scheme behind a path decomposition (and its respective terminology) directly translates to a marking scheme of the positions of α.

▶ **Lemma 7.** *Let α be a condensed word with $|\alpha| \geq 2$. Then $\mathit{loc}(\alpha) \leq \mathit{pw}(G_\alpha) \leq 2\,\mathit{loc}(\alpha)$.*

Proof Sketch. We only sketch how a marking sequence translates into a path decomposition and vice versa. Let $\sigma = (x_1, x_2, \ldots, x_m)$ be a marking sequence for a condensed word α with $\pi_\sigma(\alpha) = k$. We describe a path decomposition Q of G_α as a marking scheme. First, for every i, $1 \leq i \leq m$, let p_i be a step of Q (corresponding to one of the bags of Q) that represents stage i of σ: every position that is a border position of a marked block is `active`, every other marked position is `closed`, and all other positions are `open`. The path decomposition produces these steps in the order $p_1, p_2, \ldots, p_m$ and such a step p_i is reached from the predecessor step p_{i-1} by a sequence of intermediate steps as follows. Step p_1 is obtained from the initial one by setting all positions in $\mathsf{ps}_{x_1}(\alpha)$ to `active`, the final step of Q, where all positions are `closed`, is obtained from step p_m by setting the only `active` positions 1 and $|\alpha|$ to `closed`. In order to produce step p_{i+1} from step p_i, we do the following. For every $j \in \mathsf{ps}_{x_{i+1}}(\alpha)$ that does not create a new marked block of size 1, we set position j to `active` and immediately after that, we set all `active` neighbours of j to `closed` if they do not have `open` neighbours anymore. Next, we set all remaining positions from $\mathsf{ps}_{x_{i+1}}(\alpha)$ to `active` and, finally, we set all positions from $\mathsf{ps}_{x_{i+1}}(\alpha)$ that have no `open` neighbours to `closed`. It can be verified with a moderate effort that we have now obtained step p_{i+1} and also that Q is, in fact, a valid path decomposition of G_α. In order to see that $\mathsf{pw}(Q) \leq 2k$, we first note that the number of `active` positions in each step p_i is clearly bounded by $2k$. In going

from p_i to p_{i+1}, we necessarily reach a step where all positions of $\mathsf{ps}_{x_{i+1}}(\alpha)$ are now `active`, while some of the previous border positions might also still be `active`. It requires a more involved and careful counting argument to see that the total number of `active` positions in these intermediate steps never exceeds $2k+1$.

For the other direction, let $Q = (B_0, B_1, B_2, \ldots, B_{2|\alpha|})$ be an arbitrary nice path decomposition of G_α. For every i, $1 \leq i \leq m$, let p_i be the first step of Q where all positions of $\mathsf{ps}_{x_i}(\alpha)$ are `active`. We order the characters x_i so that $p_1 < p_2 < \ldots < p_m$ and define a marking sequence for α by setting $\sigma = (x_1, x_2, \ldots, x_m)$. It is comparatively easy to show that at every step p_i of Q there is at least one `active` position per marked block of stage i of σ. However, this only shows that $\pi_\sigma(\alpha) - 1 \leq \mathsf{pw}(Q)$ and in order to prove that, in fact, $\pi_\sigma(\alpha) \leq \mathsf{pw}(Q)$ holds, we show that either there is a step of Q with at least $\pi_\sigma(\alpha)+1$ `active` positions or, if this is not the case, then there must be a marking sequence σ' with $\pi_{\sigma'}(\alpha) = \pi_\sigma(\alpha) - 1$. This implies that, for every path decomposition Q of G_α, $\mathsf{loc}(\alpha) \leq \mathsf{pw}(Q)$ and therefore also $\mathsf{loc}(\alpha) \leq \mathsf{pw}(G_\alpha)$. Proving this claim requires a long and technically involved case analysis. ◀

Note that Lemma 7 is not true for condensed words α of size 1, since then $\mathsf{loc}(\alpha) = 1$ and $\mathsf{pw}(G_\alpha) = 0$. The reason why $\mathsf{pw}(G_\alpha)$ can range between $\mathsf{loc}(\alpha)$ and $2\,\mathsf{loc}(\alpha)$ (rather than $\mathsf{pw}(G_\alpha) = 2\,\mathsf{loc}(\alpha)$) is that in a marking sequence, every marked block accounts for one unit of the quantity $\mathsf{loc}(\alpha)$, while in the path decomposition, a marked block is represented either by two `active` vertices or by only one (if the block has size one). There are (condensed) examples that reach the extremes $\mathsf{loc}(\alpha)$ and $2\,\mathsf{loc}(\alpha)$, i.e., the bounds of Lemma 7 are tight.

▶ **Proposition 8.** *Let $\alpha = (x_1 x_2 \ldots x_n x_{n-1} \ldots x_2)^k x_1$ with $n \geq 3$, and let $\beta = (x_1 x_2)^k$. Then we have $\mathsf{loc}(\alpha) = k$ and $\mathsf{pw}(G_\alpha) = 2k$, and $\mathsf{loc}(\beta) = \mathsf{pw}(G_\beta) = k$.*

Note that the construction of a graph G_α from a word α does not technically provide a reduction from the decision problem Loc to Pathwidth (due to the fact that $\mathsf{pw}(G_\alpha)$ lies between $\mathsf{loc}(\alpha)$ and $2\,\mathsf{loc}(\alpha)$) and therefore cannot be used to solve MinLoc exactly. Its main purpose is to carry over approximation results from MinPathwidth to MinLoc, which is formally stated by the next lemma (in this regard, note that exact algorithms for MinLoc are obtained in Section 3 via a reduction to MinCutwidth instead).

▶ **Lemma 9.** *If* MinPathwidth *admits an $\mathrm{O}(f(n))$-time $r(\mathsf{opt}, n)$-approximation algorithm, then* MinLoc *admits an $\mathrm{O}(f(|\alpha|) + |\alpha|^2)$-time $2r(2\,\mathsf{opt}, |\alpha|)$-approximation algorithm.*

Consequently, approximation algorithms for MinPathwidth carry over to MinLoc. To the knowledge of the authors, the currently best approximation algorithm for MinPathwidth is due to [20], with an approximation ratio of $\mathrm{O}(\sqrt{\log(\mathsf{opt})}\log(n))$. This implies the following.

▶ **Theorem 10.** *There is an $\mathrm{O}(\sqrt{\log(\mathsf{opt})}\log(n))$-approximation algorithm for* MinLoc.

Another consequence that is worth mentioning is due to the fact that an optimal path decomposition can be computed faster than $\mathrm{O}^*(2^n)$. More precisely, it is shown in [46] that for computing path decompositions, there is an exact algorithm with running time $\mathrm{O}^*((1.9657)^n)$, and even an additive approximation algorithm with running time $\mathrm{O}^*((1.89)^n)$. Consequently, there is a 2-approximation algorithm for MinLoc with running time $\mathrm{O}^*((1.9657)^n)$ and an asymptotic 2-approximation algorithm with running time $\mathrm{O}^*((1.89)^n)$ for MinLoc.

By combining the reduction from MinCutwidth to MinLoc from Section 3 with the reduction from MinLoc to MinPathwidth defined above, we obtain a reduction from MinCutwidth to MinPathwidth that carries over the pathwidth-approximation from [20] to MinCutwidth as follows (in particular, this improves the state-of-the-art approximation algorithm for MinCutwidth from [35]).

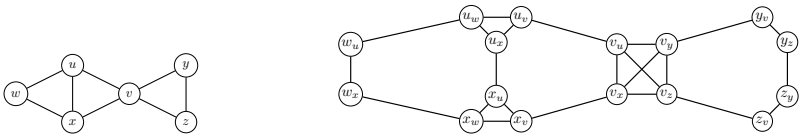


Figure 4 A graph G and the corresponding graph G' obtained by the reduction.

▶ **Theorem 11.** *There is a* $O(\sqrt{\log(\mathsf{opt})}\log(n))$*-approximation for* MinCutwidth.

Note that Theorem 11 only applies to simple graphs; see Section 5 for the case of multigraphs.

Many existing algorithms constructing path decompositions are of theoretical interest only, and this disadvantage carries over to the possible algorithms computing the locality number or cutwidth based on them. However, the reduction of Lemma 7 is also applicable in a purely practical scenario, since any kind of practical algorithm constructing path decompositions can be used in order to compute marking sequences (the additional tasks of building G_α and the translation of a path decomposition for it back to a marking sequence are computationally simple). This observation is particularly interesting since developing practical algorithms constructing tree and path decompositions of small width is a vibrant research area.[2]

5 Pathwidth and Cutwidth

Since pathwidth and cutwidth are classical graph parameters that play an important role for graph algorithms, independent from our application for computing the locality number, we also present a direct reduction from MinCutwidth to MinPathwidth.

For a graph $G = (V, E)$, we construct the graph $G' = (V', E')$ with $V' = \{v_u \mid \{u, v\} \in E\}$ and $E' = \{\{u_v, v_u\} \mid \{u, v\} \in E\} \cup \{\{v_u, v_w\} \mid \{u, v\}, \{w, v\} \in E, u \neq w\}$; see Figure 4.

▶ **Lemma 12.** *Let G be a graph with at least one edge. Then* $\mathsf{cw}(G) \leq \mathsf{pw}(G') \leq 2\,\mathsf{cw}(G)$.

Lemma 12 does not only prove that $\mathsf{cw}(G) \leq \mathsf{pw}(G') \leq 2\,\mathsf{cw}(G)$, but also yields a constructive way to compute a linear arrangement for G of cut at most k from a path decomposition of width k for G'. Further, Lemma 12 remains true if G is a multigraph; observe that the reduction still constructs a simple graph G'. This gives the following result.

▶ **Lemma 13.** *If there is an $r(\mathsf{opt}, |V|)$-approximation algorithm for* MinPathwidth *with running-time $O(f(|V|))$, then there is also an $2r(2\,\mathsf{opt}, h)$-approximation algorithm for* MinCutwidth *on multigraphs with running time $O(f(h) + h^2 + n)$, where n is the number of vertices and h is the number of edges.*

With the $O(\sqrt{\log(\mathsf{opt})}\log(n))$-approximation for MinPathwidth from [20], Lemma 13 gives the following approximation for MinCutwidth on multigraphs.

▶ **Theorem 14.** *There is a (polynomial-time)* $O(\sqrt{\log(\mathsf{opt})}\log(h))$*-approximation algorithm for* MinCutwidth *on multigraphs with h edges.*

In accordance with Theorem 11, Theorem 14 yields an $O(\sqrt{\log(\mathsf{opt})}\log(n))$-approximation algorithm for simple graphs. Analogously, Theorem 11 could be formulated for multigraphs, which would also change the approximation-ratio to $O(\sqrt{\log(\mathsf{opt})}\log(h))$.

[2] See, e.g., the work [13] and the references therein for practical algorithms constructing path decompositions; also note that designing exact and heuristic algorithms for constructing tree decompositions was part of the "PACE 2017 Parameterized Algorithms and Computational Experiments Challenge" [16].

References

1. Amihood Amir and Igor Nor. Generalized function matching. *Journal of Discrete Algorithms*, 5(3):514–523, 2007. doi:10.1016/j.jda.2006.10.001.
2. Dana Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21(1):46–62, 1980. doi:10.1016/0022-0000(80)90041-0.
3. Sanjeev Arora, Boaz Barak, and David Steurer. Subexponential Algorithms for Unique Games and Related Problems. *Journal of the ACM*, 62(5):42:1–42:25, 2015. doi:10.1145/2775105.
4. Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM*, 56(2):5:1–5:37, 2009. doi:10.1145/1502793.1502794.
5. Giorgio Ausiello, Alberto Marchetti-Spaccamela, Pierluigi Crescenzi, Giorgio Gambosi, Marco Protasi, and Viggo Kann. *Complexity and approximation: combinatorial optimization problems and their approximability properties*. Springer, 1999. doi:10.1007/978-3-642-58412-1.
6. Boaz Barak, Prasad Raghavendra, and David Steurer. Rounding Semidefinite Programming Hierarchies via Global Correlation. In Rafail Ostrovsky, editor, *52nd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2011*, pages 472–481. IEEE, 2011. doi:10.1109/focs.2011.95.
7. Pablo Barceló, Leonid Libkin, Anthony W. Lin, and Peter T. Wood. Expressive Languages for Path Queries over Graph-Structured Data. *ACM Transactions on Database Systems*, 37(4):1–46, 2012. doi:10.1145/2389241.2389250.
8. Hans L. Bodlaender. A Tourist Guide through Treewidth. *Acta Cybernetica*, 11(1–2):1–21, 1993. URL: http://www.inf.u-szeged.hu/actacybernetica/edb/vol11n1_2/pdf/Bodlaender_1993_ActaCybernetica.pdf.
9. Hans L. Bodlaender. A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth. *SIAM Journal on Computing*, 25(5):1305–1317, 1996. doi:10.1137/s0097539793251219.
10. Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1–2):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.
11. Hans L. Bodlaender. Fixed-Parameter Tractability of Treewidth and Pathwidth. In Hans L. Bodlaender, Rod Downey, Fedor V. Fomin, and Dániel Marx, editors, *The Multivariate Algorithmic Revolution and Beyond*, volume 7370 of *LNCS*, pages 196–227, 2012. doi:10.1007/978-3-642-30891-8_12.
12. Hans L. Bodlaender, Fedor V. Fomin, Arie M. C. A. Koster, Dieter Kratsch, and Dimitrios M. Thilikos. A Note on Exact Algorithms for Vertex Ordering Problems on Graphs. *Theory of Computing Systems*, 50(3):420–432, 2012. doi:10.1007/s00224-011-9312-0.
13. David Coudert, Dorian Mazauric, and Nicolas Nisse. Experimental Evaluation of a Branch-and-Bound Algorithm for Computing Pathwidth and Directed Pathwidth. *ACM Journal of Experimental Algorithmics*, 21(1):1.3:1–1.3:23, 2016. doi:10.1145/2851494.
14. Joel D. Day, Pamela Fleischmann, Florin Manea, and Dirk Nowotka. Local Patterns. In Satya V. Lokam and R. Ramanujam, editors, *Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017*, volume 93 of *LIPIcs*, pages 24:1–24:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.FSTTCS.2017.24.
15. Joel D. Day, Pamela Fleischmann, Florin Manea, Dirk Nowotka, and Markus L. Schmid. On Matching Generalised Repetitive Patterns. In Mizuho Hoshi and Shinnosuke Seki, editors, *Developments in Language Theory, DLT 2018*, volume 11088 of *LNCS*, pages 269–281. Springer, 2018. doi:10.1007/978-3-319-98654-8_22.
16. Holger Dell, Christian Komusiewicz, Nimrod Talmon, and Mathias Weller. The PACE 2017 Parameterized Algorithms and Computational Experiments Challenge: The Second Iteration. In Daniel Lokshtanov and Naomi Nishimura, editors, *Parameterized and Exact Computation, IPEC 2017*, volume 89 of *LIPIcs*, pages 30:1–30:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.IPEC.2017.30.
17. Josep Díaz, Jordi Petit, and Maria Serna. A Survey of Graph Layout Problems. *ACM Computing Surveys*, 34(3):313–356, 2002. doi:10.1145/568522.568523.

18 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

19 Thomas Erlebach, Peter Rossmanith, Hans Stadtherr, Angelika Steger, and Thomas Zeugmann. Learning one-variable pattern languages very efficiently on average, in parallel, and by asking queries. *Theoretical Computer Science*, 261(1):119–156, 2001. doi:10.1016/s0304-3975(00)00136-5.

20 Uriel Feige, MohammadTaghi HajiAghayi, and James R. Lee. Improved Approximation Algorithms for Minimum Weight Vertex Separators. *SIAM Journal on Computing*, 38(2):629–657, 2008. doi:10.1137/05064299x.

21 Henning Fernau, Florin Manea, Robert Mercas, and Markus L. Schmid. Pattern Matching with Variables: Fast Algorithms and New Hardness Results. In Ernst W. Mayr and Nicolas Ollinger, editors, *Symposium on Theoretical Aspects of Computer Science, STACS 2015*, volume 30 of *LIPIcs*, pages 302–315. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.STACS.2015.302.

22 Henning Fernau, Florin Manea, Robert Mercaş, and Markus L. Schmid. Revisiting Shinohara's Algorithm for Computing Descriptive Patterns. *Theoretical Computer Science*, 733:44–54, 2018. doi:10.1016/j.tcs.2018.04.035.

23 Henning Fernau and Markus L. Schmid. Pattern matching with variables: A multivariate complexity analysis. *Information and Computation*, 242:287–305, 2015. doi:10.1016/j.ic.2015.03.006.

24 Henning Fernau, Markus L. Schmid, and Yngve Villanger. On the Parameterised Complexity of String Morphism Problems. *Theory of Computing Systems*, 59(1):24–51, 2016. doi:10.1007/s00224-015-9635-3.

25 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006. doi:10.1007/3-540-29953-X.

26 Dominik D. Freydenberger. Extended Regular Expressions: Succinctness and Decidability. *Theory of Computing Systems*, 53(2):159–193, 2013. doi:10.1007/s00224-012-9389-0.

27 Dominik D. Freydenberger and Markus L. Schmid. Deterministic regular expressions with back-references. *Journal of Computer and System Sciences*, 2019. doi:10.1016/j.jcss.2019.04.001.

28 Jeffrey E. F. Friedl. *Mastering Regular Expressions*. O'Reilly, Sebastopol, CA, 3rd edition, 2006.

29 Venkatesan Guruswami and Ali Kemal Sinop. Lasserre Hierarchy, Higher Eigenvalues, and Approximation Schemes for Graph Partitioning and Quadratic Integer Programming with PSD Objectives. In Rafail Ostrovsky, editor, *52nd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2011*, pages 482–491. IEEE, 2011. doi:10.1109/FOCS.2011.36.

30 Juhani Karhumäki, Filippo Mignosi, and Wojciech Plandowski. The expressibility of languages and relations by word equations. *Journal of the ACM*, 47(3):483–505, 2000. doi:10.1145/337244.337255.

31 Michael Kearns and Leonard Pitt. A polynomial-time algorithm for learning k-variable pattern languages from examples. In Ronald L. Rivest, David Haussler, and Manfred K. Warmuth, editors, *Computational Learning Theory, COLT 1989*, pages 57–71. Morgan Kaufmann, 1989. doi:10.1016/b978-0-08-094829-4.50007-6.

32 Subhash Khot. On the power of unique 2-prover 1-round games. In John H. Reif, editor, *34th Annual ACM Symposium on Theory of Computing, STOC 2002*, pages 767–775. ACM, 2002. doi:10.1145/509907.510017.

33 Subhash Khot. On the Unique Games Conjecture (Invited Survey). In *Computational Complexity, CCC 2010*, pages 99–121. IEEE, 2010. doi:10.1109/CCC.2010.19.

34 Ton Kloks, editor. *Treewidth, Computations and Approximations*, volume 842 of *LNCS*. Springer, 1994. doi:10.1007/BFb0045375.

35 Tom Leighton and Satish Rao. Multicommodity Max-flow Min-cut Theorems and Their Use in Designing Approximation Algorithms. *Journal of the ACM*, 46(6):787–832, 1999. doi:10.1145/331524.331526.

36 M. Lothaire, editor. *Algebraic Combinatorics on Words*. Cambridge University Press, 2002. doi:10.1017/cbo9781107326019.

37 Fillia Makedon, Christos H. Papadimitriou, and Ivan Hal Sudborough. Topological Bandwidth. *SIAM Journal on Algebraic and Discrete Methods*, 6(3):418–444, 1985. doi:10.1137/0606044.

38 Yen Kaow Ng and Takeshi Shinohara. Developments from enquiries into the learnability of the pattern languages from positive data. *Theoretical Computer Science*, 397(1–3):150–165, 2008. doi:10.1016/j.tcs.2008.02.028.

39 Jordi Petit. Addenda to the Survey of Layout Problems. *Bulletin of the EATCS*, 105:177–201, 2011. URL: http://eatcs.org/beatcs/index.php/beatcs/article/view/98.

40 Prasad Raghavendra and David Steurer. Graph expansion and the unique games conjecture. In Leonard J. Schulman, editor, *42nd ACM Symposium on Theory of Computing, STOC 2010*, pages 755–764. ACM, 2010. doi:10.1145/1806689.1806792.

41 Prasad Raghavendra, David Steurer, and Madhur Tulsiani. Reductions between Expansion Problems. In *Computational Complexity, CCC 2012*, pages 64–73. IEEE, 2012. doi:10.1109/CCC.2012.43.

42 Daniel Reidenbach. Discontinuities in pattern inference. *Theoretical Computer Science*, 397(1–3):166–193, 2008. doi:10.1016/j.tcs.2008.02.029.

43 Daniel Reidenbach and Markus L. Schmid. Patterns with bounded Treewidth. *Information and Computation*, 239:87–99, 2014. doi:10.1016/j.ic.2014.08.010.

44 Markus L. Schmid. Characterising REGEX languages by regular languages equipped with factor-referencing. *Information and Computation*, 249:1–17, 2016. doi:10.1016/j.ic.2016.02.003.

45 Takeshi Shinohara. Polynomial Time Inference of Pattern Languages and Its Application. In *7th IBM Symposium on Mathematical Foundations of Computer Science*, pages 191–209, 1982.

46 Karol Suchan and Yngve Villanger. Computing Pathwidth Faster Than 2^n. In Jianer Chen and Fedor V. Fomin, editors, *Parameterized and Exact Computation, IWPEC 2009*, volume 5917 of *LNCS*, pages 324–335. Springer, 2009. doi:10.1007/978-3-642-11269-0_27.

47 Dimitrios M. Thilikos, Maria J. Serna, and Hans L. Bodlaender. Cutwidth I: A linear time fixed parameter algorithm. *Journal of Algorithms*, 56(1):1–24, 2005. doi:10.1016/j.jalgor.2004.12.001.

48 Yu (Ledell) Wu, Per Austrin, Toniann Pitassi, and David Liu. Inapproximability of Treewidth and Related Problems. *Journal of Artificial Intelligence Research*, 49(1):569–600, 2014. doi:10.1613/jair.4030.

Solutions Sets to Systems of Equations in Hyperbolic Groups Are EDT0L in PSPACE

Laura Ciobanu
Heriot-Watt University, Edinburgh EH14 4AS, Scotland
l.ciobanu@hw.ac.uk

Murray Elder
University of Technology Sydney, Ultimo NSW 2007, Australia
murray.elder@uts.edu.au

Abstract

We show that the full set of solutions to systems of equations and inequations in a hyperbolic group, with or without torsion, as shortlex geodesic words, is an EDT0L language whose specification can be computed in NSPACE$(n^2 \log n)$ for the torsion-free case and NSPACE$(n^4 \log n)$ for the torsion case. Our work combines deep geometric results by Rips, Sela, Dahmani and Guirardel on decidability of existential theories of hyperbolic groups, work of computer scientists including Plandowski, Jeż, Diekert and others on PSPACE algorithms to solve equations in free monoids and groups using compression, and an intricate language-theoretic analysis.

The present work gives an essentially optimal formal language description for all solutions in all hyperbolic groups, and an explicit and surprising low space complexity to compute them.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory; Theory of computation → Grammars and context-free languages; Theory of computation → Complexity classes; Mathematics of computing → Combinatorics on words

Keywords and phrases Hyperbolic group, Existential theory, EDT0L language, PSPACE

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.110

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version `https://arxiv.org/abs/1902.07349`

Funding Research supported by Australian Research Council (ARC) Project DP160100486, and a Follow-On Grant from the International Centre of Mathematical Sciences (ICMS), Edinburgh.
Laura Ciobanu: Supported by EPSRC grant EP/R035814/1.

Acknowledgements The authors wish to thank Yago Antolín, Alex Bishop, François Dahmani, Volker Diekert, Michal Ferov and Jim Howie for helpful conversations, and the anonymous reviewers for their feedback and corrections.

1 Introduction

Hyperbolic groups were introduced by Gromov in 1987 [25], and play a significant role in group theory and geometry [12, 33, 40]. Virtually free groups, small cancellation groups, and the fundamental groups of extensive classes of negative curvature manifolds are important examples (see [1] for background). In a certain probabilistic sense made precise in [26, 37, 41], almost all finitely generated groups are hyperbolic. They admit very efficient solutions to the word and conjugacy problems [21, 27, 28], and extremely nice language-theoretic properties, for example the set of all geodesics over any generating set is regular (see Lemma 16), and forms a biautomatic structure [22]. They are exactly the groups which admit context-free multiplication tables [23], and have a particularly simple characterisation in terms of rewriting systems [6, 35] (see Lemma 13).

In this paper we consider systems of equations and inequations in hyperbolic groups, building on and generalising work done in the area of solving equations over various groups and monoids in PSPACE. Starting with work of Plandowski [38], many prominent researchers have given PSPACE algorithms [7, 14, 16, 17, 18, 30, 31] to find (all) solutions to systems of equations over free monoids, free groups, partially commutative monoids and groups, and virtually free groups (that is, groups which have a free subgroup of finite index).

The satisfiability of equations over torsion-free hyperbolic groups is decidable by the work of Rips and Sela [39], who reduced the problem in hyperbolic groups to solving equations in free groups, and then calling on Makanin's algorithm [36]. Kufleitner proved PSPACE for decidability in the torsion-free case [34], without an explicit complexity bound, by following Rips-Sela and then using Plandowski's result [38]. Dahmani and Guirardel radically extended Rips and Sela's work to all hyperbolic groups (with torsion), by reducing systems of equations to systems over virtually free groups, which they then reduced to systems of *twisted* equations over free groups [11]. In terms of describing solution sets, Grigorchuck and Lysionok gave efficient algorithms for the special case of quadratic equations [24].

Here we combine Rips, Sela, Dahmani and Guirardel's approach with recent work of the authors with Diekert [7, 14, 15] to obtain the following results.

▶ **Theorem 1** (Torsion-free). *Let G be a torsion-free hyperbolic group with finite symmetric generating set S. Let Φ be a system of equations and inequations of size n (see Section 2 for a precise definition of input size). Then the set of all solutions, as tuples of shortlex geodesic words over S, is EDT0L. Moreover there is an* $\mathsf{NSPACE}(n^2 \log n)$ *algorithm which on input Φ prints a description for the EDT0L grammar.*

▶ **Theorem 2** (Torsion). *Let G be a hyperbolic group with torsion with finite symmetric generating set S. Let Φ be a system of equations and inequations of size n (see Section 2 for a precise definition of input size). Then the set of all solutions, as tuples of shortlex geodesic words over S, is EDT0L. Moreover there is an* $\mathsf{NSPACE}(n^4 \log n)$ *algorithm which on input Φ prints a description for the EDT0L grammar.*

A corollary of Theorems 1 and 2 is that the existential theory for hyperbolic groups can be decided in $\mathsf{NSPACE}(n^2 \log n)$ for torsion-free and $\mathsf{NSPACE}(n^4 \log n)$ for groups with torsion. Another consequence of our work is that we can decide in the same space complexity as above whether or not the solution set is empty, finite or infinite (see [8]).

EDT0L is a surprisingly low language complexity for this problem. EDT0L languages are playing an increasingly useful role in group theory, not only in describing solution sets to equations in groups [7, 14, 17], but more generally [4, 5, 9].

The paper is organised as follows. We briefly set up some notation for solution sets and input size in Section 2. We then give an informal description of the entire argument for the torsion-free case in Section 3. This overview uses various concepts which are defined more carefully afterwards, but we hope that having the entire argument in one place is useful for the reader to understand the "big picture" before descending into the details. Section 4 develops necessary material on EDT0L and space complexity. Section 5 covers the necessary background on hyperbolic groups, including the key step to obtain a full solution set (as tuples of shortlex geodesics) from a *covering solution set* (see Definition 3(iii)). In Section 6 we use Rips and Sela's *canonical representatives* (see [8, Appendix A]) in torsion-free hyperbolic groups, to reduce the problem of finding solutions in a torsion-free hyperbolic group to finding solutions in the free group on the same generators as the hyperbolic one. We show that if the input system has size n then the resulting system in the free group has size $O(n^2)$. Applying [7] produces a covering solution set in $O(n^2 \log n)$ nondeterministic space, from

which we obtain the full set of solutions as shortlex geodesics in the original group, as an EDT0L language, in the same space complexity. In Section 7 we prove the general case for hyperbolic groups with torsion, following Dahmani and Guirardel who construct canonical representatives in a graph containing the Cayley graph of the hyperbolic group, and working in an associated virtually-free group.

2 Notations for equations and solution sets

Let G be a fixed group with finite symmetric generating set S. Let $\pi\colon S^* \to G$ be the natural projection map. Let $\{X_1, \ldots, X_m\}$, $m \geqslant 1$, be a set of variables to which we adjoin their formal inverses X_i^{-1} and denote by $\mathcal{X}$ the union $\{X_i, X_i^{-1} \mid 1 \leqslant i \leqslant m\}$. Let $\mathcal{C} = \{a_1, \ldots, a_k\} \subseteq G$ be a set of constants and

$$\Phi = \{\varphi_j(\mathcal{X}, \mathcal{C}) = 1\}_{j=1}^{h} \cup \{\varphi_j(\mathcal{X}, \mathcal{C}) \neq 1\}_{j=h+1}^{s} \tag{1}$$

be a set of s equations and inequations in G, where the length of each (in)equation is l_i. Then the total length of the equations is $n = \sum_{i=1}^{s} l_i$, and we take $|\Phi| = n$ as the input size in the remainder of the paper.

A tuple $(g_1, \ldots, g_m) \in G^m$ *solves* an equation [resp. inequation] φ_j in Φ if replacing each variable X_i by g_i (and X_i^{-1} by g_i^{-1}) produces an identity [resp. inequality] in the group as follows:

$$\varphi_j(g_1, \ldots, g_m, a_1, \ldots, a_k) = 1 \text{ [resp. } \varphi_j(g_1, \ldots, g_m, a_1, \ldots, a_k) \neq 1\text{]}.$$

A tuple $(g_1, \ldots, g_m) \in G^m$ solves Φ if it simultaneously solves φ_j for all $1 \leqslant j \leqslant s$.

▶ **Definition 3.**

(i) *The* group element solution set *to* Φ *is the set*

$$Sol_G(\Phi) = \{(g_1, \ldots, g_m) \in G^m \mid (g_1, \ldots, g_m) \text{ solves } \Phi\}.$$

(ii) *Let* $T \subseteq S^*$ *and* $\#$ *a symbol not in* S. *The* full set of T-solutions *is the set*

$$Sol_{T,G}(\Phi) = \{w_1 \# \ldots \# w_m \mid w_i \in T, (\pi(w_1), \ldots, \pi(w_m)) \text{ solves } \Phi\}.$$

(iii) *A set* $L \subseteq \{w_1 \# \ldots \# w_m \mid w_i \in S^*, 1 \leqslant i \leqslant m\}$ *is a* covering solution set *to* Φ *if*

$$\{(\pi(w_1), \ldots, \pi(w_k)) \mid w_1 \# \ldots \# w_m \in L\} = Sol_G(\Phi).$$

3 Overview of the proof

In a free group, the equation $xy = z$ has a solution in reduced words (that is, words which do not contain factors aa^{-1} for any $a \in S$) if and only if there exist words P, Q, R with $x = PQ, y = Q^{-1}R, z = PR$ in the free monoid with involution over S ([7, Lemma 4.1]). In a hyperbolic group this direct reduction to cancellation-free equations is no longer true: a triangle $xy = z$ where x, y, z are replaced by geodesics looks as in Figure 1a.

Rips and Sela [39] proved that in a torsion-free hyperbolic group one can define certain special words called *canonical representatives* so that a system of equations of the form $X_jY_j = Z_j, 1 \leqslant j \leqslant O(n)$ has solutions which are canonical representatives with the properties that their prefixes and suffixes coincide, as shown in Figure 1b, and the inner circle is the concatenation of three words with lengths in $O(n)$. Moreover, these canonical representatives are (λ, μ)-quasigeodesics (Definition 15) where the constants λ, μ depend only on the group.

(a) Using geodesics. **(b)** Using canonical representatives.

Figure 1 Solutions to $xy = z$ in the Cayley graph of a hyperbolic group.

We use these facts to devise the following algorithm, presented here for the torsion-free case. We treat the hyperbolic group G with finite generating set S as a constant. On input a system of equations and inequations as in (1) of size n:

1. Replace inequations by equations (by using a new variable and requiring that this variable is not trivial in the group, as explained in Section 6.3).
2. Triangulate the system, so that all equations have the form $X_jY_j = Z_j$. The size of the resulting system is still in $O(n)$. Suppose there are $q \in O(n)$ such equations.
3. Enumerate, one at a time, all possible tuples $\mathbf{c} = (c_{11}, c_{12}, c_{13}, \ldots, c_{q1}, c_{q2}, c_{q3})$ of words (say, in lex order) so that the length $\ell(c_{ji})$ with respect to S is bounded by a constant in $O(n)$. Note that the size of each tuple (the sum of the lengths of the c_{ij}) is in $O(n^2)$.
4. For each tuple $\mathbf{c}$, run Dehn's algorithm to check $c_{j1}c_{j2}c_{j3} =_G 1$ for $1 \leqslant j \leqslant q$. If this holds for all j, write down a system of $3q$ equations

$$X_j = P_j c_{j1} Q_j, Y_j = Q_j^{-1} c_{j2} R_j, Z_j = P_j c_{j3} R_j.$$

   Note that the resulting system, $\Phi_{\mathbf{c}}$, has size in $O(n^2)$.
5. We now call the algorithm of the authors and Diekert [7] to find all solutions to $\Phi_{\mathbf{c}}$ in the free group generated by S. This algorithm, on input of size $O(n^2)$, runs in $\mathsf{NSPACE}(n^2 \log n)$, and prints a description of the EDT0L grammar which generates all tuples of solutions as reduced words in S^*. Specifically it prints nodes and edges of a trim NFA which is the rational control for the EDT0L grammar (see Definition 4 below). Modify the algorithm so that the nodes printed include the label $\mathbf{c}$ which has length $O(n^2)$ (so does not affect the complexity).
6. Delete the current system stored, and move to the next tuple $\mathbf{c}$.
7. At the end, print out a new start node and ϵ edges to the start node of the NFA for the system $\Phi_{\mathbf{c}}$ for all $\mathbf{c}$ already printed.

The NFA that is printed gives an EDT0L grammar that generates a language of tuples which is a covering solution to the original system in the hyperbolic group. To obtain the full set of solutions as shortlex geodesic words we need to perform further steps. Using the facts that canonical representatives are (λ, μ)-quasigeodesics, and

- the full set of (λ, μ)-quasigeodesics, $Q_{S,\lambda,\mu}$
- the set of all pairs $\{(u, v) \in Q_{S,\lambda,\mu} \mid u =_G v\}$
- the set of all shortlex geodesics in G

are all regular, we can obtain from the covering solution an ET0L language, in the same space complexity (by Proposition 9 below), which represents the full set of solutions in shortlex geodesic words. Then finally, because of the special form of our solutions, we can apply a version of the *Copying Lemma* of Ehrenfeucht and Rozenberg [19] to show that in fact the resulting language of shortlex representatives is EDT0L in $\mathsf{NSPACE}(n^2 \log n)$.

Details for handling the case of hyperbolic groups with torsion also follows this general scheme, however finding the analogue of canonical representatives is harder in this case, so further work is required, and we describe this in Section 7.

4 E(D)T0L in PSPACE

4.1 ET0L and EDT0L languages

Let C be an alphabet. A *table* for C is a finite subset of $C \times C^*$. If (c, v) is in some table t, we say that (c, v) is a *rule* for c. A table t is *deterministic* if for each $c \in C$ there is exactly one $v \in C^*$ with $(c, v) \in t$.

If t is a table and $u \in C^*$ then we write $u \longrightarrow^t v$ to mean that v is obtained by applying rules from t to each letter of u. That is, $u = a_1 \dots a_n$, $a_i \in C$, $v = v_1 \dots v_n$, $v_i \in C^*$, and $(a_i, v_i) \in t$ for $1 \leqslant i \leqslant n$. If H is a set of tables and $r \in H^*$ then we write $u \longrightarrow^r v$ to mean that there is a sequence of words $u = v_0, v_1, \dots, v_n = v \in C^*$ such that $v_{i-1} \longrightarrow^{t_i} v_i$ for $1 \leqslant i \leqslant n$ where $r = t_1 \dots t_n$. If $R \subseteq H^*$ we write $u \longrightarrow^R v$ if $u \longrightarrow^r v$ for some $r \in R$.

▶ **Definition 4** ([2]). *Let Σ be an alphabet. We say that $L \subseteq \Sigma^*$ is an* ET0L *language if there is an alphabet C with $\Sigma \subseteq C$, a finite set $H \subset \mathcal{P}(C \times C^*)$ of tables, a regular language $R \subseteq H^*$ and a letter $c_0 \in C$ such that*

$$L = \{w \in \Sigma^* \mid c_0 \longrightarrow^R w\}.$$

In the case when every table $h \in R$ is deterministic, i.e. each $h \in R$ is in fact a homomorphism, we write $L = \{r(c_0) \in \Sigma^* \mid r \in R\}$ and say that L is EDT0L*. The set R is called the* rational control*, the symbol c_0 the* start symbol *and C the* extended alphabet*.*

4.2 Space complexity for E(D)T0L

Let $f\colon \mathbb{N} \to \mathbb{N}$ be a function. Recall an algorithm is said to run in $\mathsf{NSPACE}(f(n))$ if it can be performed by a non-deterministic Turing machine with a read-only input tape, a write-only output tape, and a read-write work tape, with the work tape restricted to using $O(f(n))$ squares on input of size n. The following definition formalises the idea of producing some E(D)T0L language (such as the solution set of some system of equations) in $\mathsf{NSPACE}(f(n))$, where the language is the output of a computation with input (such as a system of equations) of size n. We use the notation $L(\mathcal{A})$ to denote the language accepted by an automaton $\mathcal{A}$.

▶ **Definition 5.** *Let Σ be a (fixed) alphabet and $f\colon \mathbb{N} \to \mathbb{N}$ a function. If there is an $\mathsf{NSPACE}(f(n))$ algorithm that on input Ω of size n outputs the specification of an ET0L language $L_\Omega \subseteq \Sigma^*$, then we say that L_Ω is* ET0L in $\mathsf{NSPACE}(f(n))$*.*

Here the specification of L_Ω consists of:

- *an extended alphabet $C \supseteq \Sigma$,*
- *a start symbol $c_0 \in C$,*
- *a finite list of nodes of a (trim) NFA $\mathcal{A}$, labeled by some data, some possibly marked as initial and/or final,*
- *a finite list $\{(u, v, h)\}$ of edges of $\mathcal{A}$ where u, v are nodes and $h \in \mathcal{P}(C \times C^*)$ is a table such that $L_\Omega = \{w \in \Sigma^* \mid c_0 \to^{L(\mathcal{A})} w\}$.*

A language L_Ω is ET0L in $\mathsf{NSPACE}(f(n))$ *if, in addition, every table h labelling an edge of $\mathcal{A}$ is deterministic.*

Note that the entire print-out is not required to be in $O(f(n))$ space. Previous results of the authors with Diekert can now be restated as follows.

▶ **Theorem 6** ([7, Theorem 2.1]). *The set of all solutions to a system of size n of equations (with rational constraints), as reduced words, in a free group is EDT0L in* $\mathsf{NSPACE}(n \log n)$.

▶ **Theorem 7** ([15, Theorem 45]). *The set of all solutions to a system of size n of equations (with rational constraints), as words in a particular quasigeodesic normal form over a certain finite generating set, in a virtually free group is EDT0L in* $\mathsf{NSPACE}(n^2 \log n)$.

▶ Remark 8. In our applications below we have Ω representing some system of equations and inequations, with $|\Omega| = n$, and we construct algorithms where the extended alphabet C has size $|C| \in O(n)$ in the torsion-free case and $|C| \in O(n^2)$ in the torsion case. This means we can write down the entire alphabet C as binary strings within our space bounds. Moreover, each element (c, v) of any table we construct has v of (fixed) bounded length, so we can write down entire tables within our space bounds.

4.3 Closure properties

It is well known [32, Theorem 2.8] that ET0L is a full AFL (closed under homomorphism, inverse homomorphism, finite union, intersection with regular languages). Here we show the space complexity of an ET0L language is not affected by these operations.

▶ **Proposition 9.** *Let Σ, Γ be finite alphabets of fixed size, M an NFA of constant size with $L(M) \subseteq \Sigma^*$, and $\varphi\colon \Gamma^* \to \Sigma^*$, $\psi\colon \Sigma^* \to \Gamma^*$ homomorphisms. If $L_{\Omega_1}, L_{\Omega_2} \subseteq \Sigma^*$ are E(D)T0L in* $\mathsf{NSPACE}(f(n))$ *(on inputs Ω_1, Ω_2, respectively, with $|\Omega_1|, |\Omega_2| \in O(n)$) then*

- *(homomorphism) $\psi(L_{\Omega_1})$ is E(D)T0L in* $\mathsf{NSPACE}(f(n))$,
- *(intersection with regular) $L_{\Omega_1} \cap L(M)$ is E(D)T0L in* $\mathsf{NSPACE}(f(n))$,
- *(union) $L_{\Omega_1} \cup L_{\Omega_2}$ is E(D)T0L in* $\mathsf{NSPACE}(f(n))$,
- *(inverse homomorphism) $\varphi^{-1}(L_{\Omega_1})$ is ET0L in* $\mathsf{NSPACE}(f(n))$.

The proof is straightforward keeping track of complexity in the standard proofs [3, 10]. Note EDT0L is not closed under inverse homomorphism [20].

▶ **Proposition 10** (Projection onto a factor). *If $L_\Omega \subseteq \Sigma^*$ is E(D)T0L in* $\mathsf{NSPACE}(f(n))$ *on an input Ω of size n, and for some fixed integer s all words in L_Ω have the form $u_1 \# \ldots \# u_s$ with $u_i \in (\Sigma \setminus \{\#\})^*$, and $1 \leqslant i \leqslant j \leqslant s$, then*

$$L = \{u_i \# \ldots \# u_j \mid u_1 \# \ldots \# u_i \# \ldots \# u_j \# \ldots \# u_s \in L_\Omega\}$$

is E(D)T0L in $\mathsf{NSPACE}(f(n))$.

4.4 From ET0L to EDT0L

In computing the full solution set to equations as shortlex geodesic words, we will need to take inverse homomorphism. Even though in general the image under an inverse homomorphism of an EDT0L language is just ET0L, because of the special structure of solution sets we can apply the *Copying Lemma* of Ehrenfeucht and Rozenberg [19] to show the following.

▶ **Proposition 11.** *Let S be an alphabet and $h : S \to S'$ be a homomorphism of from S to a disjoint alphabet $S' = \{s' \mid s \in S\}$ defined by $h(s) = s'$. Let $\wr$ be a symbol not in $S \cup S'$ and define $h(\wr) = \wr$. Let L_1 be a set of words of the form $w \wr h(w)$ where $w \in S^*$. If L_1 is ET0L in* $\mathsf{NSPACE}(f(n))$, *then $L_2 = \{w \mid w \wr h(w) \in L_1\}$ is EDT0L in* $\mathsf{NSPACE}(f(n))$.

Proof. By [19], any nondeterministic table in the grammar for L_1 can be replaced by a finite number of deterministic tables (essentially, if nondeterminism allowed some letter $c \in C$ to produce two different results, then some word in L_1 would not have the form $w \wr h(w)$). So without loss of generality we can replace a table f containing $(c, v_1), \ldots, (c, v_k)$ by k tables f_i containing (c, v_i) only). This modification is clearly in the same space bound. Project onto the prefix using Proposition 10. ◀

5 Hyperbolic groups

5.1 Definitions

Recall the *Cayley graph* for a group G with respect to a finite symmetric generating set S is a directed graph $\Gamma(G, S)$ with vertices labeled by $g \in G$ and a directed edge (g, h) labeled by $s \in S$ whenever $h =_G gs$. Let $\ell(p)$, $i(p)$ and $f(p)$ resp. be the length, initial and terminal vertices of a path p in the Cayley graph. A path p is *geodesic* if $\ell(p)$ is minimal among the lengths of all paths q with the same endpoints. If x, y are two points in $\Gamma(G, S)$, we define $d(x, y)$ to be the length of a shortest path from x to y in $\Gamma(G, S)$.

▶ **Definition 12** (δ-hyperbolic group (Gromov)). *Let G be a group with finite symmetric generating set S, and let $\delta \geqslant 0$ be a fixed real number. If p, q, r are geodesic paths in $\Gamma(G, S)$ with $f(p) = i(q), f(q) = i(r), f(r) = i(p)$, we call $[p, q, r]$ a* geodesic triangle. *A geodesic triangle is* δ-slim *if p is contained in a δ-neighbourhood of $q \cup r$, that is, every point on one side of the triangle is within δ of some point on one of the other sides. (See for example Figure 1a.) We say (G, S) is* δ-hyperbolic *if every geodesic triangle in $\Gamma(G, S)$ is δ-slim. We say (G, S) is* hyperbolic *if it is δ-hyperbolic for some $\delta \geqslant 0$.*

It is a straightforward to show that being hyperbolic is independent of choice of finite generating set. Thus we say G is hyperbolic if (G, S) is for some finite generating set S.

▶ **Lemma 13** (Dehn presentation). *G is hyperbolic if and only if there is a finite list of pairs of words $(u_i, v_i) \in S^* \times S^*$ with $|u_i| > |v_i|$ and $u_i =_G v_i$ such that the following holds: if $w \in S^*$ is equal to the identity of G then it contains some u_i as a factor.*

This gives an algorithm to decide whether or not a word $w \in S^*$ is equal to the identity: while $\ell(w) > 0$, look for some u_i factor. If there is none, then $w \neq_G 1$. Else replace u_i by v_i (which is shorter). This procedure is called *Dehn's algorithm.*

▶ **Lemma 14.** *Dehn's algorithm runs in (linear time and) linear space.*

▶ **Definition 15** (Quasigeodesic). *For $\lambda \geqslant 1, \mu \geqslant 0$ real numbers, a path p in $\Gamma(G, S)$ is a* (λ, μ)-quasigeodesic *if for any subpath q of p we have $\ell(q) \leqslant \lambda d(i(q), f(q)) + \mu$.*

Throughout this article, we assume G is a fixed hyperbolic group which we treat as a constant for complexity purposes. We assume we are given (G, S), the constant δ, the finite list of pairs for Dehn's algorithm, and any other constants depending only on the group.

5.2 Languages in hyperbolic groups

▶ **Proposition 16.** *Let G be a fixed hyperbolic group with finite generating set S, $\lambda \geqslant 1, \mu \geqslant 0$ constants with $\lambda \in \mathbb{Q}$ and μ sufficiently large. Then the following sets are regular languages.*

1. *The set of all geodesics over S.*
2. *The set of all shortlex geodesics over S.*
3. *The set of all (λ, μ)-quasigeodesics, $Q_{S,\lambda,\mu} \subseteq S^*$.*

Furthermore, the set of all pairs of words $(u, v) \in Q^2_{S,\lambda,\mu}$ such that $u =_G v$ is accepted by an asynchronous 2-tape automaton.

See [22, 29].

5.3 Main reduction result

Here is our key technical result.

▶ **Proposition 17** (Covering to full solution sets). *Let G be a hyperbolic group with finite symmetric generating set S. Let $h : S \to S'$ and $Q_{S,\lambda,\mu}$ be as defined above, $\#, \wr$ symbols not in $S \cup S'$, $h(\#) = \#, h(\wr) = \wr$, and $\mathcal{T} \subseteq Q_{S,\lambda,\mu}$ a regular set of quasigeodesic words in bijection with G. Suppose $L_1 \subseteq (S \cup S' \cup \{\#, \wr\})^*$ consists of words of the form*

$$u_1 \# \dots \# u_r \wr h(v_1) \# \dots \# h(v_r),\ u_i, v_i \in Q_{S,\lambda,\mu}, u_i =_G v_i, 1 \leqslant i \leqslant r.$$

If L_1 is ET0L in NSPACE$(f(n))$*, then*

1. $L_Q = \{w_1 \# \dots \# w_r \wr h(z_1) \# \dots \# h(z_r) \mid \exists u_1 \# \dots \# u_r \wr h(v_1) \# \dots \# h(v_r) \in L_1, w_i =_G z_i =_G u_i, w_i, z_i \in Q_{S,\lambda,\mu}\}$ *is ET0L in* NSPACE$(f(n))$.
2. $L_{\mathcal{T}} = \{w_1 \# \dots \# w_r \mid \exists u_1 \# \dots \# u_r \wr h(v_1) \# \dots \# h(v_r) \in L_1, w_i =_G u_i, w_i \in \mathcal{T}\}$ *is EDT0L in* NSPACE$(f(n))$.

The proof involves a series of operations as in Proposition 9–11, see [8] for further details. Note that the set of all shortlex geodesics is a suitable choice for $\mathcal{T}$ in the proposition.

6 Reduction from torsion-free hyperbolic to free groups

Section 3 contains an overview of the general algorithm for solving equations in torsion-free hyperbolic groups. Here we provide further details, and give a proof of the soundness and completeness of our algorithm. The algorithm relies on the existence and special properties of canonical representatives, whose construction is very technical (details are provided in [8]). Their existence guarantees that the solutions of a system in a torsion-free hyperbolic group generated by S can be found by solving an associated system in the free group on S, while the fact that they are quasigeodesics (see [8, Prop. 30]) allows us to apply the results of the previous sections to obtain the EDT0L characterisation of solutions in shortlex normal form.

▶ **Proposition 18.** *Let G be a torsion-free hyperbolic group, with finite symmetric generating set S. Let Φ be a system of equations and inequations of input size n as in Section 2. Let $h : S \to S', \#, \wr$ be as in Proposition 17. Then there exist $\lambda \geqslant 1, \mu \geqslant 0$ and*

$$L = \{w_1 \# \dots \# w_m \wr h(w_1) \# \dots \# h(w_m) \mid w_i \in Q_{S,\lambda,\mu}, 1 \leqslant i \leqslant m\}$$

such that $\{w \mid w \wr h(w) \in L\}$ is a covering solution for Φ, and L is EDT0L in NSPACE$(n^2 \log n)$.

Applying Proposition 17 immediately gives Theorem 1.

Proof. We produce a language L of quasigeodesic words over S such that the projection of any tuple in L is in the group element solution set $\mathrm{Sol}_G(\Phi)$ (soundness). We then prove (using [39, Corollary 4.4]) that any solution in $\mathrm{Sol}_G(\Phi)$ is the projection of some tuple in L (completeness). Our proof follows the outline presented in Section 3.

1. **Preprocessing.**
   - (Remove inequations) We first transform Φ into a system consisting entirely of equations by adding a variable x_D to $\mathcal{X}$ and replacing any inequation $\varphi_j(\mathcal{X}, \mathcal{A}) \neq 1$ by $\varphi_j(\mathcal{X}, \mathcal{A}) = x_D$, with the constraint $x_D \neq_G 1$.

- (Triangulation) We transform each equation into several equations of length 3, by introducing new variables. This can always be done (see the discussion in [7, Section 4]), and it produces approximately $\sum_{i=1}^{s} l_i \in O(n)$ triangular equations with set of variables $\mathcal{Z}$ where $m \leqslant |\mathcal{Z}| \in O(n)$ and $\mathcal{X} \subset \mathcal{Z}$. From now on assume that the system Φ consists of $q \in O(n)$ equations of the form $X_j Y_j = Z_j$ where $1 \leqslant j \leqslant q$.

2. **Lifting Φ to the free group on S.** In [39, Theorem 4.2] Rips and Sela define a constant, which they call "*bp*", that roughly bounds the circumference of the "centres" of the triangles whose edges are canonical representatives. We denote here *bp* by ρ, and note that $\rho \in O(q) = O(n)$ depends on δ and linearly on q. As described in Section 3 we run in lex order through all possible tuples of words $\mathbf{c} = (c_{11}, c_{12}, c_{13}, \ldots, c_{q1}, c_{q2}, c_{q3})$ with $c_{ji} \in S^*, \ell(c_{ji}) \leqslant \rho \in O(n)$. For each tuple $\mathbf{c}$ we use Dehn's algorithm to check $c_{j1}c_{j2}c_{j3} =_G 1$, and if this holds for all $1 \leqslant j \leqslant q$ we then construct a system $\Phi_{\mathbf{c}}$ of equations of the form

$$X_j = P_j c_{j1} Q_j,\ Y_j = Q_j^{-1} c_{j2} R_j,\ Z_j = P_j c_{j3} R_j,\ \ 1 \leqslant j \leqslant q, \tag{2}$$

which has size $O(n^2)$. In order to avoid an exponential size complexity we write down each system $\Phi_{\mathbf{c}}$ one at a time, so the space required for this step is $O(n^2)$. Let $\mathcal{Y} \supset \mathcal{Z} \supset \mathcal{X}$ be the new set of variables.

3. **Some observations.** We pause to make the following observations. Any solution to $\Phi_{\mathbf{c}}$ in the free group $F(S)$ is guaranteed to be a solution to Φ in the original hyperbolic group G. Thus if $S_1 \subseteq F(S)^m$ is a group element solution to $\Phi_{\mathbf{c}}$ then $\pi(S_1)$ is a group element solution to Φ in G. This will show soundness below.
   Secondly, if $(g_1, \ldots, g_m) \in G^m$ is a solution to Φ in the original hyperbolic group, [39, Theorem 4.2 and Corollary 4.4] (see Theorem 31 in [8]) guarantees that there exist canonical representatives $w_i \in Q_{S,\lambda,\mu}$ with $w_i =_G g_i$ for $1 \leqslant i \leqslant m$, which have reduced forms $u_i =_G w_i$ for $1 \leqslant i \leqslant m$, and our construction is guaranteed to capture any such collection of words. This will show completeness below.
   Thirdly, note that the constraint that a word $w \in S^*$ must be a (λ, μ)-quasigeodesic and satisfy $w =_G 1$ implies that $\ell(w) \leqslant \mu$. Therefore we can construct a DFA $\mathcal{D}$ which accepts all words in S^* equal to 1 in the hyperbolic group G of length at most μ in constant space (using for example Dehn's algorithm). In our next step, we will use this rational constraint to handle the variable x_D added in the first step above (to remove inequalities).
   Now let us complete the construction by finding the covering solution required.

4. **Covering solution set.** We now run the algorithm from [7] (which we will refer to as the CDE algorithm) which takes input $\Phi_{\mathbf{c}}$, which has size in $O(n^2)$, plus the rational constraint $x_D \notin L(\mathcal{D})$, plus for each $y \in \mathcal{Y}$ the rational constraint that the solution for y is a word in $Q_{S,\lambda,\mu}$. Since these constraints have constant size (depending only on the group G, not the system Φ), they do not contribute to the $O(n^2)$ size of the input to the CDE algorithm.
   We make two modifications to the details of the CDE algorithm. First, every node printed by the algorithm should include the additional label $\mathbf{c}$. (This ensures the NFA we print for each system $\Phi_{\mathbf{c}}$ is distinct.) This does not affect the complexity since $\mathbf{c}$ has size in $O(n^2)$.
   Second, so that we can apply Proposition 11 later, we modify the form of "extended equations" in [7] by inserting the factor $\wr h(W)$ in the appropriate position(s). This simply increases the size of the nodes by a factor (of two).

We run the CDE algorithm to print an NFA (possibly empty) for each $\Phi_{\mathbf{c}}$, which is the rational control for an EDT0L grammar that produces all solutions as freely reduced words for elements of $F(S)$ which correspond to solutions as (λ, μ)-quasigeodesics to the same system $\Phi_{\mathbf{c}}$ in the hyperbolic group. If $(w_1, \ldots, w_m)$ is a solution in canonical representatives to Φ then $(u_1, \ldots, u_m, \ldots u_{|\mathcal{Y}|})$ will be included in the solution to $\Phi_{\mathbf{c}}$ output by the CDE algorithm, with u_i the reduced forms of w_i for $1 \leqslant i \leqslant m$. This shows completeness once we union the grammars from all systems $\Phi_{\mathbf{c}}$ together.
Adding a new start node with edges to each of the start nodes of the NFA's with label $\mathbf{c}$, we obtain a rational control for the EDT0L grammar generating L as required. The space required is exactly that required by the CDE algorithm on input $O(n^2)$, which is $\mathsf{NSPACE}(n^2 \log n)$. ◀

7 Reduction from hyperbolic with torsion to virtually free groups

In the case of a hyperbolic group G with torsion, the general approach of Rips and Sela can still be applied, but the existence of canonical representatives is not always guaranteed (see Delzant [13, Rem.III.1]). To get around this, Dahmani and Guirardel "fatten" the Cayley graph $\Gamma(G, S)$ of G to a larger graph $\mathcal{K}$ which contains $\Gamma(G, S)$ (in fact $\Gamma(G, S)$ with midpoints of edges included), and solve equations in G by considering equalities of paths in $\mathcal{K}$. More precisely, $\mathcal{K}$ is the 1-skeleton of the barycentric subdivision of a Rips complex of G (see [8] for definitions).

▶ **Definition 19.** *Let γ, γ' be paths in $\mathcal{K}$.*

- **(i)** *We denote by $i(\gamma)$ the initial vertex of γ, by $f(\gamma)$ the final vertex of γ, and by $\overline{\gamma}$ the reverse of γ starting at $f(\gamma)$ and ending at $i(\gamma)$.*
- **(ii)** *We say that γ is* reduced *if it contains no backtracking, that is, no subpath of length 2 of the form $e\overline{e}$.*
- **(iii)** *We write $\gamma\gamma'$ for the concatenation of γ, γ' if $i(\gamma') = f(\gamma)$.*
- **(iv)** *Two paths in $\mathcal{K}$ are homotopic if one can obtain a path from the other by adding or deleting backtracking subpaths. Each homotopy class has a unique reduced representative.*

Let V be the set of all homotopy classes $[\gamma]$ of paths γ in $\mathcal{K}$ with $i(\gamma) = 1_G$, and $f(\gamma) \in G$. For $[\gamma]$, $[\gamma'] \in V$ define their product $[\gamma][\gamma'] = [\gamma^v \gamma']$, where $\gamma^v \gamma'$ denotes the concatenation of γ and the translate ${}^v\gamma'$ of γ' by $v = f(\gamma)$, and let $[\gamma]^{-1}$ be the homotopy class of ${}^{v^{-1}}\overline{\gamma}$. Then V is a group that projects onto G by the final vertex map f, that is, $f : V \twoheadrightarrow G$ is a surjective homomorphism. Moreover, since G has an action on $\mathcal{K}$ induced by the natural action on its Rips complex, V will act on $\mathcal{K}$ as well. This gives rise to an action of V onto the universal cover T (which is a tree) of $\mathcal{K}$, and [11, Lemma 9.9] shows that the quotient T/V is a finite graph (isomorphic to $\mathcal{K}/G$) of finite groups, and so V is virtually free.

We assume that the algorithmic construction (see [11, Lemma 9.9]) of a presentation for V is part of the preprocessing of the algorithm, will be treated as a constant, and will not be included in the complexity discussion.

The first step in solving a system Φ of equations in G is to translate Φ into identities between quasigeodesic paths (with start and end point in G) in $\mathcal{K}$, defined as $\mathcal{QG}_{\lambda_1,\mu_1}(V)$ in (5) in [8], paths which can be seen as the analogues of the canonical representatives from the torsion-free case. This can be done by Proposition 9.8 [11]. The second step in solving Φ is to express the equalities of quasigeodesic paths in $\mathcal{K}$ in terms of equations in the virtually free group V based on $\mathcal{K}$. Finally, Proposition 9.10 [11] shows it is sufficient to solve the systems of equations in V in order to obtain the solutions of the system Φ in G.

In the virtually free group V we will use the results from [15]. Let Y be the generating set of V and $T \subseteq Y^*$ the set of normal forms for V over Y as in [15, Remark 44, page 50], and let

$$\mathrm{Sol}_{T,V}(\Psi) = \{w_1 \# \dots \# w_m) \in T^n \mid (\pi(w_1), \dots, \pi(w_m)) \text{ solves } \Psi \text{ in } V\}$$

be the language of T-solutions in V of a system Ψ of size $|\Psi| = O(k)$; by [15] the language $\mathrm{Sol}_{T,V}(\Psi)$ consists of (λ_Y, μ_Y)-quasigeodesics and is EDT0L in $\mathsf{NSPACE}(k^2 \log k)$ over Y.

▶ **Proposition 20.** *Let G be a hyperbolic group with torsion, with finite symmetric generating set S. Let Φ be a system of equations and inequations with $|\Phi| = n$ as in Section 2. Let $h : S \to S', \#, \wr$ be as in Proposition 17. Then there exist $\lambda \geqslant 1, \mu \geqslant 0$ and*

$$L = \{w_1 \# \dots \# w_m \wr h(v_1) \# \dots \# h(v_m) \mid w_i, v_i \in Q_{S,\lambda,\mu}, w_i =_G v_i, 1 \leqslant i \leqslant m\}$$

such that $\{w \mid w \wr h(v) \in L\}$ is a covering solution for Φ, and L is ET0L in $\mathsf{NSPACE}(n^4 \log n)$.

Again applying Proposition 17 immediately gives Theorem 2.

Before proving Proposition 20 we need to show how one can translate between elements and words in V over the generating set Y, and elements and words in G over S via the graph $\mathcal{K}$, so that the EDT0L characterisation of languages is preserved.

▶ **Notation.** *Let Z be some generating set of V and let $\pi \colon Z^* \to V$ be the standard projection map from words to group elements in V.*

(i) *For each $z_i \in Z$ there exists a unique reduced path p_i in $\mathcal{K}$ with $i(p_i) = 1_G$ and $f(p_i) \in G$; by concatenation for each word $w = z_{i_1} \dots z_{i_k}$ over Z there is then a unique path denoted*

$$p_w = p_{i_1} \dots p_{i_k} \tag{3}$$

with $i(p_w) = 1_{\mathcal{K}} = 1_G$ and $f(p_w) \in G$.

(ii) *For each $z_i \in Z$, assign a geodesic path γ_i in the Cayley graph $\Gamma(G, S)$ such that $i(\gamma_i) = 1_G$ and $f(\gamma_i) = f(p_i) \in G$, where p_i as in (i). Let $\sigma \colon Z^* \to S^*$ be the map/substitution given by $\sigma(z_i) = \gamma_i$; by concatenation one can associate to each word $w = z_{i_1} \dots z_{i_k}$ over Z a path in $\Gamma(G, S)$ denoted*

$$\gamma_w = \gamma_{i_1} \dots \gamma_{i_k} = \sigma(w) \tag{4}$$

with $i(p_w) = 1_G$ and $f(\gamma_w) = f(w) \in G$.

(iii) *There exists a unique reduced path, denoted $p_{\pi(w)}$, which is homotopic to p_w.*

Proof of Proposition 20. The algorithm to produce the language of solutions for Φ is similar to that outlined in Section 3 and detailed in the proof of Proposition 18, but it applies to different groups. The triangulation of Φ and introduction of a variable with rational constraint to deal with the inequations proceeds in the same manner. Again, we suppose after preprocessing we have $q \in O(n)$ triangular equations.

Then for $\kappa \in O(n)$ as in [8, Prop 36] define $V_{\leqslant \kappa} = \{[\gamma] \in V \mid \gamma \text{ reduced and } \ell_{\mathcal{K}}(\gamma) \leqslant \kappa\}$. One lifts the system Φ in G to a finite set of systems $\Psi_{\mathbf{c}}$ in the virtually free group V, one system for each q-tuple $\mathbf{c}$ of triples (c_1, c_2, c_3) with $c_i \in V_{\leqslant \kappa}$ and such that $f(c_1 c_2 c_3) = 1_G$, as in [8, Prop 36]. We enumerate these tuples by enumerating triples of words (v_1, v_2, v_3) over the generating set Y of V with $\ell_Y(v_i) \leqslant \kappa_Y$, where $\kappa_Y \in O(q)$ is a constant depending on κ, as in Lemma 21(ii). By Lemma 21(ii) the tuples of path triples $(p_{v_1}, p_{v_2}, p_{v_3})$ (see (3))

in $\mathcal{K}$ contain all q-tuples of triples (c_1, c_2, c_3) with $c_i \in V_{\leqslant \kappa_Y}$, up to homotopy. Then for each triple (v_1, v_2, v_3) we check whether $f(v_1v_2v_3) = 1_G$, and this is done by checking whether $\sigma(v_1)\sigma(v_2)\sigma(v_3) =_G 1$ using the Dehn algorithm in G.

Then each system $\Psi_{\mathbf{c}}$ is obtained as in (2) in the proof of Proposition 18 and has input size $O(q^2) \in O(n^2)$ since it has $O(q)$ equations, each of length in $O(q)$, and the factors c_i inserted also have length in $O(q)$. For each system $\Psi_{\mathbf{c}}$ over V we apply the algorithm in [15] and obtain the set of solutions $\mathrm{Sol}_{T,V}(\Psi_{\mathbf{c}})$ as an EDT0L in $\mathsf{NSPACE}((q^2)^2 \log(q^2)) = \mathsf{NSPACE}(n^4 \log n)$ of (λ_Y, μ_Y)-quasigeodesics over Y.

Now let $\mathcal{Q}\mathrm{Sol}_{T,V}(\Psi_{\mathbf{c}})$ be the set of all (λ_1', μ_1')-quasigeodesics which represent solutions of $\Psi_{\mathbf{c}}$ in V over Y. By Proposition 17 this language is ET0L and by Corollary 22 it contains at least one word over Y for each solution in $\mathcal{QG}_{\lambda_1,\mu_1}(V)$.

Then $\sigma(\mathcal{Q}\mathrm{Sol}_{T,V}(\Psi_{\mathbf{c}}))$ is ET0L since ET0L languages are preserved by substitutions, and by [8, Prop 36] $\mathcal{S} = \cup_{\mathbf{c}}\sigma(\mathcal{Q}\mathrm{Sol}_{T,V}(\Psi_{\mathbf{c}}))$ contains $\mathrm{Sol}_G(\Phi)$, so it is a covering solution set of Φ. By Lemma 23 the set $\mathcal{S}$ consists of at least one (λ_G, μ_G)-quasigeodesic over S for each solution, and then by intersection with the regular set Q_{S,λ_G,μ_G} of quasigeodesics in G over S we obtain a set of solutions for Φ consisting of (λ_G, μ_G)-quasigeodesics.

Finally, we run the modified DE algorithm (inserting the additional $\wr h(W)$ and label $\mathbf{c}$ for each node printed) to print an NFA for each $\Phi_{\mathbf{c}}$ for the EDT0L grammar for $\mathrm{Sol}_{T,V}(\Psi_{\mathbf{c}})$, which we union using an extra start node as before. From the above work this grammar generates the language L as required. ◀

▶ **Lemma 21.**

(i) *If $c \in V$ and the reduced path representing c in $\mathcal{K}$ is an (a, b)-quasigeodesic, then there exists a word w on Y representing c such that w is an (a', b')-quasigeodesic, where a', b' depend on a, b and Y.*

(ii) *If $c \in V$ and the length of the reduced path representing c in $\mathcal{K}$ is $\leqslant L$, then there exists a word w on Y representing c such that $\ell_Y(w) \leqslant L_Y$, where L_Y depends on L and Y.*

▶ **Corollary 22.** *For any element $v \in \mathcal{QG}_{\lambda_1,\mu_1}(V)$ there is a (λ_1', μ_1')-quasigeodesic word over Y representing v, where λ_1', μ_1' depend on λ_1, μ_1 and Y.*

▶ **Lemma 23.** *Let w be a (λ_1', μ_1')-quasigeodesic word over Y. Then if the reduced path $p_{\pi(w)}$ is (a, b)-quasigeodesic in $\mathcal{K}$ the (unreduced) path p_w is $(a_\mathcal{K}, b_\mathcal{K})$-quasigeodesic in $\mathcal{K}$, where $(a_\mathcal{K}, b_\mathcal{K})$ depend on a, b, λ_1', μ_1' and Y.*

Moreover, $\sigma(w)$ is a (λ_G, λ_G)-quasigeodesic over S in the hyperbolic group G, where λ_G, λ_G depend on λ_1, μ_1 and Y.

Proof. Consider the generating set $Z = Y \cup V_{\leqslant 3}$ for V and let λ_Z, μ_Z be such that any (λ_1', μ_1')-quasigeodesic over Y is (λ_Z, μ_Z)-quasigeodesic over Z. Let $M = \max\{l_\mathcal{K}(p_y) \mid y \in Y\}$. That is, M is the maximal length of a generator in Y with respect to the associated reduced path length in $\mathcal{K}$. We will show the statement in the lemma holds for $(a_\mathcal{K}, b_\mathcal{K}) = (a, b + M\mu_Z)$.

We say that a subpath s_w of p_w is a maximal backtrack if $p_w = ps_wp'$, s_w is homotopic to an empty path (via the elimination of backtrackings), and s_w is not contained in a longer subpath of p_w with the same property. This implies there is a point A on p_w such that s_w starts and ends at A, and such a maximal backtrack traces a tree in $\mathcal{K}$. We can then write $p_w = a_1s_1a_2 \dots s_{n-1}a_n$, where a_i are (possibly empty) subpaths of p_w and s_i are maximal backtracks; thus $p_{\pi(w)} = a_1a_2 \dots a_n$. If $l_\mathcal{K}(s_i) \leqslant M\mu_Z$ for all i, then the result follows immediately. Otherwise there exists an s_i with $l_\mathcal{K}(s_i) > M\mu_Z$, and we claim that we can write s_i in terms of a word over Z that is not a quasigeodesic, which contradicts the assumption that w is quasigeodesic.

To prove the claim, suppose $i(s_i) = f(s_i) = A$. We have two cases: in the first case $A \in G$ then $\pi(s_i) =_V 1$ and s_i corresponds to a subword v of w for which $l_{\mathcal{K}}(p_v) \geqslant M\mu_Z$. But v represents a word over Z, so $l_{\mathcal{K}}(p_v) \leqslant l_Z(v)M$, and altogether $M\mu_Z \leqslant l_{\mathcal{K}}(p_v) \leqslant l_Z(v)M$. Since $|v|_Z = 0$ and v is a (λ_Z, μ_Z)-quasigeodesic word over Z^*, $l_Z(v) \leqslant \mu_Z$, which contradicts $l_Z(v) \geqslant \mu_Z$ from above.

In the second case $A \notin G$, so take a point $B \in G$ at distance 1 from A in $\mathcal{K}$ (this can always be done), and modify the word w to get w' over Z so that $p_{w'}$ in $\mathcal{K}$ includes the backtrack $[AB, BA]$ off the path p_w. Also modify s_i to obtain a new backtrack s'_i. Clearly $\pi(p_w) = \pi(p_{w'})$ and $\pi(s_i) = \pi(s'_i)$, and s'_i becomes a maximal backtrack of $p_{w'}$ which can be written as a word over the generators Z that represents the trivial element in V. We can the apply the argument from the first case.

The fact that $\sigma(w)$ is a (λ_G, μ_G)-quasigeodesic over S in the hyperbolic group G follows from the fact that $\mathcal{K}$ and $\Gamma(G, S)$ are quasi-isometric. ◀

References

1. J. M. Alonso, T. Brady, D. Cooper, V. Ferlini, M. Lustig, M. Mihalik, M. Shapiro, and H. Short. Notes on word hyperbolic groups. In *Group theory from a geometrical viewpoint (Trieste, 1990)*, pages 3–63. World Sci. Publ., River Edge, NJ, 1991.
2. Peter R. J. Asveld. Controlled iteration grammars and full hyper-AFL's. *Information and Control*, 34(3):248–269, 1977.
3. Peter R.J. Asveld. *A Characterization of ET0L and EDT0L Languages*. Number 129 in Memorandum / Department of Applied Mathematics. Department of Applied Mathematics, University of Twente, 1976.
4. Alex Bishop and Murray Elder. Bounded Automata Groups are co-ET0L. In Carlos Martín-Vide, Alexander Okhotin, and Dana Shapira, editors, *Language and Automata Theory and Applications*, pages 82–94, Cham, 2019. Springer International Publishing.
5. Tara Brough, Laura Ciobanu, Murray Elder, and Georg Zetzsche. Permutations of context-free, ET0L and indexed languages. *Discrete Math. Theor. Comput. Sci.*, 17(3):167–178, 2016.
6. James W. Cannon. The theory of negatively curved spaces and groups. In *Ergodic theory, symbolic dynamics, and hyperbolic spaces (Trieste, 1989)*, Oxford Sci. Publ., pages 315–369. Oxford Univ. Press, New York, 1991.
7. Laura Ciobanu, Volker Diekert, and Murray Elder. Solution sets for equations over free groups are EDT0L languages. *Internat. J. Algebra Comput.*, 26(5):843–886, 2016. doi:10.1142/S0218196716500363.
8. Laura Ciobanu and Murray Elder. Solutions sets to systems of equations in hyperbolic groups are EDT0L in PSPACE. *arXiv e-prints*, abs/1902.07349, February 2019. arXiv:1902.07349.
9. Laura Ciobanu, Murray Elder, and Michal Ferov. Applications of L systems to group theory. *Internat. J. Algebra Comput.*, 28(2):309–329, 2018. doi:10.1142/S0218196718500145.
10. Karel Culik, II. On some families of languages related to developmental systems. *Internat. J. Comput. Math.*, 4:31–42, 1974. doi:10.1080/00207167408803079.
11. François Dahmani and Vincent Guirardel. Foliations for solving equations in groups: free, virtually free, and hyperbolic groups. *J. Topol.*, 3(2):343–404, 2010. doi:10.1112/jtopol/jtq010.
12. François Dahmani and Vincent Guirardel. The isomorphism problem for all hyperbolic groups. *Geom. Funct. Anal.*, 21(2):223–300, 2011. doi:10.1007/s00039-011-0120-0.
13. T. Delzant. L'image d'un groupe dans un groupe hyperbolique. *Comment. Math. Helv.*, 70(2):267–284, 1995. doi:10.1007/BF02566008.
14. Volker Diekert and Murray Elder. Solutions of twisted word equations, EDT0L languages, and context-free groups. In *44th International Colloquium on Automata, Languages, and Programming*, volume 80 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 96, 14. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2017.

15 Volker Diekert and Murray Elder. Solutions to twisted word equations and equations in virtually free groups. *arXiv e-prints*, January 2017. arXiv:1701.03297.

16 Volker Diekert, Claudio Gutiérrez, and Christian Hagenah. The existential theory of equations with rational constraints in free groups is PSPACE-complete. In A. Ferreira and H. Reichel, editors, *Proc. 18th Annual Symposium on Theoretical Aspects of Computer Science (STACS'01), Dresden (Germany), 2001*, volume 2010 of *Lecture Notes in Computer Science*, pages 170–182. Springer-Verlag, 2001. doi:10.1007/3-540-44693-1_15.

17 Volker Diekert, Artur Jeż, and Manfred Kufleitner. Solutions of word equations over partially commutative structures. In *43rd International Colloquium on Automata, Languages, and Programming*, volume 55 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 127, 14. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2016.

18 Volker Diekert, Artur Jeż, and Wojciech Plandowski. Finding all solutions of equations in free groups and monoids with involution. *Inform. and Comput.*, 251:263–286, 2016. doi:10.1016/j.ic.2016.09.009.

19 A. Ehrenfeucht and G. Rozenberg. On Decomposing Some ET0L Languages Into Deterministic ET0L Languages, 1974. CU-CS-045-74. Computer Science Technical Reports (44). URL: https://scholar.colorado.edu/csci_techreports/44.

20 A. Ehrenfeucht and G. Rozenberg. On inverse homomorphic images of deterministic ET0L languages. In *Automata, languages, development*, pages 179–189. North-Holland, Amsterdam, 1976.

21 David Epstein and Derek Holt. The linearity of the conjugacy problem in word-hyperbolic groups. *Internat. J. Algebra Comput.*, 16(2):287–305, 2006. doi:10.1142/S0218196706002986.

22 David B. A. Epstein, James W. Cannon, Derek F. Holt, Silvio V. F. Levy, Michael S. Paterson, and William P. Thurston. *Word processing in groups*. Jones and Bartlett Publishers, Boston, MA, 1992.

23 Robert H. Gilman. On the definition of word hyperbolic groups. *Math. Z.*, 242(3):529–541, 2002.

24 R. I. Grigorchuk and I. G. Lysionok. A description of solutions of quadratic equations in hyperbolic groups. *Internat. J. Algebra Comput.*, 2(3):237–274, 1992. doi:10.1142/S0218196792000153.

25 M. Gromov. Hyperbolic groups. In *Essays in group theory*, volume 8 of *Math. Sci. Res. Inst. Publ.*, pages 75–263. Springer, New York, 1987. doi:10.1007/978-1-4613-9586-7_3.

26 M. Gromov. Random walk in random groups. *Geom. Funct. Anal.*, 13(1):73–146, 2003. doi:10.1007/s000390300002.

27 Derek F. Holt. Word-hyperbolic groups have real-time word problem. *Internat. J. Algebra Comput.*, 10(2):221–227, 2000. doi:10.1142/S0218196700000078.

28 Derek F. Holt, Markus Lohrey, and Saul Schleimer. Compressed decision problems in hyperbolic groups. In *36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019)*, volume 34 of *LIPIcs. Leibniz Int. Proc. Inform.* Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2019. doi:10.4230/LIPIcs.STACS.2019.34.

29 Derek F. Holt and Sarah Rees. Regularity of quasigeodesics in a hyperbolic group. *Internat. J. Algebra Comput.*, 13(5):585–596, 2003. doi:10.1142/S0218196703001560.

30 Artur Jeż. Recompression: a simple and powerful technique for word equations. *J. ACM*, 63(1):Art. 4, 51, 2016. doi:10.1145/2743014.

31 Artur Jeż. Word equations in nondeterministic linear space. In *44th International Colloquium on Automata, Languages, and Programming*, volume 80 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 95, 13. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2017.

32 Lila Kari, Grzegorz Rozenberg, and Arto Salomaa. L systems. In *Handbook of formal languages, Vol. 1*, pages 253–328. Springer, Berlin, 1997.

33 Olga Kharlampovich and Alexei Myasnikov. Definable sets in a hyperbolic group. *Internat. J. Algebra Comput.*, 23(1):91–110, 2013. doi:10.1142/S021819671350001X.

34 Manfred Kufleitner. Wortgleichungen in hyperbolischen Gruppen. Diplomarbeit, Universität Stuttgart, Fakultät Informatik, Germany, July 2001. URL: `http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=DIP-1922&engl=0`.

35 I. G. Lysionok. Some algorithmic properties of hyperbolic groups. *Izv. Akad. Nauk SSSR Ser. Mat.*, 53(4):814–832, 912, 1989. `doi:10.1070/IM1990v035n01ABEH000693`.

36 Gennadiĭ Semyonovich Makanin. Equations in a free group. *Izv. Akad. Nauk SSR*, Ser. Math. 46:1199–1273, 1983. English transl. in Math. USSR Izv. 21 (1983).

37 A. Yu. Ol′ shanskiĭ. Almost every group is hyperbolic. *Internat. J. Algebra Comput.*, 2(1):1–17, 1992. `doi:10.1142/S0218196792000025`.

38 Wojciech Plandowski. Satisfiability of word equations with constants is in PSPACE. *J. ACM*, 51(3):483–496, 2004. `doi:10.1145/990308.990312`.

39 E. Rips and Z. Sela. Canonical representatives and equations in hyperbolic groups. *Invent. Math.*, 120(3):489–512, 1995. `doi:10.1007/BF01241140`.

40 Z. Sela. Diophantine geometry over groups and the elementary theory of free and hyperbolic groups. In *Proceedings of the International Congress of Mathematicians, Vol. II (Beijing, 2002)*, pages 87–92. Higher Ed. Press, Beijing, 2002.

41 L. Silberman. Addendum to: "Random walk in random groups" [Geom. Funct. Anal. **13** (2003), no. 1, 73–146; MR1978492] by M. Gromov. *Geom. Funct. Anal.*, 13(1):147–177, 2003. `doi:10.1007/s000390300003`.

Differential Logical Relations, Part I: The Simply-Typed Case

Ugo Dal Lago
University of Bologna, Italy
INRIA Sophia Antipolis, France
ugo.dallago@unibo.it

Francesco Gavazzo
IMDEA Software Institute, Spain
francesco.gavazzo@gmail.com

Akira Yoshimizu
INRIA Sophia Antipolis, France
akiray@bp.iij4u.or.jp

Abstract

We introduce a new form of logical relation which, in the spirit of metric relations, allows us to assign each pair of programs a quantity measuring their distance, rather than a boolean value standing for their being equivalent. The novelty of differential logical relations consists in measuring the distance between terms not (necessarily) by a numerical value, but by a mathematical object which somehow reflects the interactive complexity, i.e. the type, of the compared terms. We exemplify this concept in the simply-typed lambda-calculus, and show a form of soundness theorem. We also see how ordinary logical relations and metric relations can be seen as instances of differential logical relations. Finally, we show that differential logical relations can be organised in a cartesian closed category, contrarily to metric relations, which are well-known *not* to have such a structure, but only that of a monoidal closed category.

2012 ACM Subject Classification Theory of computation → Lambda calculus

Keywords and phrases Logical Relations, λ-Calculus, Program Equivalence, Semantics

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.111

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version `https://arxiv.org/abs/1904.12137`

Funding The authors are partially supported by the ERC Consolidator Grant DLV-818616 DIAPASoN, as well as by the ANR projects 14CE250005 ELICA and 16CE250011 REPAS.

1 Introduction

Modern software systems tend to be heterogeneous and complex, and this is reflected in the analysis methodologies we use to tame their complexity. Indeed, in many cases the only way to go is to make use of compositional kinds of analysis, in which *parts* of a large system can be analysed in isolation, without having to care about the rest of the system, the *environment*. As an example, one could consider a component A and replace it with another (e.g. more efficient) component B without looking at the context $\mathcal{C}$ in which A and B are supposed to operate, see Figure 1. Of course, for this program transformation to be safe, A should be *equivalent* to B or, at least, B should be a *refinement* of A.

Program equivalences and refinements, indeed, are the cruxes of program semantics, and have been investigated in many different programming paradigms. When programs have an interactive behaviour, like in concurrent or higher-order languages, even *defining* a notion of

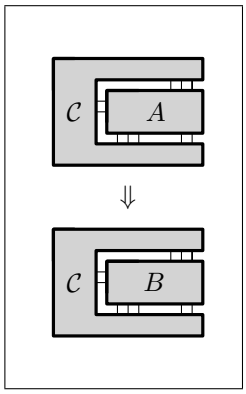


Figure 1 Replacing A with B.

program equivalence is not trivial, while coming out with handy methodologies for *proving* concrete programs to be equivalent can be quite challenging, and has been one of the major research topics in programming language theory, stimulating the development of techniques like logical relations [23, 20], applicative bisimilarity [1], and to some extent denotational semantics [26, 27] itself.

Coming back to our example, may we say anything about the case in which A and B are *not* equivalent, although behaving very similarly? Is there anything classic program semantics can say about this situation? Actually, the answer is negative: the program transformation turning such an A into B cannot be justified, simply because there is no guarantee about what the possible negative effects that turning A into B could have on the overall system formed by $\mathcal{C}$ and A. There are, however, many cases in which program transformations like the one we just described are indeed of interest, and thus desirable. Many examples can be, for instance, drawn from the field of *approximate computing* [21], in which equivalence-breaking program transformations are considered as beneficial *provided* the overall behaviour of the program is not affected too much by the transformation, while its intensional behaviour, e.g. its performance, is significantly improved.

One partial solution to the problem above consists in considering program *metrics* rather than program *equivalences*. This way, any pair of programs are dubbed being at a certain numerical distance rather than being merely equivalent (or not). This, for example, can be useful in the context of differential privacy [24, 6, 32] and has also been studied in the realms of domain theory [13, 5, 14, 16, 4] (see also [28] for an introduction to the subject) and coinduction [30, 29, 15, 9]. The common denominator among all these approaches is that on the one hand, the notion of a congruence, crucial for compositional reasoning, is replaced by the one of a *Lipschitz-continuous* map: any context should not amplify (too much) the distance between any pair of terms, when it is fed with either the former or the latter:

$$\delta(C[M], C[N]) \leq c \cdot \delta(M, N).$$

This enforces compositionality, and naturally leads us to consider metric spaces and Lipschitz functions as the underlying category. As is well known, this is not a cartesian closed category, and thus does *not* form a model of typed λ-calculi, unless one adopts linear type systems, or type systems in which the number of uses of each variable is kept track of, like FUZZ [24]. This somehow limits the compositionality of the metric approach [13, 17].

Even if one considers affine calculi, there are program transformations which are intrinsically unjustifiable in the metric approach. Consider the following two programs of type $REAL \to REAL$

$$M_{SIN} := \lambda x.\mathtt{sin}(x) \qquad M_{ID} := \lambda x.x.$$

The two terms compute two very different functions on the real numbers, namely the sine trigonometric function and the identity on $\mathbb{R}$, respectively. The euclidean distance $|\sin x - x|$ is unbounded when x ranges over $\mathbb{R}$. As a consequence, comparing M_{SIN} and M_{ID} using the so-called sup metric[1] as it is usually done in metric logical relations [24, 13] and applicative distances [17, 10], we see that their distance is infinite, and that the program transformation turning M_{SIN} into M_{ID} cannot be justified this way, for very good reasons. As highlighted by Westbrook and Chaudhuri [31], this is not the end of the story, at least if the environment in which M_{SIN} and M_{ID} operate feed either of them *only with* real numbers close to 0. If this is the case, M_{SIN} can be substituted with M_{ID} without affecting *too much* the overall behaviour of the system.

The key insight by Westbrook and Chaudhuri is that justifying program transformations like the one above requires taking the difference $\delta(M_{SIN}, M_{ID})$ between M_{SIN} and M_{ID} not merely as a number, but as a more structured object. What they suggest is to take $\delta(M_{SIN}, M_{ID})$ as *yet another program*, which however describes the difference between M_{SIN} and M_{ID}:

$$\delta(M_{SIN}, M_{ID}) := \lambda x.\lambda\varepsilon.|\sin x - x| + \varepsilon.$$

This reflects the fact that the distance between M_{SIN} and M_{ID}, namely the discrepancy between their output, depends not only on the discrepancy on the input, namely on ε, but also *on the input itself*, namely on x. It both x and ε are close to 0, $\delta(M_{SIN}, M_{ID})$ is itself close to 0.

In this paper, we develop Westbrook and Chaudhuri's ideas, and turn them into a framework of *differential logical relations*. We will do all this in a simply-typed λ-calculus with real numbers as the only base type. Starting from such a minimal calculus has at least two advantages: on the one hand one can talk about meaningful examples like the one above, and on the other hand the induced metatheory is simple enough to highlight the key concepts.

The contributions of this paper can be summarised as follows:

- After introducing our calculus $ST^{\lambda}_{\mathbb{R}}$, we define differential logical relations inductively on types, as ternary relations between pairs of programs and *differences*. The latter are mere set theoretic entities here, and the nature of differences between terms depends on terms' types.
- We prove a soundness theorem for differential logical relations, which allows us to justify compositional reasoning about terms' differences. We also prove a *finite difference theorem*, which stipulates that the distance between two simply-typed λ-terms is finite if mild conditions hold on the underlying set of function symbols.
- We give embeddings of logical and metric relations into differential logical relations. This witnesses that the latter are a generalisation of the former two.

[1] Recall that given (pseudo)metric spaces (X, d_X), (Y, d_Y) we can give the set Y^X of non-expansive maps between X and Y a (pseudo)metric space structure setting $d_{Y^X}(f, g) = \sup_{x \in X} d_Y(f(x), g(x))$

$$\frac{x:\tau\in\Gamma}{\Gamma\vdash x:\tau}\qquad\frac{}{\Gamma\vdash r:REAL}\qquad\frac{f_n\in\mathcal{F}_n}{\Gamma\vdash f_n:REAL^n\to REAL}\qquad\frac{\Gamma,x:\tau\vdash M:\rho}{\Gamma\vdash\lambda x.M:\tau\to\rho}$$

$$\frac{\Gamma\vdash M:\tau\to\rho\quad\Gamma\vdash N:\tau}{\Gamma\vdash MN:\rho}\qquad\frac{\Gamma\vdash M:\tau\quad\Gamma\vdash N:\rho}{\Gamma\vdash\langle M,N\rangle:\tau\times\rho}\qquad\frac{}{\Gamma\vdash\pi_1:\tau\times\rho\to\tau}\qquad\frac{}{\Gamma\vdash\pi_2:\tau\times\rho\to\rho}$$

$$\frac{\Gamma\vdash M:\tau\quad\Gamma\vdash N:\tau}{\Gamma\vdash \texttt{iflz}\ M\ \texttt{else}\ N:REAL\to\tau}\qquad\frac{\Gamma\vdash M:\tau\to\tau\quad\Gamma\vdash N:\tau}{\Gamma\vdash \texttt{iter}\ M\ \texttt{base}\ N:REAL\to\tau}$$

Figure 2 Typing rules for $ST^{\lambda}_{\mathbb{R}}$.

- Finally, we show that generalised metric domains, the mathematical structure underlying differential logical relations, form a cartesian closed category, contrarily to the category of metric spaces, which is well known not to have the same property.

Due to space constraints, many details have to be omitted, but can be found in an Extended Version of this work [12].

2 A Simply-Typed λ-Calculus with Real Numbers

In this section, we introduce a simply-typed λ-calculus in which the only base type is the one of real numbers, and constructs for iteration and conditional are natively available. The choice of this language as the reference calculus in this paper has been made for the sake of simplicity, allowing us to concentrate on the most crucial aspects, at the same time guaranteeing a minimal expressive power.

Terms and Types

$ST^{\lambda}_{\mathbb{R}}$ is a typed λ-calculus, so its definition starts by giving the language of *types*, which is defined as follows:

$$\tau,\rho ::= REAL \mid \tau\to\rho \mid \tau\times\rho.$$

The expression τ^n stands for $\underbrace{\tau\times\cdots\times\tau}_{n\text{ times}}$. The set of *terms* is defined as follows:

$$M,N ::= x \mid r \mid f_n \mid \lambda x.M \mid MN \mid \langle M,N\rangle \mid \pi_1 \mid \pi_2 \mid \texttt{iflz}\ M\ \texttt{else}\ N \mid \texttt{iter}\ M\ \texttt{base}\ N$$

where x ranges over a set $\mathbb{V}$ of variables, r ranges over the set $\mathbb{R}$ of real numbers, n is a natural number, and f_n ranges over a set $\mathcal{F}_n$ of total real functions of arity n. We do not make any assumption on $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$, apart from the predecessor $pred_1$ being part of $\mathcal{F}_1$. The family, in particular, could in principle contain non-continuous functions. The expression $\langle M_1,\ldots,M_n\rangle$ is simply a shortcut for $\langle\ldots\langle\langle M_1,M_2\rangle,M_3\rangle\ldots,M_n\rangle$. All constructs are self-explanatory, except for the `iflz` and `iter` operators, which are conditional and iterator combinators, respectively. An *environment* Γ is a set of assignments of types to variables in $\mathbb{V}$ where each variable occurs at most once. A *type judgment* has the form $\Gamma\vdash M:\tau$ where Γ is an environment, M is a term, and τ is a type. Rules for deriving correct typing judgments are in Figure 2, and are standard. The set of terms M for which $\cdot\vdash M:\tau$ is derivable is indicated as $CT(\tau)$.

$$\frac{}{V \Downarrow V} \qquad \frac{M \Downarrow f_n \quad N \Downarrow \langle L_1, \ldots, L_n \rangle \quad L_i \Downarrow r_i}{MN \Downarrow f(r_1, \ldots, r_n)} \qquad \frac{M \Downarrow \lambda x.L \quad N \Downarrow V \quad L\{V/x\} \Downarrow W}{MN \Downarrow W}$$

$$\frac{M \Downarrow \pi_1 \quad N \Downarrow \langle L, P \rangle \quad L \Downarrow V}{MN \Downarrow V} \qquad \frac{M \Downarrow \pi_2 \quad N \Downarrow \langle L, P \rangle \quad P \Downarrow V}{MN \Downarrow V}$$

$$\frac{M \Downarrow \texttt{iflz}\ L\ \texttt{else}\ P \quad N \Downarrow r \quad r < 0 \quad L \Downarrow V}{MN \Downarrow V} \qquad \frac{M \Downarrow \texttt{iflz}\ L\ \texttt{else}\ P \quad N \Downarrow r \quad r \geq 0 \quad P \Downarrow V}{MN \Downarrow V}$$

$$\frac{M \Downarrow \texttt{iter}\ L\ \texttt{base}\ P \quad N \Downarrow r \quad r < 0 \quad P \Downarrow V}{MN \Downarrow V}$$

$$\frac{M \Downarrow \texttt{iter}\ L\ \texttt{base}\ P \quad N \Downarrow r \quad r \geq 0 \quad L((\texttt{iter}\ L\ \texttt{base}\ P)(pred_1(r))) \Downarrow V}{MN \Downarrow V}$$

Figure 3 Operational semantics for $ST^{\lambda}_{\mathbb{R}}$.

Call-by-Value Operational Semantics

A static semantics is of course not enough to give meaning to a paradigmatic programming language, the dynamic aspects being captured only once an *operational* semantics is defined. The latter turns out to be very natural. *Values* are defined as follows:

$$V, W ::= r \mid f_n \mid \lambda x.M \mid \langle M, N \rangle \mid \pi_1 \mid \pi_2 \mid \texttt{iflz}\ M\ \texttt{else}\ N \mid \texttt{iter}\ M\ \texttt{base}\ N$$

The set of closed values of type τ is $CV(\tau) \subseteq CT(\tau)$, and the evaluation of $M \in CT(\tau)$ produces a value $V \in CV(\tau)$, as formalised by the rules in Figure 3, through the judgment $M \Downarrow V$. We write $M \Downarrow$ if $M \Downarrow V$ is derivable *for some* V. The absence of full recursion has the nice consequence of guaranteeing a form of termination:

▶ **Theorem 1.** *The calculus $ST^{\lambda}_{\mathbb{R}}$ is terminating: if $\cdot \vdash M : \tau$ then $M \Downarrow$.*

Theorem 1 can be proved by way of a standard reducibility argument. Termination implies the following.

▶ **Corollary 2.** *If $\cdot \vdash M : REAL$ then there exists a unique $r \in \mathbb{R}$ satisfying $M \Downarrow r$, which we indicate as $NF(M)$.*

Context Equivalence

A *context* C is nothing more than a term containing a single occurrence of a placeholder $[\cdot]$. Given a context C, $C[M]$ indicates the term one obtains by substituting M for the occurrence of $[\cdot]$ in C. Typing rules in Figure 2 can be lifted to contexts by generalising judgments to the form $\Gamma \vdash C[\Delta \vdash \cdot : \tau] : \rho$, by which one captures that whenever $\Delta \vdash M : \tau$, it holds that $\Gamma \vdash C[M] : \rho$. Two terms M and N such that $\Gamma \vdash M, N : \tau$ are said to be *context equivalent* [22] if for every C such that $\emptyset \vdash C[\Gamma \vdash \cdot : \tau] : REAL$ it holds that $NF(C[M]) = NF(C[N])$. Context equivalence is the largest adequate congruence, and is thus considered as the coarsest "reasonable" equivalence between terms. It can also be turned into a pseudometric [11, 10] – called *context distance* – by stipulating that

$$\delta(M, N) = \sup_{\emptyset \vdash C[\Gamma \vdash \cdot : \tau] : REAL} |NF(C[M]) - NF(C[M])|.$$

The obtained notion of distance, however, is bound to trivialise [11], given that $ST^{\lambda}_{\mathbb{R}}$ is not affine. Trivialisation of context distance highlights an important limit of the metric approach to program difference which, ultimately, can be identified with the fact that program distances

are sensitive to interactions with the environment. Our notion of a differential logical relation tackles such a problem from a different perspective, namely refining the concept of program distance to something which is not just a number, but is now able to take into account interactions with the environment.

Set-Theoretic Semantics

Before introducing differential logical relations, it is useful to remark that we can give $ST^{\lambda}_{\mathbb{R}}$ a standard set-theoretic semantics. To any type τ we associate the set $[\![\tau]\!]$, the latter being defined by induction on the structure of τ as follows:

$$[\![REAL]\!] = \mathbb{R}; \qquad [\![\tau \to \rho]\!] = [\![\tau]\!] \to [\![\rho]\!]; \qquad [\![\tau \times \rho]\!] = [\![\tau]\!] \times [\![\rho]\!].$$

This way, any closed term $M \in CT(\tau)$ is interpreted as an element $[\![M]\!]$ of $[\![\tau]\!]$ in a natural way (see, e.g. [20]). Up to now, everything we have said about $ST^{\lambda}_{\mathbb{R}}$ is absolutely standard, and only serves to set the stage for the next sections.

3 Making Logical Relations Differential

Logical relations can be seen as one of the *many* ways of defining when two programs are to be considered equivalent. Their definition is type driven, i.e., they can be seen as a *family* $\{\delta_\tau\}_\tau$ of binary relations indexed by types such that $\delta_\tau \subseteq CT(\tau) \times CT(\tau)$. This section is devoted to showing how all this can be made into differential logical relations.

The first thing that needs to be discussed is how to define the space of *differences* between programs. These are just boolean values in logical relations, become real numbers in ordinary metrics, and is type-dependent itself. A function $(\!|\cdot|\!)$ that assigns a set to each type is defined as follows:

$$(\!|REAL|\!) = \mathbb{R}^{\infty}_{\geq 0}; \qquad (\!|\tau \to \rho|\!) = [\![\tau]\!] \times (\!|\tau|\!) \to (\!|\rho|\!); \qquad (\!|\tau \times \rho|\!) = (\!|\tau|\!) \times (\!|\rho|\!);$$

where $\mathbb{R}^{\infty}_{\geq 0} = \mathbb{R}_{\geq 0} \cup \{\infty\}$. The set $(\!|\tau|\!)$ is said to be the *difference space* for the type τ and is meant to model the outcome of comparisons between closed programs of type τ. As an example, when τ is $REAL \to REAL$, we have that $(\!|\tau|\!) = \mathbb{R} \times \mathbb{R}^{\infty}_{\geq 0} \to \mathbb{R}^{\infty}_{\geq 0}$. This is the type of the function $\delta(M, N)$ we used to compare the two programs described in the Introduction.

Now, which structure could we endow $(\!|\tau|\!)$ with? First of all, we can define a partial order $\leq_\tau$ over $(\!|\tau|\!)$ for each type τ as follows:

$$\begin{array}{ll} r \leq_{REAL} s & \text{if } r \leq s \text{ as the usual order over } \mathbb{R}^{\infty}_{\geq 0}; \\ f \leq_{\tau\to\rho} g & \text{if } \forall x \in [\![\tau]\!].\forall t \in (\!|\tau|\!).f(x,t) \leq_\rho g(x,t); \\ (t,u) \leq_{\tau\times\rho} (s,r) & \text{if } t \leq_\tau s \text{ and } u \leq_\rho r. \end{array}$$

This order has least upper bounds and greater lower bounds, thanks to the nice structure of $\mathbb{R}^{\infty}_{\geq 0}$:

▶ **Proposition 3.** *For each type τ, $((\!|\tau|\!), \leq_\tau)$ forms a complete lattice.*

The fact that $(\!|\tau|\!)$ has a nice order-theoretic structure is not the end of the story. For every type τ, we define a binary operation $*_\tau$ as follows:

$$\begin{array}{ll} r *_{REAL} s = r + s \text{ if } r, s \in \mathbb{R}_{\geq 0}; & (f *_{\tau\to\rho} g)(V,t) = f(V,t) *_\rho g(V,t); \\ r *_{REAL} s = \infty \text{ if } r = \infty \vee s = \infty; & (t,s) *_{\tau\times\rho} (u,r) = (t *_\tau u, s *_\rho r). \end{array}$$

This is precisely what it is needed to turn $(\!|\tau|\!)$ into a *quantale*[2] [25].

▶ **Proposition 4.** *For each type τ, $(\!|\tau|\!)$ forms a commutative unital non-idempotent quantale.*

The fact that $(\!|\tau|\!)$ is a quantale means that it has, e.g., the right structure to be the codomain of generalised metrics [19, 18]. Actually, a more general structure is needed for our purposes, namely the one of a generalised metric domain, which will be thoroughly discussed in Section 6 below. For the moment, let us concentrate our attention to programs:

▶ **Definition 5** (Differential Logical Relations). *We define a* differential logical relation $\{\delta_\tau \subseteq \Lambda_\tau \times (\!|\tau|\!) \times \Lambda_\tau\}_\tau$ *as a set of ternary relations indexed by types satisfying*

$$\begin{aligned}
\delta_{REAL}(M, r, N) &\Leftrightarrow |NF(M) - NF(N)| \leq r;\\
\delta_{\tau\times\rho}(M, (d_1, d_2), N) &\Leftrightarrow \delta_\tau(\pi_1 M, d_1, \pi_1 N) \wedge \delta_\rho(\pi_2 M, d_2, \pi_2 N)\\
\delta_{\tau\to\rho}(M, d, N) &\Leftrightarrow (\forall V \in CV(\tau).\ \forall x \in (\!|\tau|\!).\ \forall W \in CV(\tau).\\
&\qquad \delta_\tau(V, x, W) \Rightarrow \delta_\rho(MV, d([\![V]\!], x), NW) \wedge \delta_\rho(MW, d([\![V]\!], x), NV)).
\end{aligned}$$

An intuition behind the condition required for $\delta_{\tau\to\rho}(M, d, N)$ is that $d([\![V]\!], x)$ overapproximates both the "distance" between MV and NW and the one between MW and NV, this *whenever* W is within the error x from V.

3.1 A Fundamental Lemma

Usually, the main result about any system of logical relations is the so-called *Fundamental Lemma*, which states that any typable term is in relation *with itself*. But how would the Fundamental Lemma look like here? Should any term be at *somehow minimal distance* to itself, in the spirit of what happens, e.g. with metrics [24, 13]? Actually, there is no hope to prove anything like that for differential logical relations, as the following example shows.

▶ **Example 6.** Consider again the term $M_{ID} = \lambda x.x$, which can be given type $\tau = REAL \to REAL$ in the empty context. Please recall that $(\!|\tau|\!) = \mathbb{R} \times \mathbb{R}^\infty_{\geq 0} \to \mathbb{R}^\infty_{\geq 0}$. Could we prove that $\delta_\tau(M_{ID}, 0_\tau, M_{ID})$, where 0_τ is the constant-0 function? The answer is negative: given two real numbers r and s at distance ε, the terms $M_{ID}r$ and $M_{ID}s$ are themselves ε apart, thus at nonnull distance. The best one can say, then, is that $\delta_\tau(M_{ID}, f, M_{ID})$, where $f(x, \varepsilon) = \varepsilon$.

As the previous example suggests, a term M being at self-distance d is a witness of M being *sensitive to changes* to the environment according to d. Indeed, the only terms which are at self-distance 0 are the constant functions. This makes the underlying theory more general than the one of logical or metric relations, although the latter can be proved to be captured by differential logical relations, as we will see in the next section.

Coming back to the question with which we opened the section, we can formulate a suitable fundamental lemma for differential logical relations.

▶ **Theorem 7** (Fundamental Lemma, Version I). *For every $\cdot \vdash M : \tau$ there is a $d \in (\!|\tau|\!)$ such that $(M, d, M) \in \delta_\tau$.*

Proof sketch. The proof proceeds, as usual, by induction on the derivation of $\cdot \vdash M : \tau$. In order to deal with e.g. λ-abstractions we have to strengthen our statement taking into account *open* terms. This turns out to be non-trivial and requires to extend our notion of

[2] Recall that a quantale $\mathbb{Q} = (Q, \leq_Q, 0_Q, *_Q)$ consists of a complete lattice $(Q, \leq_Q)$ and a monoid $(Q, 0_Q, *_Q)$ such that the lattice and monoid structures properly interact (meaning that monoid multiplication distributes over joins). We refer to [25, 18] for details.

a differential logical relation to arbitrary terms. First of all, we need to generalise $(\!|\cdot|\!)$ and $[\![\cdot]\!]$ to environments. For instance, $(\!|\Gamma|\!)$ is the set of families in the form $\alpha = \{\alpha_x\}_{(x:\rho)\in\Gamma}$, where $\alpha_x \in (\!|\rho|\!)$. Similarly for $[\![\Gamma]\!]$. This way, a natural space for differences between terms $\Gamma \vdash M, N : \tau$ can be taken as $(\!|\tau|\!)^{[\![\Gamma]\!]\times(\!|\Gamma|\!)}$, namely the set of maps from $[\![\Gamma]\!] \times (\!|\Gamma|\!)$ to $(\!|\tau|\!)$. Given an environment Γ, a family $\mathbf{V} = \{V_x\}_{(x:\rho_x)\in\Gamma}$ such that $V_x \in CV(\rho_x)$ is said to be a Γ-*family of values*. Such a Γ-family of values can naturally be seen as a substitution $\mathbf{V}$ mapping each variable $(x : \rho) \in \Gamma$ to $V_x \in CV(\rho_x)$. As it is customary, for a term $\Gamma \vdash M : \tau$ we write $M\mathbf{V}$ for the closed term of type τ obtained applying the substitution $\mathbf{V}$ to M. We denote by $CV(\Gamma)$ the set of all Γ-family of values. Given a set Z, an environment Γ, and two Γ-indexed families $\alpha = \{\alpha_x\}_{(x:\rho)\in\Gamma}$, $\beta = \{\beta_x\}_{(x:\rho)\in\Gamma}$ over Z (meaning that e.g. $\alpha_x \in Z$, for each $(x : \rho) \in \Gamma$), we introduce the following notational convention. For a Γ-indexed family $B = \{b_x\}_{(x:\rho)\in\Gamma}$ such that $b_x \in \{0, 1\}$, we can construct a "choice" Γ-indexed family B^α_β as follows:

$$(B^\alpha_\beta)_x = \begin{cases} \alpha_x & \text{if } b_x = 0 \\ \beta_x & \text{if } b_x = 1. \end{cases}$$

Moreover, given a family B as above, we can construct the *inverse* family $\overline{B}$ as the family $\{1 - b_x\}_{(x:\rho)\in\Gamma}$. We can now talk about *open terms*, and from a differential logical relation $\{\delta_\tau \subseteq \Lambda_\tau \times (\!|\tau|\!) \times \Lambda_\tau\}_\tau$ construct a family of relations $\{\delta^\Gamma_\tau \subseteq \Lambda^\Gamma_\tau \times (\!|\tau|\!)^{[\![\Gamma]\!]\times(\!|\Gamma|\!)} \times \Lambda^\Gamma_\tau\}_{\tau,\Gamma}$ by stipulating that $\delta^\Gamma_\tau(M, d, N)$ iff

$$\delta_\Gamma(\mathbf{V}, Y, \mathbf{W}) \Longrightarrow \forall B \in \{0, 1\}^\Gamma.\delta_\tau(MB^{\mathbf{V}}_{\mathbf{W}}, d([\![\mathbf{V}]\!], Y), N\overline{B}^{\mathbf{V}}_{\mathbf{W}}).$$

We now prove the following strengthening of our main thesis: for any term $\Gamma \vdash M : \tau$, there is a $d \in (\!|\tau|\!)^{[\![\Gamma]\!]\times(\!|\Gamma|\!)}$ such that $\delta^\Gamma_\tau(M, d, M)$. At this point the proof is rather standard, and proceeds by induction on the derivation of $\Gamma \vdash M : \tau$. ◀

But what do we gain from Theorem 7? In the classic theory of logical relations, the Fundamental Lemma has, as an easy corollary, that logical relations are compatible: it suffices to invoke the theorem with any context C *seen as a term* $C[x]$, such that $x : \tau, \Gamma \vdash C[x] : \rho$. Thus, ultimately, logical relations are proved to be a *compositional* methodology for program equivalence, in the following sense: if M and N are equivalent, then $C[M]$ and $C[N]$ are equivalent, too.

In the realm of differential logical relations, the Fundamental Lemma plays a similar role, although with a different, *quantitative* flavor: once C has been proved sensitive to changes according to d, and V, W are proved to be at distance e, then, e.g., the impact of substituting V with W in C can be measured by composing d and e (and $[\![V]\!]$), i.e. by computing $d([\![V]\!], e)$. Notice that the sensitivity analysis on C and the relational analysis on V and W are decoupled. What the Fundamental Lemma tells you is that d can *always* be found.

3.2 Our Running Example, Revisited

It is now time to revisit the example we talked about in the Introduction. Consider the following two programs, both closed and of type $REAL \to REAL$:

$$M_{SIN} = \lambda x.sin_1(x); \qquad M_{ID} = \lambda x.x.$$

First of all, let us observe that, as already remarked, comparing M_{SIN} and M_{ID} using the sup metric on $\mathbb{R} \to \mathbb{R}$, as it is done in metric logical relations and applicative distances, naturally assigns them distance ∞, the euclidean distance $|x - sin(x)|$ being unbounded when x ranges over $\mathbb{R}$.

Let us now prove that $(M_{SIN}, f, M_{ID}) \in \delta_{REAL \to REAL}$, where $f(x,y) = y + |x - \sin x|$. Consider any pair of real numbers $r, s \in \mathbb{R}$ such that $|r - s| \leq \varepsilon$, where $\varepsilon \in \mathbb{R}^{\infty}_{\geq 0}$. We have that:

$$\begin{aligned} |\sin r - s| &= |\sin r - r + r - s| \leq |\sin r - r| + |r - s| \leq |\sin r - r| + \varepsilon = f(r, \varepsilon) \\ |\sin s - r| &= |\sin s - \sin r + \sin r - r| \leq |\sin s - \sin r| + |\sin r - r| \leq |s - r| + |\sin r - r| \\ &\leq \varepsilon + |\sin r - r| = f(r, \varepsilon). \end{aligned}$$

The fact that $|\sin s - \sin r| \leq |s - r|$ is a consequence of sin being 1-Lipschitz continuous (see, e.g., [12] for a simple proof).

Now, consider a context C which makes use of either M_{SIN} or M_{ID} by feeding them with a value close to 0, call it θ. Such a context could be, e.g., $C = (\lambda x.x(x\theta))[\cdot]$. C can be seen as a term having type $\tau = (REAL \to REAL) \to REAL$. A self-distance d for C can thus be defined as an element of

$$(\!|\tau|\!) = [\![REAL \to REAL]\!] \times (\!|REAL \to REAL|\!) \to \mathbb{R}^{\infty}_{\geq 0}.$$

namely $F = \lambda\!\!\lambda \langle g, h \rangle . h(g(\theta), h(\theta, 0))$. This allows for compositional reasoning about program distances: the overall impact of replacing M_{SIN} by M_{ID} can be evaluated by computing $F([\![M_{SIN}]\!], f)$. Of course the context C needs to be taken into account, but *once and for all*: the functional F can be built without knowing with which term(s) it will be fed with.

4 Logical and Metric Relations as DLRs

The previous section should have convinced the reader about the peculiar characteristics of differential logical relations compared to (standard) metric and logical relations. In this section we show that despite the apparent differences, logical and metric relations can somehow be retrieved as specific kinds of program differences. This is, however, bound to be nontrivial. The naïve attempt, namely seeing program equivalence as being captured by *minimal* distances in logical relations, fails: the distance between a program *and itself* can be nonnull.

How should we proceed, then? Isolating those distances which witness program equivalence is indeed possible, but requires a bit of an effort. In particular, the sets of those distances can be, again, defined by induction on τ. For every τ, we give $(\!|\tau|\!)^0 \subseteq (\!|\tau|\!)$ by induction on the structure of τ:

$$(\!|REAL|\!)^0 = \{0\}; \qquad (\!|\tau \times \rho|\!)^0 = (\!|\tau|\!)^0 \times (\!|\rho|\!)^0;$$

$$(\!|\tau \to \rho|\!)^0 = \{f \in (\!|\tau \to \rho|\!) \mid \forall x \in [\![\tau]\!] . \forall y \in (\!|\tau|\!)^0 . f(x,y) \in (\!|\rho|\!)^0\}.$$

Notice that $(\!|\tau \to \rho|\!)^0$ is not defined as $[\![\tau]\!] \times (\!|\tau|\!)^0 \to (\!|\rho|\!)^0$ (doing so would violate $(\!|\tau \to \rho|\!)^0 \subseteq (\!|\tau \to \rho|\!)$). The following requires some effort, and testifies that, indeed, program equivalence in the sense of logical relations precisely corresponds to being at a distance in $(\!|\tau|\!)^0$:

▶ **Theorem 8.** *Let $\{\mathcal{L}_\tau\}_\tau$ be a logical relation. There exists a differential logical relation $\{\delta_\tau\}_\tau$ satisfying $\mathcal{L}_\tau(M,N) \iff \exists d \in (\!|\tau|\!)^0 . \delta_\tau(M, d, N)$.*

What if we want to generalise the argument above to metric relations, as introduced, e.g., by Reed and Pierce [24]? The set $(\!|\tau|\!)^0$ becomes a set of distances parametrised by a single real number:

$$(\!|REAL|\!)^r = \{r\}; \qquad (\!|\tau \times \rho|\!)^r = (\!|\tau|\!)^r \times (\!|\rho|\!)^r;$$

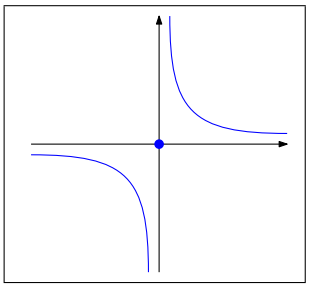


Figure 4 A total, but highly discontinuous, function.

$$(\!|\tau \to \rho|\!)^r = \{f \in (\!|\tau \to \rho|\!) \mid \forall x \in [\![\tau]\!].\forall y \in (\!|\tau|\!)^s.f(x,y) \in (\!|\rho|\!)^{r+s}\}.$$

A result similar to Theorem 8 is unfortunately outside the scope of this paper, but can be found in the Extended Version [12]. In particular, metric relations are only available in calculi, like FUZZ [24], which rely on *linear* type systems, thus more refined than the one we endow $ST^{\lambda}_{\mathbb{R}}$ with.

5 Strengthening the Fundamental Theorem through Finite Distances

Let us now ask ourselves the following question: given any term $M \in CT(\tau)$, what can we say about its sensitivity, i.e., about the values $d \in (\!|\tau|\!)$ such that $\delta_\tau(M,d,M)$? Two of the results we have proved about $ST^{\lambda}_{\mathbb{R}}$ indeed give partial answers to the aforementioned question. On the one hand, Theorem 7 states that such a d can *always* be found. On the other hand, Theorem 8 tells us that such a d can be taken in $(\!|\tau|\!)^0$. Both these answers are not particularly informative, however. The mere existence of such a $d \in (\!|\tau|\!)$, for example, is trivial since d can always be taken as d_∞, the maximal element of the underlying quantale. The fact that such a d can be taken from $(\!|\tau|\!)^0$ tells us that, e.g. when $\tau = \rho \to \xi$, M returns equivalent terms when fed with equivalent arguments: there is no quantitative guarantee about the behaviour of the term when fed with non-equivalent arguments.

Is this the best one can get about the sensitivity of $ST^{\lambda}_{\mathbb{R}}$ terms? The absence of full recursion suggests that we could hope to prove that infinite distances, although part of the underlying quantale, can in fact be useless. In other words, we are implicitly suggesting that self-distances could be elements of $(\!|\tau|\!)^{<\infty} \subset (\!|\tau|\!)$, defined as follows:

$$(\!|REAL|\!)^{<\infty} = \mathbb{R}_{\geq 0}; \qquad (\!|\tau \times \rho|\!)^{<\infty} = (\!|\tau|\!)^{<\infty} \times (\!|\rho|\!)^{<\infty};$$

$$(\!|\tau \to \rho|\!)^{<\infty} = \{f \in (\!|\tau \to \rho|\!) \mid \forall x \in [\![\tau]\!].\forall t \in (\!|\tau|\!)^{<\infty}.f(x,t) \in (\!|\rho|\!)^{<\infty}\}.$$

Please observe that $(\!|\tau|\!)^{<\infty}$ is in general a much larger set of differences than $\bigcup_{r \in \mathbb{R}^{\infty}_{\geq 0}} (\!|\tau|\!)^r$: the former equals the latter only when τ is $REAL$. Already when τ is $REAL \to REAL$, the former includes, say, functions like $f(r,\varepsilon) = (r+\varepsilon)^2$, while the latter does not.

Unfortunately, there are terms in $ST^{\lambda}_{\mathbb{R}}$ which cannot be proved to be at self-distance in $(\!|\tau|\!)^{<\infty}$, and, surprisingly, this is *not* due to the higher-order features of $ST^{\lambda}_{\mathbb{R}}$, but to $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$ being arbitrary, and containing functions which do not map finite distances to finite distances, like

$$h(r) = \begin{cases} 0 & \text{if } r = 0 \\ \frac{1}{r} & \text{otherwise} \end{cases}$$

(see Figure 4). Is this phenomenon *solely* responsible for the necessity of finite self-distances in $ST^{\lambda}_{\mathbb{R}}$? The answer is positive, and the rest of this section is devoted precisely to formalising and proving the aforementioned conjecture.

First of all, we need to appropriately axiomatise the absence of unbounded discontinuities from $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$. A not-so-restrictive but sufficient axiom turns out to be weak boundedness: a function $f_n : \mathbb{R}^n \to \mathbb{R}$ is said to be *weakly bounded* if and only if it maps bounded subsets of $\mathbb{R}^n$ into bounded subsets of $\mathbb{R}$. As an example, the function h above is *not* weakly bounded, because $h([-\varepsilon, \varepsilon])$ is

$$\left(-\infty, -\frac{1}{\varepsilon}\right] \cup \{0\} \cup \left[\frac{1}{\varepsilon}, \infty\right)$$

which is unbounded for any as $\varepsilon > 0$. Any term M is said to be weakly bounded iff any function symbol f_n occurring in M is itself weakly bounded. Actually, this is precisely what one needs to get the strengthening of the Fundamental Theorem we are looking for.

▶ **Theorem 9** (Fundamental Theorem, Version II). *For any weakly bounded term $\cdot \vdash M : \tau$, there is $d \in (\!|\tau|\!)^{<\infty}$ such that $(M, d, M) \in \delta_\tau$.*

The reader may have wondered about how restrictive a condition weak boundedness really is. In particular, whether it corresponds to some form of continuity. In fact, the introduced condition only rules out unbounded discontinuities. In other words, weak boundedness can be equivalently defined by imposing local boundedness *at any point* in the domain $\mathbb{R}$. This is weaker than asking for boundedness, which requires the existence of a global bound.

6 A Categorical Perspective

Up to now, differential logical relations have been treated very concretely, without looking at them through the lens of category theory. This is in contrast to, e.g., the treatment of metric relations from [13], in which soundness of metric relations for FUZZ is obtained as a byproduct of a proof of symmetric monoidal closedness for the category **MET** of pseudometric spaces and Lipschitz functions.

But what could take the place of pseudometric spaces in a categorical framework capturing differential logical relations? The notion of a metric needs to be relaxed along at least two axes. On the one hand, the codomain of the "metric" δ is not necessarily the set of real numbers, but a more general structure, namely a quantale. On the other, as we already noticed, it is not necessarily true that equality implies indistancy, but rather than indistancy implies inequality. What comes out of these observations is, quite naturally, the notion of a generalized metric domain, itself a generalisation of partial metrics [7]. The rest of this section is devoted to proving that the category of generalised metric domains is indeed cartesian closed, thus forming a model of simply typed λ-calculi.

Formally, given a quantale $\mathbb{Q} = (Q, \leq_Q, 0_Q, *_Q)$[3], a *generalised metric domain* on $\mathbb{Q}$ is a pair (A, δ_A), where A is a set and δ_A is a subset of $A \times \mathbb{Q} \times A$ satisfying some axioms akin to those of a metric domain:

$$\begin{aligned}
\delta_A(x, 0_Q, y) &\Rightarrow x = y; && \text{(Indistancy Implies Equality)}\\
\delta_A(x, d, y) &\Rightarrow \delta_A(y, d, x); && \text{(Symmetry)}\\
\delta_A(x, d, y) \wedge \delta_A(y, e, y) \wedge \delta_A(y, f, z) &\Rightarrow \delta_A(x, d * e * f, z). && \text{(Triangularity)}
\end{aligned}$$

[3] When unambiguous, we will omit subscripts in $\leq_Q$, 0_Q, and $*_Q$.

Please observe that δ_A is a *relation* rather than a function. Moreover, the first axiom is dual to the one typically found in, say, pseudometrics. The third axiom, instead, resembles the usual triangle inequality for pseudometrics, but with the crucial difference that since objects can have non-null self-distance, such a distance has to be taken into account. Requiring equality to imply indistancy (and thus $\delta_A(x, 0_Q, y) \Leftrightarrow x = y$), we see that (Triangularity) gives exactly the usual triangle inequality (properly generalised to quantale and relations [18, 19]).

In this section we show that generalised metric domains form a cartesian closed category, unlike that of metric spaces (which is known to be non-cartesian closed). As a consequence, we obtain a firm categorical basis of differential logical relations. The category of generalised metric domain, denoted by **GMD**, is defined as follows.

▶ **Definition 10.** *The category* **GMD** *has the following data.*

- *An object $\mathcal{A}$ is a triple $(A, \mathbb{Q}, \delta)$ where $\mathbb{Q}$ is a quantale and (A, δ) is a generalized metric domain on $\mathbb{Q}$.*
- *An arrow $(A, \mathbb{Q}, \delta) \to (B, \mathbb{S}, \rho)$ is a pair (f, ζ) consisting of a function $f \colon A \to B$ and another function $\zeta \colon Q \times A \to S$ satisfying $\forall a, a' \in A. \forall q \in Q. \delta(a, q, a') \Rightarrow \rho(f(a), \zeta(q, a), f(a'))$ and $\rho(f(a), \zeta(q, a'), f(a'))$.*

We can indeed give **GMD** the structure of a category. In fact, the identity on the object $\mathcal{A} = (A, \mathbb{Q}, \delta)$ in **GMD** is given by $(\mathrm{id}_{\mathcal{A}}, \mathrm{id}'_{\mathcal{A}})$ where $\mathrm{id}_{\mathcal{A}} \colon A \to A$ is the set-theoretic identity on A and $\mathrm{id}'_{\mathcal{A}} \colon Q \times A \to Q$ is defined by $\mathrm{id}'_{\mathcal{A}}(q, a) = q$. The composition of two arrows $(f, \zeta) \colon (A, \mathbb{Q}, \delta) \to (B, \mathbb{S}, \rho)$ and $(g, \eta) \colon (B, \mathbb{S}, \rho) \to (C, \mathbb{T}, \nu)$ is the pair (h, θ) where $h \colon A \to C$ is given by the function composition $g \circ f \colon A \to C$ and $h \colon Q \times A \to T$ is given by $\theta(q, a) = \eta(\zeta(q, a), f(a))$. Straightforward calculations show that composition is associative, and that the identity arrow behaves as its neutral element.

Most importantly, we can give **GMD** a cartesian closed structure, as shown by the following result[4].

▶ **Theorem 11.** **GMD** *is a cartesian closed category.*

Proof sketch. Before entering details, it is useful to remark that the cartesian product of two quantales is itself a quantale (with lattice and monoid structure defined pointwise). Similarly, for any quantale $\mathbb{Q}$ and set X, the function space $\mathbb{Q}^X$ inherits a quantale structure from $\mathbb{Q}$ pointwise. Let us now show that **GMD** is cartesian closed. We begin showing that **GMD** has a terminal object and binary products. The former is defined as $(\{*\}, \mathbb{O}, \delta_0)$, where $\mathbb{O}$ is the one-element quantale $\{0\}$, and $\delta_0 = \{(*, 0, *)\}$ (notice that $(\{*\}, \delta_0)$ is a generalized metric domain on $\mathbb{O}$), whereas the binary product $\mathcal{A} \times \mathcal{B}$ of two objects $\mathcal{A}$ and $\mathcal{B}$ in **GMD** is given by a triple $(A \times B, \mathbb{Q} \times \mathbb{S}, \delta \times \rho)$. Finally, we define exponentials in **GMD**. Given $\mathcal{C}$, $\mathcal{B}$ in **GMD**, their exponential $\mathcal{C}^{\mathcal{B}}$ is the triple $(C^B, \mathbb{T}^{\mathbb{S} \times B}, \nu^{\rho})$, where C^B is the function space $\{f \mid f \colon B \to C\}$, $\mathbb{T}^{\mathbb{S} \times B}$ is the exponential quantale, and ν^{ρ} is a ternary relation over $C^B \times T^{S \times B} \times C^B$ defined by: if $\rho(b, s, b')$ then $\nu(f(b), d(s, b), f'(b'))$ and $\nu(f(b), d(s, b'), \zeta(b'))$. Please notice that the relation ν^{ρ} is indeed a differential logical relation. ◀

Interestingly, the constructions of product and exponential objects in the proof of Theorem 11 closely match the definition of a differential logical relation. In other words, differential logical relations as given in Definition 5 can be seen as providing a denotational model of $ST^{\lambda}_{\mathbb{R}}$ in which base types are interpreted by the generalised metric domain corresponding to the Euclidean distance.

[4] See [12] for a detailed proof.

7 Conclusion

In this paper, we introduced differential logical relations as a novel methodology to evaluate the "distance" between programs of higher-order calculi akin to the λ-calculus. We have been strongly inspired by some unpublished work by Westbrook and Chaudhuri [31], who were the first to realise that evaluating differences between interactive programs requires going beyond mere real numbers. We indeed borrowed our running example from the aforementioned work.

This paper's contribution, then consists in giving a simple definition of differential logical relations, together with some results about their underlying metatheory: two formulations of the Fundamental Lemma, a result relating differential logical relations and ordinary logical relations, and a proof that generalised metric domains – the metric structure corresponding to differential logical relations – form a cartesian closed category. Such results give evidence that, besides being *more expressive* than metric relations, differential logical relations are somehow *more canonical*, naturally forming a model of simply-typed λ-calculi.

As the title of this paper suggests, we see the contributions above just as a very first step towards understanding the nature of differences in a logical environment. In particular, at least two directions deserve to be further explored.

- The first one concerns *language features*: admittedly, the calculus $ST^{\lambda}_{\mathbb{R}}$ we consider here is very poor in terms of its expressive power, lacking full higher-order recursion and thus not being universal. Moreover, $ST^{\lambda}_{\mathbb{R}}$ does not feature any form of effect, including probabilistic choices, in which evaluating differences between programs would be very helpful. Addressing such issues seems to require to impose a domain structure on generalised metric domains, on one hand, and to look at monads on **GMD**, on the other hand (for the latter, the literature on monadic lifting for quantale-valued relations might serve as a guide [18]).
- The second one is about *abstract differences*: defining differences as functions with *the same rank* as that of the compared programs implies that reasoning about them is complex. Abstracting differences so as to facilitate differential reasoning could be the way out, given that deep connections exist between logical relations and abstract interpretation [2]. Another way to understand program difference better is to investigate whether differential logical relations can be related to abstract structures for differentiation, as in [3]. Indeed, Example 6 suggests that an interesting distance between a program and itself can be taken as its derivative, the latter being defined as in [8].

References

1 S. Abramsky. The Lazy Lambda Calculus. In D. Turner, editor, *Research Topics in Functional Programming*, pages 65–117. Addison Wesley, 1990.

2 Samson Abramsky. Abstract Interpretation, Logical Relations and Kan Extensions. *J. Log. Comput.*, 1(1):5–40, 1990.

3 Mario Alvarez-Picallo and C.-H. Luke Ong. Change Actions: Models of Generalised Differentiation. In *Proc. of FOSSACS 2019*, pages 45–61, 2019.

4 A. Arnold and M. Nivat. Metric Interpretations of Infinite Trees and Semantics of non Deterministic Recursive Programs. *Theor. Comput. Sci.*, 11:181–205, 1980.

5 C. Baier and M.E. Majster-Cederbaum. Denotational Semantics in the CPO and Metric Approach. *Theor. Comput. Sci.*, 135(2):171–220, 1994.

6 Gilles Barthe, Marco Gaboardi, Justin Hsu, and Benjamin C. Pierce. Programming language techniques for differential privacy. *SIGLOG News*, 3(1):34–53, 2016.

7 Michael A. Bukatin, Ralph Kopperman, Steve Matthews, and Homeira Pajoohesh. Partial Metric Spaces. *The American Mathematical Monthly*, 116(8):708–718, 2009.

8 Yufei Cai, Paolo G. Giarrusso, Tillmann Rendel, and Klaus Ostermann. A theory of changes for higher-order languages: incrementalizing λ-calculi by static differentiation. In *Proc. of PLDI*, pages 145–155, 2014.

9 Konstantinos Chatzikokolakis, Daniel Gebler, Catuscia Palamidessi, and Lili Xu. Generalized Bisimulation Metrics. In *CONCUR 2014 - Concurrency Theory - 25th International Conference, CONCUR 2014, Rome, Italy, September 2-5, 2014. Proceedings*, pages 32–46, 2014.

10 Raphaëlle Crubillé and Ugo Dal Lago. Metric Reasoning about λ-Terms: The Affine Case. In *Proc. of LICS 2015*, pages 633–644, 2015.

11 Raphaëlle Crubillé and Ugo Dal Lago. Metric Reasoning About λ-Terms: The General Case. In *Proc. of ESOP 2017*, pages 341–367, 2017.

12 Ugo Dal Lago, Francesco Gavazzo, and Akira Yoshimizu. Differential Logical Relations, Part I: The Simply-Typed Case (Extended Version), 2018. `arXiv:1904.12137`.

13 A.A. de Amorim, M. Gaboardi, J. Hsu, S. Katsumata, and I. Cherigui. A semantic account of metric preservation. In *Proc. of POPL 2017*, pages 545–556, 2017.

14 J.W. de Bakker and J.I. Zucker. Denotational Semantics of Concurrency. In *STOC*, pages 153–158, 1982.

15 Josee Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. Metrics for labelled Markov processes. *Theor. Comput. Sci.*, 318(3):323–354, 2004.

16 M.H. Escardo. A metric model of PCF. In *Workshop on Realizability Semantics and Applications*, 1999.

17 Francesco Gavazzo. Quantitative Behavioural Reasoning for Higher-order Effectful Programs: Applicative Distances. In *Proc. of LICS 2018*, pages 452–461, 2018.

18 D. Hofmann, G.J. Seal, and W. Tholen, editors. *Monoidal Topology. A Categorical Approach to Order, Metric, and Topology.* Number 153 in Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2014.

19 F.W. Lawvere. Metric spaces, generalized logic, and closed categories. *Rend. Sem. Mat. Fis. Milano*, 43:135–166, 1973.

20 John C. Mitchell. *Foundations for Programming Languages.* MIT Press, 1996.

21 Sparsh Mittal. A Survey of Techniques for Approximate Computing. *ACM Comput. Surv.*, 48(4), 2016.

22 J. Morris. *Lambda Calculus Models of Programming Languages.* PhD thesis, MIT, 1969.

23 Gordon D. Plotkin. Lambda-Definability and Logical Relations. Memorandum SAI-RM-4, University of Edinburgh, 1973.

24 J. Reed and B.C. Pierce. Distance makes the types grow stronger: a calculus for differential privacy. In *Proc. of ICFP 2010*, pages 157–168, 2010.

25 K.I. Rosenthal. *Quantales and their applications.* Pitman research notes in mathematics series. Longman Scientific & Technical, 1990.

26 Dana Scott. Outline of a mathematical theory of computation. Technical Report PRG02, OUCL, November 1970.

27 Dana Scott and Christopher Strachey. Toward a mathematical semantics for computer languages. Technical Report PRG06, OUCL, August 1971.

28 F. Van Breugel. An introduction to metric semantics: operational and denotational models for programming and specification languages. *Theor. Comput. Sci.*, 258(1-2):1–98, 2001.

29 F. Van Breugel and J. Worrell. A behavioural pseudometric for probabilistic transition systems. *Theor. Comput. Sci.*, 331(1):115–142, 2005.

30 Franck van Breugel and James Worrell. Towards Quantitative Verification of Probabilistic Transition Systems. In *Proc. of ICALP 2001*, pages 421–432, 2001.

31 Edwin M. Westbrook and Swarat Chaudhuri. A Semantics for Approximate Program Transformations. *CoRR*, abs/1304.5531, 2013. `arXiv:1304.5531`.

32 Lili Xu, Konstantinos Chatzikokolakis, and Huimin Lin. Metrics for Differential Privacy in Concurrent Systems. In *Proc. of FORTE 2014*, pages 199–215, 2014.

Approximations of Isomorphism and Logics with Linear-Algebraic Operators

Anuj Dawar
University of Cambridge, UK
anuj.dawar@cl.cam.ac.uk

Erich Grädel
RWTH Aachen University, Germany
graedel@logic.rwth-aachen.de

Wied Pakusa
RWTH Aachen University, Germany
pakusa@logic.rwth-aachen.de

Abstract

Invertible map equivalences are approximations of graph isomorphism that refine the well-known Weisfeiler-Leman method. They are parameterized by a number k and a set Q of primes. The intuition is that two equivalent graphs $G \equiv^{\mathrm{IM}}_{k,Q} H$ cannot be distinguished by means of partitioning the set of k-tuples in both graphs with respect to *any* linear-algebraic operator acting on vector spaces over fields of characteristic p, for any $p \in Q$. These equivalences have first appeared in the study of rank logic, but in fact they can be used to delimit the expressive power of any extension of fixed-point logic with linear-algebraic operators. We define $\mathrm{LA}^k(Q)$, an infinitary logic with k variables and all linear-algebraic operators over finite vector spaces of characteristic $p \in Q$ and show that $\equiv^{\mathrm{IM}}_{k,Q}$ is the natural notion of elementary equivalence for this logic. The logic $\mathrm{LA}^\omega(Q) = \bigcup_{k\in\omega} \mathrm{LA}^k(Q)$ is then a natural upper bound on the expressive power of any extension of fixed-point logics by means of Q-linear-algebraic operators.

By means of a new and much deeper algebraic analysis of a generalized variant, for any prime p, of the CFI-structures due to Cai, Fürer, and Immerman, we prove that, as long as Q is not the set of *all* primes, there is no k such that $\equiv^{\mathrm{IM}}_{k,Q}$ is the same as isomorphism. It follows that there are polynomial-time properties of graphs which are not definable in $\mathrm{LA}^\omega(Q)$, which implies that no extension of fixed-point logic with linear-algebraic operators can capture PTIME, unless it includes such operators for all prime characteristics. Our analysis requires substantial algebraic machinery, including a homogeneity property of CFI-structures and Maschke's Theorem, an important result from the representation theory of finite groups.

2012 ACM Subject Classification Theory of computation → Finite Model Theory

Keywords and phrases Finite Model Theory, Graph Isomorphism, Descriptive Complexity, Algebra

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.112

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version A full version of this paper is available at `https://arxiv.org/abs/1902.06648`.

1 Introduction

The graph isomorphism problem (or more generally, the structure isomorphism problem) is an important computational problem which is also very interesting from the point of view of complexity theory. It is not known to be in P nor known to be NP-complete. It is known to be solvable in quasi-polynomial time by Babai's algorithm [3].

An important theoretical approach to understanding the nature of the graph isomorphism problem is the Weisfeiler-Leman method. For each positive integer k, the k-dimensional Weisfeiler-Leman method (k-WL method for short) defines an equivalence relation $\equiv^k$ which

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 112; pp. 112:1–112:14

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

over-approximates isomorphism in the sense that if $G \cong H$ for a pair of graphs G and H, then $G \equiv^k H$ for any k. The relations form a refining family in the sense that if $G \not\equiv^k H$ then $G \not\equiv^{k'} H$ for all $k' > k$. Thus, the equivalence relation gets finer with increasing k and approaches isomorphism in the limit. Moreover, if G and H are n-vertex graphs then $G \equiv^n H$ if, and only if, $G \cong H$. For each fixed k, the equivalence relation $\equiv^k$ is decidable in polynomial time, indeed in time $n^{O(k)}$. Thus, if there were a fixed k such that $\equiv^k$ were the same as isomorphism, we would have a polynomial-time algorithm for graph isomorphism. However, we know this is not the case. Cai, Fürer and Immerman [6] showed that there are pairs of non-isomorphic graphs G and H with $O(k)$ vertices such that $G \equiv^k H$. We call the construction of such graphs the CFI construction.

The Weisfeiler-Leman equivalences arise naturally in the study of graphs in many different guises. We have definitions based on combinatorics (such as Babai's original definition, see [6]); in logic as the equivalences induced by bounded variable fragments of first-order logic with counting; linear programming (see [2, 21]); and algebra (as in the original definition of Weisfeiler and Leman, extended to dimension k in [13]). The equivalences have proved to be of central importance in the area of descriptive complexity theory. In particular, they delimit the power of fixed-point logic with counting (FPC), an important logic in the study of symmetric polynomial-time computation (see [9]). On many important classes of structures, it turns out that there is a fixed k for which k-WL suffices to distinguish all non-isomorphic graphs. Most significantly, Grohe [20] has shown that for any proper minor-closed class $\mathcal{C}$ of graphs, there is a k such that $\equiv^k$ coincides with isomorphism on graphs in $\mathcal{C}$.

Despite its importance in the interplay of graph structure theory and logic, and its theoretical significance in understanding the graph isomorphism problem, the Weisfeiler-Leman method does not give the most efficient algorithms for solving the isomorphism problem. The CFI construction demonstrates that using the WL method to decide isomorphism would yield an algorithm of complexity $n^{\Omega(n)}$ which is asymptotically no better than trying all permutations and far removed from the quasi-polynomial time algorithms known. This has inspired the search for other structured families of equivalences (see for example [4, 16]). One particularly interesting such family are the *invertible-map equivalences* defined in [14]. This gives, for each k and each set Q of prime numbers, an equivalence relation $\equiv^{\mathrm{IM}}_{k,Q}$. The precise definition is given in Section 2 but the intuition is that if $G \equiv^{\mathrm{IM}}_{k,Q} H$, then G and H are not distinguishable by a refinement of k-tuples given by linear operators acting on vector spaces over fields of characteristic p, for any $p \in Q$. The reason for considering such equivalences stems from the realisation that the CFI-construction codes in graph form the problem of solving equations over $\mathbb{F}_2$ – the 2-element field (see [1]). It can then be shown that the family of equivalences $\equiv^{\mathrm{IM}}_{k,\{2\}}$ properly refine the Weisfeiler-Leman equivalences in that $G \equiv^{\mathrm{IM}}_{k',\{2\}} H$ for sufficiently large k' implies $G \equiv^k H$ for all k, yet $G \not\equiv^{\mathrm{IM}}_{3,\{2\}} H$ for the pairs G, H obtained in the CFI construction.

Furthermore, for any finite Q, the relation $\equiv^{\mathrm{IM}}_{k,Q}$ is decidable in time $n^{O(k)}$. We can also vary Q with n. For instance, we could let Q_s be the collection of all primes up to $s(n)$ for some growing function s. In this case $\equiv^{\mathrm{IM}}_{k,Q_s}$ is decidable in time $s(n)n^{O(k)}$. It is therefore an interesting question whether the family of equivalence relations is (like the Weisfeiler-Leman equivalences) infinitely refining. Do increasing values of k yield ever finer equivalence relations? The rôle of the parameter Q is also worth investigating. If there were a fixed polynomial s and constant k for which $\equiv^{\mathrm{IM}}_{k,Q_s}$ was the same as isomorphism, we would have a polynomial-time test for isomorphism. Even if we could prove this for k growing poly-logarithmically, and s quasi-polynomial, this would yield a new (and more systematic) quasi-polynomial algorithm for isomorphism. We have no reason to conjecture that either of these upper bounds holds, but they have not been ruled out.

One reason for the interest in the invertible-map equivalences is the connection with logic. In the long-running quest for a logic for PTIME (see [19]), an important direction is the study of extensions of fixed-point logic with rank operators (FPR) [12] or other algebraic operators (see [10]). The relations $\equiv^{\mathrm{IM}}_{k,Q}$ were introduced first as a tool to study the expressive power of FPR. It was shown in [14] that for every formula φ of FPR (as originally defined in [12]) there is a k and a finite Q such that the class of models of φ is closed under $\equiv^{\mathrm{IM}}_{k,Q}$. For the more powerful rank logic FPR^* defined in [18], we can show that for any formula φ, there is a k and a polynomial s such that φ is invariant under $\equiv^{\mathrm{IM}}_{k,Q_s}$. This implies, in particular that, if we could show that there is no fixed k such that $\equiv^{\mathrm{IM}}_{k,Q}$ is the same as isomorphism when Q is the set of all primes, we could, by means of padding and results from [14], separate FPR^* from PTIME. In short, any advance in understanding the structure of these equivalence relations is a significant step for resolving important questions.

The equivalence relations tell us about more than just rank logic. They can be used to delimit the expressive power of any extension of fixed-point logic with linear-algebraic operators. In this paper we introduce $\mathrm{LA}^k(Q)$, an infinitary logic with k variables and all linear-algebraic operators (which we define formally below) over finite vector spaces of characteristic $p \in Q$. This is the logic for which $\equiv^{\mathrm{IM}}_{k,Q}$ is the natural notion of elementary equivalence. Then, $\mathrm{LA}^\omega(Q) = \bigcup_{k\in\omega} \mathrm{LA}^k(Q)$ is a natural upper bound on the expressive power of any extension of fixed-point logics by means of Q-linear-algebraic operators.

Our main results can now be stated as follows. As long as Q is not the set of all primes, there is no k such that $\equiv^{\mathrm{IM}}_{k,Q}$ is the same as isomorphism. From this, it follows that there are classes of graphs which are not definable in $\mathrm{LA}^\omega(Q)$. Moreover, we can construct polynomial-time decidable such classes. This implies that any logic with linear-algebraic operators, unless it includes such operators for all prime characteristics, does not capture PTIME. Note, this does not separate FPR^* from PTIME, due to the restriction on Q, but it shows that if FPR^* is to capture PTIME, we need to use the set of all primes.

Establishing the result requires significant technical innovation. In particular, we develop novel algebraic machinery that has not previously been deployed in the field of finite model theory. As noted above, the CFI construction codes, in graph form, the problem of solving systems of linear equations over $\mathbb{F}_2$. We can give a similar construction that codes linear equations over $\mathbb{F}_p$ for any prime p. Such a construction was given in [22], where it was used to establish that the resulting non-isomorphic graphs were not distinguished by a variant of $\equiv^{\mathrm{IM}}_{k,\{q\}}$ for any $q \neq p$, where the matrix operations are restricted to a particularly simple form. A more refined analysis of the construction was used in [18] to separate the expressive power of FPR from that of FPR^*. To be precise, they showed that the formulas of FPR that do not use an operator with the prime p are no more expressive than formulas of FPC over these graphs. Our result uses the same graph construction but brings significant new algebraic machinery to its analysis.

We are able to show, in this paper, that, on graphs obtained by the CFI construction for $\mathbb{F}_p$, the distinguishing power of $\equiv^{\mathrm{IM}}_{k,Q}$, where $p \notin Q$, is no greater than $\equiv^{k'}$ for some fixed k'. Note that the graphs are definitely distinguished in $\equiv^{\mathrm{IM}}_{k,Q}$ when $p \in Q$. We establish the result by showing that on these graphs, the equivalence relation $\equiv^{\mathrm{IM}}_{k,\{q\}}$ is itself definable in FPC when $q \neq p$. This is done by implementing a matrix similarity test in FPC, based on the module isomorphism algorithm of Chistov et al. [8]. There are two key ingredients by which this yields an FPC definition. The first is that, on the graphs obtained in the construction, the equivalence relation $\equiv^k$ (now understood as an equivalence relation on k-tuples of vertices rather than on graphs) coincides with the partition into automorphism orbits, for sufficiently large but constantly bounded k. We say that the graphs are C^k-homogeneous for large

enough k. The second ingredient is that, because the automorphism groups of the graphs are Abelian p-groups, this partition induces a matrix algebra over $\mathbb{F}_q$, when $q \neq p$, which is semisimple and so admits a nice decomposition by Maschke's theorem. Maschke's theorem is a central result in the representation theory of finite groups, which states conditions under which a linear-algebraic representation of a finite group admits a decomposition into irreducible representations. It is a powerful tool and we hope that its use opens the door to further applications of representation theory in the context of finite model theory. Indeed, we see a major contribution of the present work as being the introduction of Maschke's theorem and related tools into the subject.

Much technical detail is omitted due to space reasons. Full proofs, and more detailed background on the algebra we use, can be found in the full version of the paper that is available [11].

2 The Invertible Map Equivalence and Linear-Algebraic Logics

The invertible map equivalence relation was introduced in [14, 22] as a family of approximations of isomorphism. It was shown that it is at least as fine an approximation as that induced by the infinitary logic with rank quantifiers, introduced in [12]. Dawar and Holm posed the question whether there is a logic which corresponds to the invertible map equivalences. Here we answer the question by showing that these equivalence relations are the right notions of elementary equivalence for an infinitary logic extended with *all* linear algebraic operations.

We begin by defining the equivalence relations $\equiv^{\mathrm{IM}}_{k,Q}$ for $k \in \mathbb{N}$ and Q a set of prime numbers. It is worth reviewing the definition of the counting-logic equivalence $\equiv^k$ first. This is not only an equivalence relation among finite structures, it also induces an equivalence on the set of A^k (the set k-tuples over A) inside (any) structure $\mathfrak{A}$ that yields an approximation to the partition of A^k into orbits.

On a structure $\mathfrak{A}$, the relation $\equiv^k$ can be obtained by an iterative refinement process. Suppose we are given a partition $\mathcal{P} = \{P_i\}_{i\in I}$ of A^k indexed by a set I. Now, we say that a pair of tuples $\bar{a}_1$ and $\bar{a}_2$ are *$\mathcal{P}$-similar* if they are in the same part of $\mathcal{P}$ *and* for each $i \in I$ and each $j \in [k]$ the sets $\{b \in A \mid \bar{a}_1[b/j] \in P_i\}$ and $\{b \in A \mid \bar{a}_2[b/j] \in P_i\}$ have the same number of elements. The equivalence relation $\equiv^k$ can then be characterised as the coarsest partition $\mathcal{P}$ of A^k that refines the partition into atomic types, such that any two tuples in the same part of $\mathcal{P}$ are $\mathcal{P}$-similar. This means that we can arrive at this partition by starting with the partition of A^k into atomic types and repeatedly refine it until we get a partition $\mathcal{P}$ for which the notions of $\mathcal{P}$-equivalence and $\mathcal{P}$-similarity are the same.

We now modify this in two ways to obtain the definition of $\equiv^{\mathrm{IM}}_{k,Q}$. First we define similarity not in terms of the substitution of a single element b into a tuple $\bar{a} \in A^k$ but of an ℓ-tuple $\bar{b} \in A^\ell$ for some $\ell < k$. So, for each injective function $\gamma : [\ell] \to [k]$, let $\bar{a}[\bar{b}/\gamma]$ denote the tuple in A^k obtained from $\bar{a}$ by simultaneously substituting b_i in position $\gamma(i)$ for all $i \in [\ell]$. If Γ denotes the set of all injective functions from $[\ell]$ to $[k]$, we say tuples $\bar{a}_1$ and $\bar{a}_2$ are *$\mathcal{P}$-similar* if they are in the same part of $\mathcal{P}$ *and* for each $\gamma \in \Gamma$ and each $i \in I$, the sets $\{\bar{b} \in A^\ell \mid \bar{a}_1[\bar{b}/\gamma] \in P_i\}$ and $\{\bar{b} \in A^\ell \mid \bar{a}_2[\bar{b}/\gamma] \in P_i\}$ have the same size. Taking the coarsest relation that is stable in this sense still gives us $\equiv^k$ (though see [15] for some nuances when comparing with the Weisfeiler-Leman equivalences).

For our purposes, we want a different notion of similarity. Assume that $\ell = 2m$ for some m. We can view any set $C \subseteq A^\ell$ as giving us an $A^m \times A^m$ 0-1 matrix, which we denote M. So the entry in row $\bar{b}_1 \in A^m$ and column $\bar{b}_2 \in A^m$ of M is 1 if, and only if, the ℓ-tuple $\bar{b}_1\bar{b}_2$ is in C. Hence, given, as before, a partition $\mathcal{P} = \{P_i\}_{i\in I}$ of A^k, and an

injective function $\gamma : [\ell] \to [k]$, each tuple $\bar{a}$ induces a partition of tuples $\bar{b}$ in A^ℓ according to which part P_i contains $\bar{a}[\bar{b}/\gamma]$. We think of this partition as a collection $(M_i^{\bar{a}})_{i\in I}$ of 0-1 matrices. For a prime number p, we say that two tuples $\bar{a}_1$ and $\bar{a}_2$ are *$\mathcal{P}$-p-m-similar* if they are in the same part of $\mathcal{P}$ *and* for every γ there is an invertible matrix $S \in \mathbb{F}_p^{A^m \times A^m}$ such that for each $i \in I$ we have $SM_i^{\bar{a}_1}S^{-1} = M_i^{\bar{a}_2}$. In other words, the sequences $(M_i^{\bar{a}_1})_{i\in I}$ and $(M_i^{\bar{a}_2})_{i\in I}$ are simultaneously similar, witnessed by S. We say the tuples are *$\mathcal{P}$-p-similar* if they are $\mathcal{P}$-p-m-similar for all $m \leq k/2$. The equivalence relation $\equiv^{\mathrm{IM}}_{k,p}$ is then the coarsest partition $\mathcal{P}$ that refines the partition into atomic types and such that any two tuples in the same part of $\mathcal{P}$ are $\mathcal{P}$-p-similar. Finally, for a set Q of prime numbers, $\bar{a}_1 \equiv^{\mathrm{IM}}_{k,Q} \bar{a}_2$ if, and only if, $\bar{a}_1 \equiv^{\mathrm{IM}}_{k,p} \bar{a}_2$ for each $p \in Q$. So, $\equiv^{\mathrm{IM}}_{k,Q}$ is the coarsest common refinement of the relations $(\equiv^{\mathrm{IM}}_{k,p})_{p\in Q}$.

Given a fixed set Q of primes with $|Q| = s$, it is possible to compute, for a structure $\mathfrak{A}$ with n elements, the partition of A^k into $\equiv^{\mathrm{IM}}_{k,Q}$ equivalence classes in time $sn^{O(k)}$. To see this, we note that the equivalence relation can be obtained by an iterated refinement process. First, let $\mathcal{P}_0$ be the partition of A^k into atomic types. Then, for each i, let $\mathcal{P}_{i+1}$ be the partition which places two tuples in the same class if, and only if, they are $\mathcal{P}_i$-p-similar for all $p \in Q$. This refinement process converges in at most n^k steps to the partition into $\equiv^{\mathrm{IM}}_{k,Q}$-equivalence classes. At each stage we compute, for each tuple $\bar{a} \in A^k$ and each injective function $\gamma\colon [2m] \to [k]$, the partition of A^{2m} into types, where $m = \lfloor k/2 \rfloor$. This suffices because $\mathcal{P}$-p-m-similarity implies $\mathcal{P}$-p-m'-similarity for all $m' < m$. Having computed the partition, we need to check for each pair of tuples and for each p in Q, whether the induced partitions are simultaneously similar. For this, we use the simultaneous matrix similarity test of Chistov et al. [8]. Since this runs in polynomial time, it follows that the whole procedure can be completed in time $sn^{O(k)}$.

Linear-Algebraic Logic. The study of logics with linear-algebraic operators over finite fields was initiated in [12], where FPR, the fixed-point logic with rank operators, was first introduced. As with fixed-point logics generally, the expressive power of FPR is naturally analysed by seeing it as a fragment of an infinitary logic, in this case with rank *quantifiers*. The notion of elementary equivalence that corresponds to this logic was given in terms of a game characterisation in [14], where the invertible map equivalences were also introduced. Here, we define, for any set Q of primes, an infinitary logic $\mathrm{LA}^\omega(Q)$ with quantifiers for *all* linear-algebraic operators over finite fields of characteristics in Q. This logic is not really intended for practical use. Instead it is designed to be strong enough so that inexpressibility results for $\mathrm{LA}^\omega(Q)$ carry over to any well-defined logic that extends first-order or fixed-point logic by any kind of linear-algebraic operators over Q.

We begin with a definition of linear-algebraic operators. Let $\mathbb{F}$ be a field and let $\mathcal{B}$ be a (non-empty, finite) set that serves as a supply of abstract basis elements. We consider the $\mathbb{F}$-vector space $\mathbb{F}^{\mathcal{B}}$. For each subset $K \subseteq \mathcal{B}$ we identify the vector space $\mathbb{F}^K$ with a subspace of $\mathbb{F}^{\mathcal{B}}$ in the natural way: since $\mathbb{F}^{\mathcal{B}} = \mathbb{F}^K \oplus \mathbb{F}^{\mathcal{B}\setminus K}$ we can (implicitly) set $\mathbb{F}^K = \mathbb{F}^K \oplus \{0\}$.

Let $m \geq 1$. Generally speaking, an *m-ary linear-algebraic operator* is just a function f that defines a linear-algebraic property $f(M_1, \ldots, M_m)$ of m-tuples of $\mathbb{F}$-linear transformations M_i on (subspaces of) $\mathbb{F}^{\mathcal{B}}$. To make things more precise, let $K_i, L_i \subseteq \mathcal{B}$, for $i \in [m]$, denote pairs of (non-empty) subsets of basis elements. We set $V_i = \mathbb{F}^{K_i}$ and $W_i = \mathbb{F}^{L_i}$. We consider m-tuples $(M_1, \ldots, M_m)$ consisting of $\mathbb{F}$-linear mappings $M_i\colon V_i \to W_i$ which are represented succinctly in terms of m-tuples $(M_1, \ldots, M_m)$ of $L_i \times K_i$-matrices with entries in $\mathbb{F}$. Then an m-ary linear-algebraic operator over $\mathbb{F}$ is a function f that takes such sequences $(M_1, \ldots, M_m)$ to some kind of linear-algebraic information $f(M_1, \ldots, M_m)$ about the sequence.

Now, to say that f outputs a "linear-algebraic information" means that the output of f is invariant under $\mathbb{F}$-vector space isomorphisms. Formally, let $\mathcal{C}$ be another (abstract) set of basis elements, where $|\mathcal{B}| = |\mathcal{C}|$, let $K'_i, L'_i \subseteq \mathcal{C}$ where $|K_i| = |K'_i|$ and $|L_i| = |L'_i|$ for $i \in [m]$, and let $(N_1, \dots, N_m)$ be a sequence of matrices $N_i \colon L'_i \times K'_i \to \mathbb{F}$, $i \in [m]$, analogously to the above. Moreover, let $V'_i = \mathbb{F}^{K'_i}$ and $W'_i = \mathbb{F}^{L'_i}$ for $i \in [m]$. Then we say that $(N_1, \dots, N_m)$ results from $(M_1, \dots, M_m)$ by means of an $\mathbb{F}$-vector space isomorphism if we can find an invertible $\mathbb{F}$-linear mapping $S \colon \mathbb{F}^{\mathcal{B}} \to \mathbb{F}^{\mathcal{C}}$ such that the following holds:

- For all $i \in [m]$, S maps each of the subspaces V_i and W_i in $\mathbb{F}^{\mathcal{B}}$ to the respective subspaces V'_i and W'_i in $\mathbb{F}^{\mathcal{C}}$. That is, if we represent S in terms of a $\mathcal{C} \times \mathcal{B}$-matrix with entries in $\mathbb{F}$, then we have that for each of the subblocks $K'_i \times K_i$, $i \in [m]$, the restriction $S \restriction_{(K'_i \times K_i)} \colon K'_i \times K_i \to \mathbb{F}$ of the matrix S to this block is invertible and we have that $S(a, b) = 0$ for all $a \in \mathcal{C} \setminus K'_i$ and $b \in K_i$ (and the analogous holds for all subblocks $L'_i \times L_i$ and the corresponding restrictions $S \restriction_{(L'_i \times L_i)} \colon L'_i \times L_i \to \mathbb{F}$ of S to the subblocks $L'_i \times L_i$).
- For each $i \in [m]$, the $\mathbb{F}$-vector space isomorphism S simultaneously transforms all linear operators $M_i \colon V_i \to W_i$ to the corresponding operators $N_i \colon V'_i \to W'_i$, that is for all $i \in [m]$ we have: $N_i \cdot S = S \cdot M_i$. Note that if we want to read this as a matrix equation, then we formally have to replace the matrix S by its restrictions to the subblocks $K'_i \times K_i$ and $L'_i \times L_i$ as we described above, that is $S \restriction_{(L'_i \times L_i)} \cdot M_i = N_i \cdot S \restriction_{(K'_i \times K_i)}$.

We require that a linear algebraic operator f outputs the same result for all pairs of matrix sequences $(M_1, \dots, M_m)$ and $(N_1, \dots, N_m)$ that are related via an $\mathbb{F}$-vector space isomorphism S (as above), that is $f(M_1, \dots, M_m) = f(N_1, \dots, N_m)$. This condition guarantees that f is not able to distinguish between isomorphic objects and here, in the realm of linear algebra, isomorphisms are $\mathbb{F}$-vector space isomorphisms. Besides this we do not put any kind of additional restrictions on f. For instance, f may not even be a computable function. Note that, though in introducing the function f, we considered a fixed set $\mathcal{B}$, really f defines, for any $\mathcal{B}$, a function on m-tuples of linear operators over subspaces of $\mathbb{F}^{\mathcal{B}}$. Without this, the notion of invariance would not make sense.

Now, we can associate with f a family of *Lindström quantifiers*. For simplicity, we restrict our attention to operators of a specific form without loss of generality. For an explanation of why no generality is lost, we refer the reader to the full paper [11]. Specifically, we assume that $K_i = L_i = \mathcal{B}$ for all i in the above definition, and we assume that the matrices are all 0-1 matrices. In other words, f is defined for a tuple of *square* 0-1 matrices all with the same index set.

Let τ_m denote a vocabulary with m distinct binary relations. Given an operator that defines such an f for each finite $\mathcal{B}$, for each $t \in \mathbb{N}$ we define a class of structures $\mathcal{K}^t_f$ in the vocabulary τ_m. We can think of an index set $\mathcal{B}$ with a collection $M_1, \dots, M_m$ of 0-1 $\mathcal{B} \times \mathcal{B}$ matrices as a τ_m-structure $(\mathcal{B}, M_1, \dots, M_m)$. The class $\mathcal{K}^t_f$ is then the collection of those τ_m-structures where $f(M_1, \dots, M_m) \geq t$. For each $\ell \geq 1$ we then have a quantifier $\mathcal{Q}^{t,\ell}_f$ such that if $I(\bar{x})$ is an $L[\sigma, \tau_m]$-interpretation of dimension ℓ, then $\mathcal{Q}^{t,\ell}_f I(\bar{x})$ is a formula true in a σ structure $\mathcal{A}$ if $I(\mathcal{A}) \in \mathcal{K}^t_f$.

The infinitary logic LA is defined as the closure of first-order logic under *infinitary* disjunction and conjunction, along with quantification $\mathcal{Q}^{t,\ell}_f$ for any linear algebraic operator f over any finite field. That is, if Φ is any set of formulas of LA, then $\bigvee \Phi$ and $\bigwedge \Phi$ are both formulas of LA. And, if f is an m-ary linear algebraic operator over a finite field, and $\Theta(\bar{x})$ is an ℓ-ary LA-interpretation of σ_m in τ, then $\mathcal{Q}^{t,\ell}_f \bar{x} \Theta$ is an LA τ-formula. We are interested in various fragments of the logic LA for which we introduce notation in the following definition.

▶ **Definition 1.** LA^k *is the collection of formulas of* LA *that contain at most* k *distinct variables, and* $\mathrm{LA}^\omega = \bigcup_{k\in\omega} \mathrm{LA}^k$. *For any set* Q *of primes, we write* $\mathrm{LA}(Q)$, $\mathrm{LA}^k(Q)$ *and* $\mathrm{LA}^\omega(Q)$ *to denote the restrictions of these logics to using only linear-algebraic operators over fields of characteristic* $p \in Q$.

If $\mathcal{L}$ *is any of the logics* LA, LA^ω, LA^k, $\mathrm{LA}(Q)$, $\mathrm{LA}^\omega(Q)$ *or* $\mathrm{LA}^k(Q)$, *and* $\ell \in \mathbb{N}$ *we write* ℓ-$\mathcal{L}$ *to denote the fragment of* $\mathcal{L}$ *where all algebraic quantifiers are* $\mathcal{Q}_f^{t,\ell}$ *for some* t *and* f. *In other words, interpretations are restricted to be of dimension* ℓ.

As pointed out above, the LA-logics are merely interesting from a theoretical point of view: in a precise sense they form a maximal extension of infinitary logics by linear-algebraic operators. On the other hand, we do not know of any (non-trivial) property that has a natural definition in the LA-logic, but not already in (infinitary) rank logic for instance (and it is open whether such example exists at all). Having introduced the linear-algebraic logic LA^ω and the invertible-map equivalences $\equiv^{\mathrm{IM}}_{k,Q}$, we are in a position to formulate their tight relationship.

▶ **Theorem 2.** *Let* $k \geq 2$ *be a positive integer and* Q *a set of prime numbers. For any finite structure* $\mathfrak{A}$ *and* $\bar{a}, \bar{b} \in A^k$, *the following are equivalent:*

1. $\bar{a} \equiv^{IM}_{k,Q} \bar{b}$; *and*
2. *for every formula* φ *of* $\mathrm{LA}^k(Q)$, $\mathfrak{A} \models \varphi[\bar{a}]$ *if, and only if,* $\mathfrak{A} \models \varphi[\bar{b}]$.

3 Cai-Fürer-Immerman Structures and Logic

In this section we describe a generalised variant of the *CFI-construction* due to Cai, Fürer, and Immerman [6]. It associates with each

- connected, 3-regular, and ordered (undirected) graph $G = (V, E, \leq)$,
- every prime field $\mathbb{F}_p$, $p \in \mathbb{P}$ (in this article, $\mathbb{P}$ denotes the set of all primes), and
- every vector $\lambda \in \mathbb{F}_p^V$

a structure we call the *CFI-structure* $\mathsf{CFI}[G; p; \lambda]$. Its signature is $\tau_{\mathsf{CFI}} = \{\preceq, R, C, I\}$ where R is a ternary relation symbol and where $\preceq, I, C$ are binary relation symbols. The universe A of $\mathsf{CFI}[G; p; \lambda]$ is $A = E \times \mathbb{F}_p$. The linear order $\leq$ on the vertex set V extends to a linear order on the edge set E (as the lexicographic order, for example). We use this linear order on E to define the following *total preorder* $\preceq$ on A: $(e, x) \preceq (f, y)$ if $e \leq f$. Note that $\preceq$ induces a linear order on the corresponding equivalence classes $e^p = e \times \mathbb{F}_p$. Clearly, each of these classes e^p is of size p. Since G is undirected every edge $e = (v, w) \in E$ comes with its corresponding *dual edge* $f = (w, v) \in E$. In what follows, we use the notation $e^{-1} = f$ to denote the dual of the edge $e \in E$. The relations I and C are defined as follows.

- The *cycle relation* C defines the cyclic structure of the additive group of $\mathbb{F}_p$ on each of the edge classes e^p. More precisely,

$$C = \bigcup_{e\in E} \{((e, x), (e, x + 1 \bmod p)) : x \in \mathbb{F}_p\}.$$

- The *inverse relation* I relates additive inverses for dual edges. Formally,

$$I = \bigcup_{e\in E} \{((e, x), (e^{-1}, -x) : x \in \mathbb{F}_p\}.$$

Note that while the cycle relation C defines a directed graph, the inverse relation I is symmetric. Furthermore, observe that the relations $\preceq, C$ and I are defined independently of the load vector λ and so only depend on the underlying graph G and the prime field $\mathbb{F}_p$. In

contrast, the *CFI-relation* $R = R^\lambda$ is defined using the load vector λ as follows. For each $v \in V$, we let $vE \subseteq V$ denote the set of neighbours of v in G, that is $E(v) = \{v\} \times vE \subseteq E$ is the set of edges outgoing from v. Since G is 3-regular we have that $|vE| = 3$ for each $v \in V$. For $v \in V$ let $vE = \{w_1, w_2, w_3\}$ where $w_1 < w_2 < w_3$. The *CFI-relation* $R^\lambda(v)$ at vertex v is defined as follows:

$$R^\lambda(v) = \{((w_1, x_1), (w_2, x_2), (w_3, x_3)) : x_1 + x_2 + x_3 = \lambda(v) \bmod p\}.$$

The full CFI-relation R^λ of $\mathsf{CFI}\,[G; p; \lambda]$ is given as $R^\lambda = \bigcup_{v \in V} R^\lambda(v)$.

▶ **Theorem 3.** *Two CFI-structures* $\mathsf{CFI}\,[G; p; \lambda], \mathsf{CFI}\,[G; p; \sigma]$ *over the same graph* G *are isomorphic, if, and only if,*

$$\sum \lambda := \sum_{v \in V} \lambda(v) = \sum_{v \in V} \sigma(v) =: \sum \sigma.$$

The CFI-construction unfolds its full power when it is based on a family of underlying graphs that is highly connected. A good choice is to take 3-regular *expander graphs* with $\mathcal{O}(n)$ vertices, as such graphs have a linear lower bound on the size of their separators (which means that we cannot disconnect the graphs into components of size $\leq n/2$ by removing fewer than $\Omega(n)$ vertices).

▶ **Theorem 4** (see e.g. Example 2.2 in [23])**.** *There exists a family of 3-regular, connected expander graphs* $\mathcal{F} = \{G_n : n \in \mathbb{N}\}$ *such that each graph* G_n*,* $n \in \mathbb{N}$*, has* $\mathcal{O}(n)$ *vertices.*

Of course, we can also assume that the graphs in $\mathcal{F}$ are *ordered* just by adding to each graph $G_n = (V_n, E_n) \in \mathcal{F}$ an arbitrary linear order on V_n. From this family $\mathcal{F}$ of 3-regular, connected, ordered expander graphs G_n with $\mathcal{O}(n)$ many vertices we construct, for every $p \in \mathbb{P}$, the CFI-class $\mathsf{CFI}\,[\mathcal{F}; p]$ consisting of all CFI-structures over graphs from $\mathcal{F}$ that is

$$\mathsf{CFI}\,[\mathcal{F}; p] = \bigcup_{n,\lambda} \mathsf{CFI}\,[G_n; p; \lambda].$$

The *CFI-problem* (over $\mathcal{F}$ and $p \in \mathbb{P}$) is to decide, given a structure $\mathsf{CFI}\,[G; p; \lambda] \in \mathsf{CFI}\,[\mathcal{F}; p]$ whether $\sum \lambda = 0$. For the original form of the CFI-construction, it was shown in [6] that this problem is undefinable in counting logic with sublinearly many variables. Also the generalization to more powerful variants, and in particular to our class $\mathsf{CFI}\,[\mathcal{F}; p]$ is well-known.

▶ **Theorem 5.** *For any two structures* $\mathsf{CFI}\,[G_n; p; \lambda], \mathsf{CFI}\,[G_n; p; \sigma] \in \mathsf{CFI}\,[\mathcal{F}; p]$ *we have*

$$\mathsf{CFI}\,[G_n; p; \lambda] \equiv^{\Omega(n)} \mathsf{CFI}\,[G_n; p; \sigma].$$

Thus, from the perspective of counting logic (with $\Omega(n)$ many variables) CFI-structures over the same underlying graph G_n look the same although, for load vectors λ and σ with $\sum \lambda \neq \sum \sigma$, we know that $\mathsf{CFI}\,[G_n; p; \lambda]$ and $\mathsf{CFI}\,[G_n; p; \sigma]$ are not isomorphic.

▶ **Definition 6.** *Let* $\ell \geq 1$*. We say that a structure* $\mathfrak{A}$ *with automorphism group* Γ *is* ℓ-homogeneous *if for all* $k \geq 1$ *and all* k*-tuples* $\bar{a}, \bar{b} \in A^k$ *we have that*

$$(\mathfrak{A}, \bar{a}) \equiv^{\ell \cdot k} (\mathfrak{A}, \bar{b}) \text{ if, and only if, } \Gamma(\bar{a}) = \Gamma(\bar{b}).$$

In other words, the equivalence relation $\equiv^{\ell \cdot k}$ *refines* k*-tuples in* $\mathfrak{A}$ *up to orbits. Moreover, we say that a class* $\mathcal{K}$ *of structures is* homogeneous *if for some constant* $\ell \geq 1$ *each structure* $\mathfrak{A} \in \mathcal{K}$ *is* ℓ*-homogeneous.*

▶ **Theorem 7.** *For every prime p, the class $\mathsf{CFI}[\mathcal{F};p]$ is homogeneous.*

This theorem has been established in [17]. Homogeneity of CFI-structures is very useful because it implies that counting logic (indeed, FPC) can order k-tuples up to orbits. There are counting-type formulae $\mathrm{CT}_{\ell\cdot k}(\bar{x},\bar{y}) \in \mathrm{FPC}$ (see [24]) that define a linear preorder on k-tuples which distinguishes between all pairs of k-tuples in different orbits, and these formulae use only $\mathcal{O}(\ell\cdot k)$ many variables. One key consequence of homogeneity is that on the class of CFI structures, the relations $\equiv^k$ and $\equiv^{\mathrm{IM}}_{k,Q}$ coincide for k above some constant threshold. Indeed, $\equiv^{\mathrm{IM}}_{k,Q}$ is always at least as fine as $\equiv^k$ and no finer than the equivalence given by the partition into automorphism orbits. When the former and the latter are the same, $\equiv^{\mathrm{IM}}_{k,Q}$ must be the same. In particular, this means that the counting-type formulas $\mathrm{CT}_{\ell\cdot k}(\bar{x},\bar{y})$ define a pre-order on the $\equiv^{\mathrm{IM}}_{k,Q}$ equivalence classes.

4 Indistinguishability of CFI-structures using Linear-Algebraic Operators

In this section, we state our main technical result, that CFI-structures over a prime field $\mathbb{F}_p$ cannot be distinguished by means of *any* linear-algebraic operator over a field $\mathbb{F}$ with $\mathrm{char}(\mathbb{F}) \neq p$ if we apply such linear-algebraic operators to $\mathrm{C}^{\Omega(n)}$-definable matrices. For what follows, recall that we consider CFI-structures over a fixed class of expander graphs $\mathcal{F} = \{G_n : n \in \mathbb{N}\}$ where each graph G_n has $\mathcal{O}(n)$ vertices and is ordered, connected, and three-regular.

The exact formulation of the results requires some background from associative algebra. This can be found in the monograph [26], for example. The definitions and results we use are also summarized in the full version of this paper [11, Sec. 5].

The partition of 2ℓ-tuples in a structure $\mathfrak{A}$ into $\equiv^k$-classes (when $k \geq 3\ell$) induces a *coherent configuration* in the sense of [7, Chap. 3]. This implies, in particular, that if we think of this partition as a collection $M_1,\dots,M_s$ of 0-1 matrices with rows and columns indexed by A^ℓ then, for any field $\mathbb{F}$, they form the basis of an $\mathbb{F}$-algebra. That is to say, they are the basis of an $\mathbb{F}$-vector space that is also closed under matrix multiplication. We denote this algebra $\mathsf{Alg}[\mathfrak{A};\ell;\mathrm{C}^k;\mathbb{F}]$ and the *ordered* collection of matrices that forms its basis $\mathrm{Basis}[\mathfrak{A};\ell;\mathrm{C}^k]$. Note that the latter does not depend on the choice of $\mathbb{F}$.

We say that two structures $\mathfrak{A}$ and $\mathfrak{B}$ are $(\mathbb{F};\ell;\mathrm{C}^k)$-isomorphic if $\mathfrak{A} \equiv^k \mathfrak{B}$ and, if $\mathcal{M} = \mathrm{Basis}[\mathfrak{A};\ell;\mathrm{C}^k] = (M_1,\dots,M_s)$ and $\mathcal{N} = \mathrm{Basis}[\mathfrak{B};\ell;\mathrm{C}^k] = (N_1,\dots,N_s)$, then there is an invertible map $S : \mathbb{F}^{A^\ell} \to \mathbb{F}^{B^\ell}$ such that for each $i \in [s]$, $SM_iS^{-1} = N_i$. In short, the two sequences of matrices $\mathrm{Basis}[\mathfrak{A};\ell;\mathrm{C}^k]$ and $\mathrm{Basis}[\mathfrak{B};\ell;\mathrm{C}^k]$ are *simultaneously similar* as witnessed by S. In particular, the $\mathbb{F}$-algebras $\mathsf{Alg}[\mathfrak{A};\ell;\mathrm{C}^k;\mathbb{F}]$ and $\mathsf{Alg}[\mathfrak{B};\ell;\mathrm{C}^k;\mathbb{F}]$ are isomorphic.

For CFI-structures, $\mathfrak{A} = \mathsf{CFI}[G_n;p;\lambda]$ and $\mathfrak{B} = \mathsf{CFI}[G_n;p;\sigma]$, by the homogeneity property, we know that the partition into $\equiv^k$-classes is the same as the partition into automorphism orbits. This allows us to show that when such structures are $(\mathbb{F};\ell;\mathrm{C}^k)$-isomorphic, they cannot be distinguished by any $\mathbb{F}$-linear-algebraic operators. Hence, the key technical theorem we prove is the following.

▶ **Theorem 8.** *There is $\epsilon > 0$ s.t. for large enough $n > 0$ the following holds. Let $\mathfrak{A} = \mathsf{CFI}[G_n;p;\lambda]$ and $\mathfrak{B} = \mathsf{CFI}[G_n;p;\sigma]$ denote CFI-structures over G_n and let $\mathbb{F}$ be a field such that $\mathrm{char}(\mathbb{F}) \neq p$. Then $\mathfrak{A}$ and $\mathfrak{B}$ are $(\mathbb{F};\ell;\mathrm{C}^k)$-isomorphic where $\ell = \lfloor \epsilon n \rfloor$ and $k = 3\ell$.*

Theorem 8 is a consequence of Theorem 9 where we show that, for the above scenario, the simultaneous similarity of the counting-logic bases is definable in counting logic. To state Theorem 9, we introduce some terminology. Consider two sequences of matrices

$\mathcal{M} = (M_i)_{i\in[s]}$ and $\mathcal{N} = (N_i)_{i\in[s]}$, where each M_i is an $I \times I$ matrix over $\mathbb{F}$ and each N_i is a $J \times J$ matrix over $\mathbb{F}$. Here I and J are two arbitrary index sets of the same size. We define the set $\mathrm{H}_{\mathcal{M},\mathcal{N}}$ of $I \times J$-matrices X over $\mathbb{F}$ which satisfy $M_i X = X N_i$ for all $i \in [s]$. Note that the two sequences $\mathcal{M} = (M_i)_{i\in[s]}$ and $\mathcal{N} = (N_i)_{i\in[s]}$ are simultaneously similar if, and only if, $\mathrm{H}_{\mathcal{M},\mathcal{N}}$ contains an *invertible* matrix.

Note that $\mathrm{H}_{\mathcal{M},\mathcal{N}}$ is an $\mathbb{F}$-vector space. Next, consider the set $\mathrm{C}_{\mathcal{M}}$ of $I \times I$-square matrices Z over $\mathbb{F}$ such that $M_k Z = Z M_k$ for all $k \in K$. The set $\mathrm{C}_{\mathcal{M}}$ is called the *centraliser* of the matrix family $\mathcal{M}$. It is easy to verify that $\mathrm{C}_{\mathcal{M}}$ forms an $\mathbb{F}$-algebra. Moreover, by considering matrix multiplication (from the left) by elements from $\mathrm{C}_{\mathcal{M}}$, the $\mathbb{F}$-vector space $\mathrm{H}_{\mathcal{M},\mathcal{N}}$ turns into a $\mathrm{C}_{\mathcal{M}}$-module. With this, we can state the technical result.

▶ **Theorem 9.** *Let $t \geq 3$ be a constant such that all CFI-structures in $\mathsf{CFI}[\mathcal{F}; p]$ are t-homogeneous for all $p \in \mathbb{P}$. Then there exists a constant $c \geq 1$ such that the following holds. Let $\ell \geq 1$ and let $k \geq t\ell$. Then for each $p \in \mathbb{P}$ there exists a C^{ck}-sentence φ such that for all pairs of CFI-structures $\mathfrak{A} = \mathsf{CFI}[G_n; p; \lambda]$ and $\mathfrak{B} = \mathsf{CFI}[G_n; p; \sigma]$ over the same underlying graph $G_n \in \mathcal{F}$ we have that $(\mathfrak{A}, \mathfrak{B}) \models \varphi$ if, and only if, over every field $\mathbb{F}$ with $\mathrm{char}(\mathbb{F}) \neq p$, the $C_{\mathcal{M}}$-module $H_{\mathcal{M},\mathcal{N}}$ contains an invertible matrix $S \in H_{\mathcal{M},\mathcal{N}}$ where $\mathcal{M} = \mathit{Basis}[\mathfrak{A}, \ell, k]$ and $\mathcal{N} = \mathit{Basis}[\mathfrak{B}, \ell, k]$.*

We can derive Theorem 8 from Theorem 9 as follows. First of all, let $c \geq 1$ and $t \geq 3$ be the constants according to Theorem 9. Let $p \in \mathbb{P}$. Then, by Theorem 5, we can find $\delta > 0$ such that for all large enough $n > 1$ we have $\mathfrak{A} \equiv^{\lfloor \delta n \rfloor} \mathfrak{B}$ where $\mathfrak{A} = \mathsf{CFI}[G_n; p; \lambda]$ and $\mathfrak{B} = \mathsf{CFI}[G_n; p; \sigma]$ are two CFI-structures over $\mathbb{F}_p$ and the same underlying expander graph $G_n \in \mathcal{F}$ with $\mathcal{O}(n)$ many vertices. Let $\epsilon = \frac{1}{tc}\delta$. Then $(\mathfrak{A}, \mathfrak{A}) \equiv^{\lfloor t c \epsilon n \rfloor} (\mathfrak{A}, \mathfrak{B})$. Let $\mathbb{F}$ be a field such that $\mathrm{char}(\mathbb{F}) \neq p$. Let $\ell = \lfloor \epsilon n \rfloor$ and $k = \lfloor t \epsilon n \rfloor$. We consider $\mathcal{M} = \mathrm{Basis}[\mathfrak{A}, \ell; \mathrm{C}^k]$ and $\mathcal{N} = \mathrm{Basis}[\mathfrak{B}, \ell; \mathrm{C}^k]$. Since the formula φ according to Theorem 9 contains at most $ck = c \cdot \lfloor t \epsilon n \rfloor \leq \lfloor \delta n \rfloor$ many variables, this formula cannot distinguish between the ordered pairs of CFI-structures $(\mathfrak{A}, \mathfrak{A})$ and $(\mathfrak{A}, \mathfrak{B})$. On the other hand, by its properties stated in Theorem 9, φ would need to distinguish between $(\mathfrak{A}, \mathfrak{A})$ and $(\mathfrak{A}, \mathfrak{B})$ if no invertible matrix $S \in \mathrm{H}_{\mathcal{M},\mathcal{N}}$ would exist. Indeed, note that the $\mathrm{C}_{\mathcal{M}}$-module $\mathrm{H}_{\mathcal{M},\mathcal{M}}$ contains an invertible matrix $S \in \mathrm{H}_{\mathcal{M},\mathcal{M}}$ over every field $\mathbb{F}$ for trivial reasons; for instance it contains the permutation matrix that corresponds to the identity automorphism of $\mathfrak{A}$. Hence, we can conclude that $\mathrm{H}_{\mathcal{M},\mathcal{N}}$ contains an invertible matrix which shows that $\mathfrak{A}$ and $\mathfrak{B}$ are $(\mathbb{F}; \ell; \mathrm{C}^k)$-isomorphic, and thus Theorem 8 follows, because $(\mathbb{F}; \ell; \mathrm{C}^k)$-isomorphic structures are also $(\mathbb{F}; \ell; \mathrm{C}^{3\ell})$-isomorphic since $k \geq 3\ell$.

We now outline a proof strategy for Theorem 9. The full proof is rather long, and is presented in detail in the full paper [11]. First, we fix a *prime field* $\mathbb{F}$ with $\mathrm{char}(\mathbb{F}) \neq p$. We construct a sentence $\varphi_{\mathbb{F}} \in \mathrm{C}^\omega$, with at most $c \cdot k$ many variables, which holds in the ordered pair $(\mathfrak{A}, \mathfrak{B})$ of CFI-structures $\mathfrak{A}$ and $\mathfrak{B}$ if, and only if, $\mathrm{H}_{\mathcal{M},\mathcal{N}}$ (considered as a $\mathrm{C}_{\mathcal{M}}$-module over the $\mathbb{F}$-algebra $\mathrm{C}_{\mathcal{M}}$) contains an invertible matrix S. The desired sentence φ according to Theorem 9 is then the conjunction over all sentences $\varphi_{\mathbb{F}}$ for *prime fields* $\mathbb{F}$ with $\mathrm{char}(\mathbb{F}) \neq p$.

Step (I). As a first step we show that it suffices to restrict our considerations to *prime fields*. This is because the matrix families $\mathcal{M}$ and $\mathcal{N}$ we are interested in only contain 0-1 matrices. Such families are simultaneously similar over a field $\mathbb{F}$ if, and only if, they are simultaneously similar over the prime subfield of $\mathbb{F}$. The restriction is important because we need to use the result (originally proved in [18]) about defining solutions to system of linear equations. The result is that if a system of linear equations over a prime field $\mathbb{F}$ is represented by a homogeneous structure $\mathfrak{A}$ with Abelian automorphism group, and such that

the order of the automorphism group of $\mathfrak{A}$ is co-prime with char($\mathbb{F}$), then a solution to the system can be defined by a formula of counting logic. This is because we can show that in this case, there must exist a solution that is symmetric, i.e. invariant under the action of the automorphism group. In fact, we need here a stronger result saying that not only single solutions, but whole solution spaces are definable in counting logic. This was recently shown in [17] and our full paper [11] also contains a proof sketch.

Now, the CFI structures in $\mathsf{CFI}[\mathcal{F}; p]$ have Abelian automorphism groups of order a power of p, so systems of linear equations suitably defined from them will have the required property. Our aim is to reduce the problem of deciding whether $\mathrm{H}_{\mathcal{M},\mathcal{N}}$ contains an invertible matrix to solving a system of linear equations over $\mathbb{F}$. The condition $M_i X = X N_i$ for all i easily yields a system of such equations with unknowns for the enrtries of X. The question is how to enforce that X is invertible.

Step (II). To carry out the reduction, we use the result from [8] that if $\mathrm{H}_{\mathcal{M},\mathcal{N}}$ contains an invertible matrix, then it is *cyclic* as a $\mathrm{C}_{\mathcal{M}}$-module. This means that $\mathrm{H}_{\mathcal{M},\mathcal{N}}$ is generated by a single element: there is a matrix $X \in \mathrm{H}_{\mathcal{M},\mathcal{N}}$ such that $\mathrm{C}_{\mathcal{M}} X = \{ZX : Z \in \mathrm{C}_{\mathcal{M}}\} = \mathrm{H}_{\mathcal{M},\mathcal{N}}$. This gives a necessary but not sufficient condition for the existence of an invertible matrix in $\mathrm{H}_{\mathcal{M},\mathcal{N}}$. To obtain a necessary and sufficient condition, we use the particular structure of the matrix families $\mathcal{M}$ and $\mathcal{N}$ that follow from the fact that they are generated by the $\equiv^k$-equivalence classes. Roughly speaking, the equivalence relation $\equiv^k$ partitions the row-column index sets of these matrices into classes, and we can think of the matrices as linear combinations of "small" matrices, which are over these individual blocks. This means that the invertible S we are looking for can also be decomposed into the sum of smaller block matrices. Our families $\mathcal{M}$ and $\mathcal{N}$ have the property that they are *locally simultaneously similar*, i.e. we can find similarity transformations for each small block. With this, it becomes possible to prove that the cyclicity of $\mathrm{H}_{\mathcal{M},\mathcal{N}}$ is both necessary and sufficient for the existence of an invertible matrix. It also means that we can restrict ourselves to a certain substructure of this module. To be precise, we let $\mathrm{C}^{\mathrm{D}}_{\mathcal{M}}$ be the subalgebra of $\mathrm{C}_{\mathcal{M}}$ consisting of those matrices that are zero outside the relevant blocks. Then, $\mathrm{H}^{\mathrm{D}}_{\mathcal{M},\mathcal{N}}$ is similarly the collection of matrices in $\mathrm{H}_{\mathcal{M},\mathcal{N}}$ that are zero outside the relevant blocks and this can be seen as a $\mathrm{C}^{\mathrm{D}}_{\mathcal{M}}$-module. For the particular matrix families $\mathcal{M} = \mathrm{Basis}[\mathfrak{A}, \ell, k]$ and $\mathcal{N} = \mathrm{Basis}[\mathfrak{B}, \ell, k]$, we are able to show that they are simultaneously similar if, and only if, $\mathrm{H}^{\mathrm{D}}_{\mathcal{M},\mathcal{N}}$ is cyclic as a $\mathrm{C}^{\mathrm{D}}_{\mathcal{M}}$-module.

As a consequence, to check whether $\mathrm{H}_{\mathcal{M},\mathcal{N}}$ contains an invertible matrix, it suffices to check whether the $\mathrm{C}^{\mathrm{D}}_{\mathcal{M}}$-module $\mathrm{H}^{\mathrm{D}}_{\mathcal{M},\mathcal{N}}$ is cyclic.

Step (III). The third step is the core of our whole argument. We combine results on the FPC-definability of the automorphism groups and orbits of CFI-structures with Maschke's Theorem, an important result from the representation theory of finite groups, to show that the $\mathbb{F}$-algebra $\mathrm{C}^{\mathrm{D}}_{\mathcal{M}}$ is *semisimple*.

Recall that for a finite group G and a field $\mathbb{F}$, the *group algebra* $\mathbb{F}[G]$ is the $\mathbb{F}$-algebra whose elements are formal sums of the form $\sum_{g \in G} r_g g$ with coefficients $r_g \in \mathbb{F}$. Addition and scalar multiplication are defined component-wise and multiplication is defined by convolution on the group elements. Maschke's theorem tells us that $\mathbb{F}[G]$ is semisimple if, and only if, char($\mathbb{F}$) does not divide the order of G.

For an algebra A, an A-module M is called *simple* if every submodule of M is trivial (either 0 or M itself) and *semisimple* if it is the direct sum of simple modules. From the semi-simplicity of $\mathrm{C}^{\mathrm{D}}_{\mathcal{M}}$ we are able to show that the $\mathrm{C}^{\mathrm{D}}_{\mathcal{M}}$-module $\mathrm{H}^{\mathrm{D}}_{\mathcal{M},\mathcal{N}}$ is semisimple.

Step (IV). The key property of semisimple modules is that they have an essentially canonical decomposition as the sum of simple modules. So, $\mathrm{H}^{\mathrm{D}}_{\mathcal{M},\mathcal{N}}$ can be decomposed as the sum of simple modules, and the isomorphism types of modules that occur in that decomposition and their respective multiplicities completely determine the isomorphism type of $\mathrm{H}^{\mathrm{D}}_{\mathcal{M},\mathcal{N}}$. Moreover, we show that we can define generating sets for the respective submodules in counting logic by using at most $c \cdot k$ many variables. This is because, essentially, these generating sets can be obtained as the solution sets of a system of linear equations, and we are able to use the results mentioned in Step (I) above.

Step (V). Finally, we construct the formula $\varphi_{\mathbb{F}}$. By (II), the formula $\varphi_{\mathbb{F}}$ needs to verify that the *semisimple* $\mathrm{C}^{\mathrm{D}}_{\mathcal{M}}$-module $\mathrm{H}^{\mathrm{D}}_{\mathcal{M},\mathcal{N}}$ is cyclic. We approach this problem by expressing a more general query, namely we determine the full isomorphism type of the module $\mathrm{H}^{\mathrm{D}}_{\mathcal{M},\mathcal{N}}$ by means of a formula of counting logic. First of all, we start by determining the isomorphism types of all simple subalgebras of $\mathrm{C}^{\mathrm{D}}_{\mathcal{M}}$. This we can easily do in counting logic because $\mathrm{C}^{\mathrm{D}}_{\mathcal{M}}$ has an (FPC-definable) ordered basis. This implies that the isomorphism type of $\mathrm{H}^{\mathrm{D}}_{\mathcal{M},\mathcal{N}}$ is (uniquely) determined by the multiplicities of the simple subalgebras of $\mathrm{C}^{\mathrm{D}}_{\mathcal{M}}$ as they occur in a decomposition of $\mathrm{H}^{\mathrm{D}}_{\mathcal{M},\mathcal{N}}$ into a direct sum of simple submodules. By using our decomposition from Step (IV), we can easily determine those multiplicities componentwise, since we can linearly order (again in an FPC-definable way) each of the "small" submodules that occur in the decomposition of $\mathrm{H}^{\mathrm{D}}_{\mathcal{M},\mathcal{N}}$. In this way we can determine the multiplicities for each individual component which add up to the total multiplicities for the whole module $\mathrm{H}^{\mathrm{D}}_{\mathcal{M},\mathcal{N}}$. Since the isomorphism type determines the cyclicity of the module, we can obtain our desired formula $\varphi_{\mathbb{F}}$ by selecting modules with appropriate isomorphism types.

5 Main results

In this section we spell out the consequences of the main technical result, Theorem 8, for approximations of isomorphism and for logics with linear-algebraic operators.

With regard to the relations $\equiv^{\mathrm{IM}}_{k,Q}$ as approximations of isomorphism, it follows immediately that as long as $Q \neq \mathbb{P}$, i.e. Q is not the set of all primes, there is no k for which $\equiv^{\mathrm{IM}}_{k,Q}$ coincides with isomorphism on all structures.

▶ **Corollary 10.** *If $Q \neq \mathbb{P}$, there is no fixed k such that $\equiv^{IM}_{k,Q}$ coincides with isomorphism on all structures.*

Proof. Fix a prime $p \notin Q$. Then, for each k, we have, by Theorem 8 a pair of structures $\mathfrak{A} = \mathsf{CFI}\,[G_n; p; \lambda]$ and $\mathfrak{B} = \mathsf{CFI}\,[G_n; p; \sigma]$ that are $(\mathbb{F}_q; \ell; \mathrm{C}^k)$-isomorphic, for all $q \neq p$, though $\sum \lambda \neq \sum \sigma$. It follows that $\mathfrak{A} \equiv^{\mathrm{IM}}_{k,Q} \mathfrak{B}$, but $\mathfrak{A} \ncong \mathfrak{B}$, by Theorem 3. ◀

The consequences for the expressive power of the logic LA^{ω} are also immediate.

▶ **Corollary 11.** *If $Q \neq \mathbb{P}$, there is a class of structures that is not definable in $\mathrm{LA}^{\omega}(Q)$.*

Proof. Fix a prime $p \notin Q$ and consider the class $\mathcal{C}$ of structures of the form $\mathsf{CFI}\,[G_n; p; \lambda]$ where $\sum \lambda = 0$. This is an isomorphism-closed class of structures by Theorem 3. Suppose it were defined by a sentence φ of $\mathrm{LA}^{\omega}(Q)$. Let ℓ be the maximum dimension of an interpretation used with any quantifier in φ and choose k such that $k \geq 3\ell$ and k is greater than the number of variables in φ. Then, by Theorem 8, we have a structure $\mathfrak{A} = \mathsf{CFI}\,[G_n; p; \lambda] \in \mathcal{C}$ which is $(\mathbb{F}_q; \ell; \mathrm{C}^k)$-isomorphic to every structure $\mathsf{CFI}\,[G_n; p; \sigma]$. Letting $\mathfrak{B}$ be such a structure where $\sigma \neq 0$, we have that $\mathfrak{B} \models \varphi$, contradicting the assumption that φ defines $\mathcal{C}$. ◀

It should be noted that the class of structures $\mathcal{C}$ defined in the proof of Corollary 11 is decidable in polynomial time. This is because the class can be decided by solving systems of linear equations, for example by Gaussian elimination. Thus, we know that there exists a PTIME property that $\mathrm{LA}^{\omega}(Q)$ cannot express as long as $Q \neq \mathbb{P}$. Since this logic subsumes any extension of fixed-point logic with Q-linear algebraic operators, we also have the following conclusion.

▶ **Corollary 12.** *If $Q \neq \mathbb{P}$, no extension of fixed-point logic with Q-linear algebraic operators captures PTIME.*

We can say more. The class $\mathcal{C}$ is not just decidable in PTIME, but also definable in *choiceless polynomial time* (CPT) (see [25]). We do not define the class CPT here but details may be found in [5]. Thus, the following corollary is immediate.

▶ **Corollary 13.** *If $Q \neq \mathbb{P}$, no extension of fixed-point logic with Q-linear algebraic operators captures CPT.*

On the other hand it remains an intriguing open question whether CPT captures all of rank logic, for example.

References

1 A. Atserias, A. Bulatov, and A. Dawar. Affine Systems of Equations and Counting Infinitary Logic. *Theoretical Computer Science*, 410:1666–1683, 2009.

2 A. Atserias and E. N. Maneva. Sherali-Adams relaxations and indistinguishability in counting logics. *SIAM J. Comput.*, 42:112–137, 2013.

3 L. Babai. Graph Isomorphism in Quasipolynomial Time [extended abstract]. In *Proc. 48th Annual ACM SIGACT Symp. Theory of Computing, STOC*, pages 684–697, 2016.

4 A. Barghi and I Ponomarenko. Non-Isomorphic Graphs with Cospectral Symmetric Powers. *Electr. J. Comb.*, 16(1), 2009.

5 A. Blass, Y. Gurevich, and S. Shelah. On Polynomial Time Computation Over Unordered Structures. *Journal of Symbolic Logic*, 67(3):1093–1125, 2002.

6 J. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

7 P.J. Cameron. *Permutation Groups*. London Mathematical Society Student Texts. Cambridge University Press, 1999.

8 A. Chistov, G. Ivanyos, and M. Karpinski. Polynomial Time Algorithms for Modules over Finite Dimensional Algebras. In *Proceedings of ISSAC '97*, pages 68–74. ACM, 1997.

9 A. Dawar. The nature and power of fixed-point logic with counting. *ACM SIGLOG News*, 2(1):8–21, 2015.

10 A. Dawar, E. Grädel, B. Holm, E. Kopczynski, and W. Pakusa. Definability of linear equation systems over groups and rings. *Logical Methods in Computer Science, Special Issue dedicated to CSL 2012*, 2013. URL: `http://www.lmcs-online.org/ojs/viewarticle.php?id=1325`.

11 A. Dawar, E. Grädel, and W. Pakusa. Approximations of Isomorphism and Logics with Linear-Algebraic Operators. arXiv abs/1902.06648. `arXiv:1902.06648`.

12 A. Dawar, M. Grohe, B. Holm, and B. Laubner. Logics with Rank Operators. In *Proceedings of LICS 2009*, pages 113–122, 2009.

13 A. Dawar and B. Holm. Tractable Approximations of Graph Isomorphism. forthcoming.

14 A. Dawar and B. Holm. Pebble Games with Algebraic Rules. *Fundam. Inform.*, 150(3-4):281–316, 2017.

15 A. Dawar and D. Vagnozzi. Generalizations of k-Weisfeiler-Leman Partitions and Related Graph Invariants. forthcoming.

16 H. Derksen. The Graph Isomorphism Problem and approximate categories. *J. Symb. Comput.*, 59:81–112, 2013. doi:10.1016/j.jsc.2013.06.002.

17 E. Grädel, M. Grohe, B. Pago, and W. Pakusa. A Finite-Model-Theoretic View on Propositional Proof Complexity. *Logical Methods in Computer Science*, 15:1:4:1–4:53, 2019.

18 E. Grädel and W. Pakusa. Rank logic is dead, long live rank logic! *Journal of Symbolic Logic*, 2019.

19 M. Grohe. The quest for a logic capturing PTIME. In *Proceedings of the 23rd IEEE Symposium on Logic in Computer Science (LICS'08)*, pages 267–271, 2008.

20 M. Grohe. *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory*. Cambridge University Press, 2017.

21 M. Grohe and M.Otto. Pebble Games and linear equations. *J. Symb. Log.*, 80:797–844, 2015.

22 B. Holm. *Descriptive Complexity of Linear Algebra*. PhD thesis, University of Cambridge, 2010.

23 S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.

24 M. Otto. *Bounded Variable Logics and Counting*. Springer, 1997.

25 W. Pakusa. *Linear Equation Systems and the Search for a Logical Characterisation of Polynomial Time*. PhD thesis, RWTH Aachen University, 2016.

26 R.S. Pierce. *Associative Algebras*. Graduate Texts in Mathematics. Springer, 1982.

Counting Answers to Existential Questions

Holger Dell
Cluster of Excellence (MMCI), Saarland Informatics Campus (SIC), Saarbrücken, Germany
hdell@mmci.uni-saarland.de

Marc Roth
Cluster of Excellence (MMCI), Saarland Informatics Campus (SIC), Saarbrücken, Germany
mroth@mmci.uni-saarland.de

Philip Wellnitz
Max Planck Institute for Informatics, Saarland Informatics Campus (SIC), Saarbrücken, Germany
wellnitz@mpi-inf.mpg.de

Abstract

Conjunctive queries select and are expected to return certain tuples from a relational database. We study the potentially easier problem of *counting* all selected tuples, rather than enumerating them. In particular, we are interested in the problem's parameterized and data complexity, where the query is considered to be small or even fixed, and the database is considered to be large. We identify two structural parameters for conjunctive queries that capture their inherent complexity: The dominating star size and the linked matching number. If the *dominating star size* of a conjunctive query is large, then we show that counting solution tuples to the query is at least as hard as counting dominating sets, which yields a fine-grained complexity lower bound under the Strong Exponential Time Hypothesis (SETH) as well as a #W[2]-hardness result in parameterized complexity. Moreover, if the *linked matching number* of a conjunctive query is large, then we show that the structure of the query is so rich that arbitrary queries up to a certain size can be encoded into it; in the language of parameterized complexity, this essentially establishes a #A[2]-completeness result.

Using ideas stemming from Lovász (1967), we lift complexity results from the class of conjunctive queries to arbitrary existential or universal formulas that might contain inequalities and negations on constraints over the free variables. As a consequence, we obtain a complexity classification that refines and generalizes previous results of Chen, Durand, and Mengel (ToCS 2015; ICDT 2015; PODS 2016) for conjunctive queries and of Curticapean and Marx (FOCS 2014) for the subgraph counting problem. Our proof also relies on graph minors, and we show a strengthening of the Excluded-Grid-Theorem which might be of independent interest: If the linked matching number (and thus the treewidth) is large, then not only can we find a large grid somewhere in the graph, but we can find a large grid whose diagonal has disjoint paths leading into an assumed node-well-linked set.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Problems, reductions and completeness

Keywords and phrases Conjunctive queries, graph homomorphisms, counting complexity, parameterized complexity, fine-grained complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.113

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version A full version of this work appears as preprint arXiv:1902.04960.

Funding *Philip Wellnitz*: Partially funded by the Saarbrücken Graduate School of Computer Science.

Acknowledgements We thank Cornelius Brand, Karl Bringmann, Radu Curticapean, Reinhard Diestel, Joshua Erde, Stephan Kreutzer, Stefan Mengel, Daniel Weißauer, and an anonymous reviewer for discussions and advice.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 113; pp. 113:1–113:15

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

1 Introduction

Conjunctive query evaluation is a core problem in database theory. Using first-order logic, conjunctive queries can be expressed by formulas of the form

$$x_1 \dots x_k \exists y_1 \dots \exists y_\ell (a_1 \wedge \dots \wedge a_m)\,, \tag{1}$$

where the x_i are the *free variables*, the y_i are the (existentially) *quantified variables*, and the a_i are *atomic formulas* (such as edge $E(x_1, y_4)$ or relational $R(x_7, y_3, y_6)$ constraints on the variables). Conjunctive queries exactly correspond to select-project-join queries; a detailed introduction can be found in the textbook of Abiteboul, Hull, and Vianu [1]. The *conjunctive query evaluation problem* is given a conjunctive query and a relational database, and is tasked to compute the set of all assignments to the free variables such that the formula is satisfied. Since enumerating all solution tuples $s_1 \dots s_k$ can be costly for reasons not inherent to the problem's complexity, it is more meaningful to consider the decision problem (Does there exist a solution tuple?) or the more general counting problem (How many solution tuples exist?). The decision problem is equivalent to setting $k = 0$ and also called the constraint satisfaction problem (CSP). In this paper, we study the problem of counting the number of all solution tuples for conjunctive and more general queries.

Perhaps the most naïve way to study the complexity of this problem is via its *combined complexity*, in which both the query and the database are considered to be worst-case inputs. Since conjunctive queries generalize the clique problem on graphs, the problem is clearly NP-hard in this setting [2]. In the real world, however, the database is much larger than the query, and thus the combined complexity may fixate on instances that we do not care about. Instead, we consider two other models in this paper: the data complexity and the parameterized complexity of conjunctive query evaluation.

The *data complexity* considers the query to be completely fixed and only the database to be worst-case input. If the query is fixed, the number of variables $k + \ell$ is a constant, and so the problem is polynomial-time solvable: even the exhaustive search algorithm just needs to try out and check all $n^{k+\ell}$ possible assignments to the variables, where n is the size of the universe. Unsurprisingly, exhaustive search is not the best strategy for every query. For example, Chekuri and Rajaraman [3] showed that the decision and counting problems can be solved in time $O(n^{t+1})$ where t is the treewidth of the query's Gaifman graph, that is, the graph containing a vertex for every variable and an edge between two vertices whenever the corresponding variables are contained in a common constraint. Since $t + 1$ is typically much smaller than $k + \ell$, this algorithm is better than exhaustive search. For each fixed query Q, the guiding question for a fine-grained understanding of data complexity is this: What is the smallest constant c_Q such that the query evaluation problem can be solved in time $O(n^{c_Q})$?

Parameterized complexity offers a third vantage point from which conjunctive query evaluation can be studied. Here the query isn't completely fixed, but it's also not completely free either. Instead, it is assumed that only certain types of queries will be used, meaning that the class of queries that are allowed as input is restricted. As a hybrid between data complexity and combined complexity, the parameterized complexity of query evaluation is more difficult to study than the combined complexity, but easier than the data complexity, while still offering some insight. For example, Grohe, Schwentick, and Segoufin [18] show that the conjunctive query evaluation problem is W[1]-hard if the class of allowed input queries has Gaifman graphs of unbounded treewidth.

1.1 Context and Previous Work

When only one constraint type E of arity two is allowed, the conjunctive query evaluation problem specializes to the graph homomorphism problem: The decision problem (where $k = 0$) is given two graphs H, G to decide whether there is a homomorphism from H to G. Dalmau et al. [9] prove that this problem can be solved in polynomial time if the homomorphic core of H has bounded treewidth, and conversely, Grohe [17] shows that the graph homomorphism problem is W[1]-complete even if H is restricted to be from an arbitrary class of graphs whose homomorphic cores have unbounded treewidth. Taken together, these two results yield a *dichotomy theorem* for the complexity of detecting graph homomorphisms: Depending on the class of allowed graphs H, the problem is either polynomial-time computable or W[1]-complete, and in particular there are not infinitely many cases of intermediate complexity. For the counting problem without quantified variables (where $\ell = 0$), such a dichotomy is also known: Dalmau and Jonsson [8] show that the number of homomorphisms from H to G is polynomial-time computable if H itself has bounded treewidth, and it is #W[1]-complete if H comes from any class of unbounded treewidth. In the mixed situation when both free and quantified variables may exist (and thus $k, \ell > 0$), then the resulting counting problem actually counts *partial homomorphisms*, that is, homomorphisms from k vertices of H that can be extended to a homomorphism on all $k + \ell$ vertices of H. A line of work [27, 25], culminating in Durand and Mengel [12] and Chen and Mengel [4], studies the parameterized complexity of this mixed problem, and depending on the class of graphs H that are allowed, they classify the complexity either as polynomial-time, W[1]-equivalent, or #W[1]-hard. A corollary to the present work is a finer classification that splits up the #W[1]-hard cases into three classes.

One way to go beyond homomorphisms is to consider *injective* homomorphisms, which leads to the corresponding decision problem that is given H, G to decide whether H is a subgraph of G – this problem can be solved in time $f(H)n^{O(t)}$ if t is the treewidth of H (e.g., [15]), that is, it is *fixed-parameter tractable* when parameterized by $|H|$ and if the treewidth is bounded. However, it is an important open problem [24] whether the subgraph detection problem is W[1]-hard when H is restricted to be from an arbitrary class of unbounded treewidth. The counting problem is better understood: Vassilevska Williams and Williams [33] (also cf. [21, 7]) show that the number of times H occurs as a subgraph in G can be computed in time $f(H)n^{\mathsf{vc}(H)+O(1)}$ where $\mathsf{vc}(H)$ is the size of the smallest vertex cover, but Curticapean and Marx [7] (also cf. [6]) show that the problem is #W[1]-complete if H is from any class of graphs whose minimum vertex cover is not bounded. Now, what do injective homomorphisms have to do with conjunctive queries? As it turns out, what we are doing is to add *inequalities* as an additional, but very restricted constraint type: Injective homomorphisms correspond to queries without quantified variables that have edge constraints and are augmented with inequalities $(x_i \neq x_j)$ for all distinct i, j. If some, but not all, inequality constraints are present, we obtain partially injective homomorphisms, the complexity of which has a known dichotomy theorem for the counting version [30], and has been studied to some extent for the decision version [20]. As part of the present work, we are able to classify the mixed situation with free and quantified variables $(k, \ell > 0)$ *as well as* some inequalities on the free variables.

The mentioned complexity classification for counting partial homomorphisms into three cases [12, 4] was actually proved in the more general setting of conjunctive queries. Chen and Mengel [5] extended their classification to queries that are monotone, but not necessarily conjunctive. That is, the corresponding formula is supposed to be an existential positive formula, which may contain existential quantifiers $\exists$, logical ands $\wedge$, and ors $\vee$.

In the present work, we are able to further extend (our finer version of) the classification to existential formulas that may have negations on constraints involving only free variables; we truly study the complexity of *counting answers to existential questions*.

1.2 Our Contributions

As already indicated in Section 1.1, we make simultaneous progress on two fronts: Our complexity classifications are finer than previous work, and we can prove the classification for more general classes of queries. An important feature of our work is that the proofs are modular and largely self-contained: We first prove the complexity results for counting partial homomorphisms, then lift them to conjunctive queries, and then further to a more general class of queries. So what is the most general class of queries that we study? We allow queries φ of the form

$$x_1 \dots x_k \exists y_1 \dots \exists y_\ell : \psi\,, \tag{2}$$

where ψ is a quantifier-free formula in first-order logic and all negations in ψ must be directly applied to constraints that only involve free variables (e.g. $E(x_1, x_7) \vee (R(x_7, y_7, y_9) \wedge \neg R(x_1, x_4, x_9))$). Constraints of the form $\neg R(x_1, x_4, x_9)$ are referred to as *non-monotone constraints* in the remainder of the paper. Furthermore φ may be equipped with a set of inequalities over the free variables (eg. $x_3 \neq x_5$).

All of our theorems also apply to the corresponding *universal* queries, where each $\exists$ in (2) is replaced with $\forall$, but for the sake of readability we will often omit this fact. We are able to generalize from conjunctive queries to queries of the form (2) by using ideas that go back to Lovász's work from 1967 [22] (also cf. [23]): We prove that queries φ of the form (2) can be expressed in a meaningful way as an abstract linear combination of conjunctive queries (which are of the form (1)); positive results (algorithms) as well as negative results (hardness) for each "summand" translate to the abstract linear combination and thus to φ.

Data Complexity

To study the data complexity of the problem, we employ the Strong Exponential Time Hypothesis (SETH) by Impagliazzo and Paturi [19], which was developed in the context of fine-grained complexity. The k-dominating set problem can be easily expressed as a (universal) conjunctive query, and Williams and Pătraşcu [28] show that this problem cannot be solved in time $O(n^{k-\varepsilon})$ unless SETH is false. We are able to lift this hardness result to all queries φ that have the k-dominating set query as a *query minor*, a notion that we translate from graphs and formalize later. The *dominating star size* $\mathsf{dss}(\varphi)$ of a conjunctive query φ is the maximum number k such that the k-dominating set query is a query minor. Equivalently, this means that some connected component in the quantified variables of φ has k neighbors in the free variables.[1] We obtain the following result:

▶ **Theorem 1.** *Let φ be a fixed query of the form (2). Given a logical structure B with a domain of size n, we wish to compute the number of solutions of φ in B. If SETH holds, this problem cannot be solved in time $O(n^{\mathsf{dss}(\varphi)-\varepsilon})$ for any $\varepsilon > 0$.*

In the full version, we also obtain an algorithm for the problem in Theorem 1, with a running time of $O(n^{\mathsf{dss}(\varphi)+t+1} + n^{t'+1})$, where t and t' are treewidths related to the query φ. Neglecting many technical details, the proof of Theorem 1 reduces the k-dominating set

[1] The dominating star size coincides with the *strict star size* from [4].

problem to the model counting problem for φ by following operations of the query minor. If φ is a query of the form (2), then it can be represented by an abstract linear combination of conjunctive queries φ'; in this case, we define $\mathsf{dss}(\varphi)$ as the maximum $\mathsf{dss}(\varphi')$ over all constituents φ' that occur in this abstract linear combination.

Theorem 1 is similar in spirit to other known conditional lower bounds for first-order model checking, such as the one of Williams [32] and Gao et al. [16]. One of their results is that first-order sentences with $k+1$ variables cannot be decided in time $O(m^{k-\varepsilon})$, where m is the size of the structure, unless SETH fails. However, these results are incomparable to Theorem 1 for several reasons: The results in [32, 16] allow negations and consider the decision problem, while we allow only limited negations and consider the counting problem. More fundamentally, however, Theorem 1 gives a hardness result for every fixed query φ, while the results in [32, 16] show that there exists a query φ that is hard. Moreover, the lower bounds in [32, 16] are in terms of the size m of the structure, not merely the size n of the domain.

Parameterized Complexity

We refine the classification of Chen and Mengel [4] for counting answers to conjunctive queries. For every class of allowed queries they show the problem to be either fixed-parameter tractable, W[1]-equivalent or #W[1]-hard. Here, W[1]-equivalent means that there are parameterized Turing reductions from and to the decision version of the k-Clique problem. Understanding the parameterized complexity of problems even beyond the usual classes W[1] and #W[1] is interesting from a structural complexity point of view, and it also provides meaningful information about the studied problem. Indeed we show that the dominating star size, i.e., the parameter considered in Theorem 1, is a structural parameter for conjunctive queries that, if unbounded, makes the problem #W[2]-hard and that, if bounded, keeps the problem #W[1]-easy.

This extension to #W[2]-hard cases only partially resolves the parameterized complexity of the problem of counting answers to conjunctive queries. It is known that the general problem of counting answers to formulas of the form

$$x_1 \dots x_k \exists y_1 \dots \exists y_\ell : \psi\,, \qquad \text{where } \psi \text{ is a quantifier-free first-order formula,} \tag{3}$$

is #A[2]-equivalent.[2] For which families of *conjunctive* queries is the counting problem as hard as for unrestricted queries as in (3)? Such families have the hardest counting problems, even harder than the #W[2]-hard cases unless #A[2] = #W[2] holds, which seems unlikely.[3] We prove that families of conjunctive queries are #A[2]-hard if their *linked matching number* is unbounded. Intuitively a conjunctive query φ with free variables X and quantified variables Y has a large linked matching if there is a large well-linked set in Y that cannot be separated from X by removing a small number of variables. We obtain the following refined complexity classification.

[2] Due to a technicality in the original definition of #A[2], we cannot establish #A[2]-completeness and will instead only talk about *equivalence* to a #A[2]-complete problem under parameterized Turing reductions.

[3] See Chapt. 8 and 14 in [14] for a discussion.

▶ **Theorem 2.** *Let Φ be a family of conjunctive queries. Given a formula φ from Φ and a logical structure B, we wish to compute the number of solutions of φ in B. When parameterized by $|\varphi|$ this problem is*

1. #W[1]*-easy if the dominating star size of Φ is bounded,*
2. #W[2]*-hard if the dominating star size of Φ is unbounded, and*
3. #A[2]*-equivalent if the linked matching number of Φ is unbounded.*

It is instructive to provide examples for the application of the above theorem. First consider the problem of, given a graph G without self-loops and a natural number k, computing the number of cliques of size k that are not maximal. While the problem of counting cliques of size k is #W[1]-complete, adding the non-maximality constraint makes the problem hard for #W[2]. To see this, we will express the problem as a conjunctive query

$$\varphi_k := x_1 \dots x_k \exists y : \bigwedge_{1 \leq i < j \leq k} E(x_i, x_j) \wedge \bigwedge_{1 \leq i \leq k} E(x_i, y)\,. \tag{4}$$

Note that the number of solutions to φ_k in G is precisely $k!$ times the number of non-maximal cliques of size k in G. Furthermore, it holds that φ_k has dominating star size k and hence that $\Phi = \{\varphi_k \mid k \in \mathbb{N}\}$ has unbounded dominating star size. By Theorem 2 the problem of counting answers to queries in Φ is #W[2]-hard. Furthermore, invoking Theorem 1, we obtain that counting non-maximal cliques of size k cannot be done in time $O(n^{k-\varepsilon})$ for any $\varepsilon > 0$. Note that this is also in sharp contrast to the problem of counting (not necessarily non-maximal) cliques of size k which can be done in time $O(n^{\omega k/3})$ [26]. Furthermore deciding the existence of a non-maximal clique of size k is equivalent to deciding the existence of a clique of size $k+1$ and hence the lower bound under SETH crucially depends on the fact that we count the solutions.

On the other hand, counting non-maximal cliques of size k is most likely not #A[2]-hard as it is #W[2]-easy[4]. An example for a #A[2]-hard problem would be the following. Assume a graph G and a natural number k are given. Then the goal is to compute the number of k-vertex sets that can be (perfectly) matched to a k-clique. Let us express the problem as a conjunctive query

$$\psi_k := x_1 \dots x_k \exists y_1 \dots \exists y_k : \bigwedge_{1 \leq i < j \leq k} E(y_i, y_j) \wedge \bigwedge_{1 \leq i \leq k} E(x_i, y_i)\,. \tag{5}$$

We point out that ψ_k does not correspond directly to the vertex sets we would like to count as x_i and x_j could be the same vertex in G. However, it can be shown that an oracle for counting answers to ψ_k allows us to compute the desired number efficiently and vice versa. Finally, as the linked matching number of ψ_k is not bounded for $k \to \infty$, #A[2]-hardness follows from Theorem 2.

Building up on Theorem 2 and using the framework of linear combinations, we obtain the following, extensive classification result.

[4] If there is a constant bound on the number of quantified variables then the problem of counting answers to conjunctive queries is reducible to a #W[2]-complete problem w.r.t. parameterized Turing reductions. We omit a proof of this statement but point out that it can be done by lifting the results of Chapt. 7.4 in [14] to the realm of counting problems.

▶ **Theorem 3.** *Let Φ be a family of existential or universal positive formulas with inequalities and non-monotone constraints, both over the free variables. Given a formula φ from Φ and a logical structure B, we wish to compute the number of solutions of φ in B.*

When parameterized by $|\varphi|$, this problem is either fixed parameter tractable, W[1]*-equivalent,* #W[1]*-equivalent,* #W[2]*-hard or* #A[2]*-equivalent.*

Note that allowing the inequalities and non-monotone constraints over all variables, not just the free ones, would in particular include the subgraph decision problem. However, the parameterized complexity of finding a subgraph in G that is isomorphic to a small pattern graph P is a long-standing open question in parameterized complexity [11, Chapt. 33.1].

1.3 Techniques and Overview

Our paper brings together questions and techniques from a wide variety of areas, such as parameterized and fine-grained complexity, logics, database theory, matroid theory, lattice theory, graph minor theory, and the theory of graph limits. The interested reader should not be alarmed, however, as we put considerable effort into making the presentation as self-contained and smooth as possible, introducing the required background material carefully and only once needed in both, the extended abstract as well as in the full version this paper: After reviewing some basic preliminaries (Section 2), we formally present our refined complexity classifications in Section 3 for the special case of partial graph homomorphisms, rather than the full query evaluation problem. Due to the incompatibility of the space constraints and the amount of results and techniques required, we deferred all proofs as well as the treatment of the full query evaluation problem over arbitrary signatures of bounded arity to the full version of this paper.

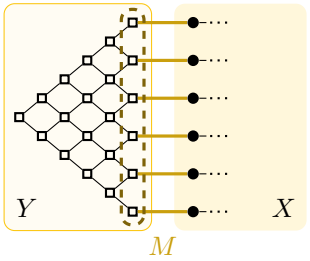
Colors and Query Minors

We will mainly work with a color-prescribed variant of the problem of counting answers to conjunctive queries. Here we assume that the elements of a given database B are colored according to the variables of the given conjunctive query φ and the goal is to compute the number of solutions that are additionally color-preserving. For this variant we will show and heavily exploit that the problem of counting answers to a conjunctive query φ is at least as hard as counting answers to any query that is a minor of φ. Minors of a query are defined via the (graph theoretic) minors of its Gaifman graph. It is then required to show that the color-prescribed variant and the uncolored variant are interreducible for all minimal queries. Intuitively, a query is minimal if it does not contain a proper subquery that produces the same set of solutions for each database. The proof of the interreducibility relies on the theory of homomorphic equivalence.

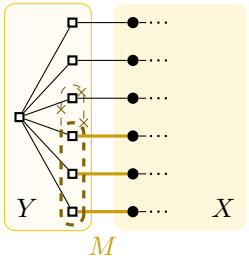
For Theorem 1 and the second case of Theorem 2 we construct a tight reduction from the problem of counting dominating sets of size k which cannot be solved in time $O(n^{k-\varepsilon})$ for any $\varepsilon > 0$ unless SETH fails [28] and which is hard for #W[2] [13].

Minor Theory

For #A[2]-hardness in Theorem 2 we take a detour to graph minor theory: Given a graph G, we call a set $S \subseteq V(G)$ *node-well-linked* if, for every pair of disjoint, equal-sized subsets A, B of S, there are $|A| = |B|$ vertex disjoint paths in G that connect the vertices in A with the vertices in B. Now, we obtain the following strengthening of the Excluded-Grid-Theorem, which might be of independent interest.



(a) A query (H, X) with a large linked matching number ($\mathsf{lmn}(H, X) = 6$): There is a large matching M (gold-colored) connecting vertices in X with a node-well-linked set in Y (enclosed within the dashed line).



(b) A query (H, X) with a small linked matching number ($\mathsf{lmn}(H, X) = 3$): While there may be larger matchings between vertices of X and Y, a largest matching M into a node-well-linked set in Y has size 3.

Figure 1 Examples for queries with large and small linked matching number.

▶ **Theorem 4** (Intuitive version). *There exists an unbounded function f such that every graph with a node-well-linked set S of k vertices has an $(f(k) \times f(k))$-grid minor with the property that there are $f(k)$ vertex-disjoint paths leading from S to the $f(k)$ vertices of the first column of the grid and without touching the grid minor elsewhere.*

If we drop the requirement that the minor model can be reached by disjoint paths from S, then this theorem is well-known and due to Diestel et al. [10].

Intuitively, we use Theorem 4 in the following way: If the quantified variables of a query contain a node-well-linked set S, we obtain a large grid-like structure that is connected to S by many vertex-disjoint paths. Next, we show that if that set S also has a large matching to a subset of the free variables of the query, then the query becomes #A[2]-hard. For this last step, we use an #A[2]-normalization theorem, which we will provide at the end of the paper.

Formally, we define the *linked matching number* of a query and prove #A[2]-hardness if this parameter is unbounded. Consider Figure 1 for examples for the linked matching number.

▶ **Definition 5** (Linked matching number). *Let (H, X) be graphical conjunctive query, let $Y = V(H) \setminus X$ be the set of quantified variables, and let M be a matching from X to Y. We call the matching M* linked *if the set $V(M) \cap Y$ is node-well-linked in $H[Y]$. The* linked matching number lmn *of (H, X) is defined as the size of the largest linked matching of H.*

Abstract Linear Combinations

To prove Theorem 3, we use abstract linear combinations that are called *quantum graphs* (or rather, quantum queries in our setting) and were developed in the theory of graph limits [23]. For our computational questions, the *complexity monotonicity property* [6] is the useful phenomenon that the quantum graph and its constituents (i.e., its abstract summands) often lead to computational problems that have precisely the same complexity. Using elementary linear-algebraic and polynomial interpolation arguments, we prove that this property holds, and we use Rota's NBC Theorem from lattice theory [29] to determine which graphs are constituents of the relevant quantum graphs. The complexity monotonicity property has been used (implicitly) by Chen and Mengel [5] for their extension from conjunctive queries to monotone queries; and the extension from homomorphisms to partially injective homomorphism [30] used Rota's Theorem in a similar fashion as we do in the present work.

2 Preliminaries

We use the notation $[n] = \{1, \ldots, n\}$ and $[m, n] = \{m, \ldots, n\}$ for natural numbers with $m < n$. We write $\#M$ for the cardinality of a finite set M. We write $f|_M$ for the restriction of a function f to elements of M. For a function $f\colon A \times B \to C$ and $a \in A$, we write $f(a, \star)$ for the function $b \mapsto f(a, b)$.

Graphs, Homomorphisms, and Formulas

Graphs in this paper are unlabeled, undirected, simple and without self-loops, unless stated otherwise. Let $V(G)$ denote the set of vertices and $E(G)$ denote the set of edges of G. We define the *size* of a graph G to be the number of vertices. Given a subset Y of $V(G)$, we write $G[Y]$ for the subgraph induced by the vertices of Y. The *complement graph* $\overline{G}$ has the same vertices as G and contains an edge uv if and only if $u \neq v$ and $uv \notin E(G)$. A *homomorphism* h from a graph F to a graph G is a mapping from $V(F)$ to $V(G)$ that is edge-preserving, that is, all $uv \in E(F)$ satisfy $h(u)h(v) \in E(G)$. We write $\mathsf{Hom}(F \to G)$ for the set of all homomorphisms from F to G. A bijective homomorphism whose inverse is also a homomorphism is called an *isomorphism*, and a homomorphism from F to F itself is called *endomorphism*. An endomorphism that is also an isomorphism is called an *automorphism*. We write $\mathsf{Aut}(F)$ for the set of all automorphisms of F.

Parameterized Counting Complexity

A *counting problem* is a function $P\colon \{0,1\}^* \to \mathbb{N}$, and a *parameterized counting problem* is a pair (P, π) where $\pi\colon \{0,1\}^* \to \mathbb{N}$ is computable and called a *parameterization*. Parameterized *decision* problems are defined likewise for decision problems $P\colon \{0,1\}^* \to \{0,1\}$. A parameterized (decision or counting) problem is *fixed-parameter tractable* if there is a computable function $t\colon \mathbb{N} \to \mathbb{N}$ such that, for every input $x \in \{0,1\}^*$, the function P can be computed in time $t(\pi(x)) \cdot \mathsf{poly}(|x|)$. We denote the class of all fixed-parameter tractable problems as FPT.

A *parameterized Turing-reduction* from (P, π) to (P', π') is an algorithm $\mathbb{A}$ with oracle access to P' that solves P, such that $\mathbb{A}$ runs in fixed-parameter tractable time when parameterized by π and there exists a computable function r such that, for every input x, the parameter $\pi'(y)$ of every query y is bounded by $r(\pi(x))$. A *parameterized parsimonious reduction* is a parameterized Turing-reduction with the additional requirement that $\mathbb{A}$ is only allowed to query the oracle a single time at the very end of the computation and then outputs the result of the query without further modification.

Clique is the parameterized (decision) problem to decide whether a given graph G contains a k-clique. Similarly, DomSet is to decide whether G has a dominating set of size k. The parameterized counting problems #Clique and #DomSet count the number of the respective objects. We define the parameterized complexity classes that appear in this paper by their well-known complete problems: W[1] contains all parameterized problems that are reducible to Clique with respect to parameterized parsimonious reductions. Similarly, #W[1], W[2], and #W[2] contain all problems reducible to #Clique, DomSet, and #DomSet, respectively. Furthermore #A[2] is the class of all parameterized counting problems that are expressible as model counting problem with one quantifier alternation. It is known that

$$\text{FPT} \leq^T \text{W[1]} \leq^T \#\text{W[1]} \subseteq \#\text{W[2]} \subseteq \#\text{A[2]}\,,$$

where $\text{C} \leq^T \text{D}$ denotes that every problem in C can be reduced to a problem in D with respect to parameterized Turing-reductions.

(a) **(b)**

Figure 2 (a) Graphical representation of the conjunctive query in (6). **(b)** A graphical conjunctive query that is "equivalent" to the example on the left in the sense that an assignment $a\colon \{x_1, \ldots, x_k\} \to V(G)$ is a partial homomorphism from the left graph to G if and only if it is a partial homomorphism from the right graph to G.

For further background on parameterized counting complexity, see [14, Chapter 14]. While the parameterized complexity classes are defined via parsimonious reductions, we will rely on Turing reductions. Hence we cannot speak of completeness but instead of equivalence.

▶ **Definition 6.** *Let* C *be a parameterized complexity class. A parameterized counting problem* (P, π) *is* C-easy *if it can be reduced to a problem in* C *and it is* C-hard *if every problem in* C *reduces to* (P, π)*, both with respect to parameterized Turing-reductions. A problem is* C-equivalent *if it is* C-easy *and* C-hard*.*

Exponential-Time Hypotheses

The strong exponential time hypothesis (SETH) asserts that for all $\delta > 0$ there is some $k \in \mathbb{N}$ such that k-SAT cannot be computed in time $O(2^{(1-\delta)n})$, where n is the number of variables of the input formula [19]. A dominating set of size k in an n-vertex graph cannot be computed in time $O(n^{k-\varepsilon})$ for any $\varepsilon > 0$ unless SETH is false [28]. The exponential time hypothesis (ETH) asserts that 3-SAT cannot be computed in time $\exp(o(m))$, where m is the number of clauses of the input formula [19].

3 Formal Statements of Our Results

It is instructive to first focus on conjunctive queries with one relation symbol E of arity two. An example of such a query is the following formula:

$$x_1 \ldots x_k \exists y : Ex_1y \wedge \cdots \wedge Ex_ky\,. \tag{6}$$

The relation E corresponds to a graph G and the free and quantified variables will be assigned vertices of G. In this example, an assignment $a_1, \ldots, a_k \in V(G)$ to the free variables satisfies the formula if and only if the vertices $a_1, \ldots, a_k$ have a common neighbor in G. It will be convenient for us to view the formula as a graph H as depicted in Figure 2. The vertices of H are partitioned into a set $X = \{x_1, \ldots, x_k\}$ of free variables and a set $Y = \{y\}$ of quantified variables. An assignment to the free variables corresponds to a function $a : X \to V(G)$, and such an assignment satisfies the formula if it can be consistently extended to a homomorphism from H to G. This motivates the following definition, where we only consider simple graphs without loops, so we do not allow atomic subformulas of the form Ezz.

▶ **Definition 7.** *A* graphical conjunctive query (H, X) *consists of a graph* H *and a set* X *of vertices of* H*. We let* $\mathsf{Hom}(H, X \to G)$ *be the set of all mappings from* X *to* $V(G)$ *that can be extended to a homomorphism from* H *to* G*, and we call these mappings partial homomorphisms. Formally, the set of partial homomorphisms is defined via*

$$\mathsf{Hom}(H, X \to G) = \left\{ a : X \to V(G) \;\middle|\; \exists h \in \mathsf{Hom}(H \to G) : h|_X = a \right\}. \tag{7}$$

Given two different graphical conjunctive queries (H, X) and $(\hat{H}, \hat{X})$ it might be the case that $\#\mathsf{Hom}(H, X \to \star)$ and $\#\mathsf{Hom}(\hat{H}, \hat{X} \to \star)$ are the same functions. An example for this is given in Figure 2. In this case, we say that (H, X) and $(\hat{H}, \hat{X})$ are *equivalent*, denoted as $(H, X) \sim (\hat{H}, \hat{X})$, and the subgraph-minimal elements of the induced equivalence classes are called *minimal*. In what follows, we classify the complexity of counting homomorphisms for classes of graphical conjunctive queries. More precisely, we consider the parameterized counting problem $\#\mathsf{Hom}(\Delta)$ for each fixed class Δ of graphical conjunctive queries. This problem is given as input a query $(H, X) \in \Delta$ and a graph G and the task is to compute the number $\#\mathsf{Hom}(H, X \to G)$. The problem is parameterized by the size of H. We start with the formal definitions of the different structural parameters of graphical conjunctive queries and present the classification theorem thereafter. All parameters, along with five example classes, are depicted in Figure 3.

▶ **Definition 8** (Contract). *The* contract *of a graphical conjunctive query (H, X) is a graph on the vertex set X, obtained by adding an edge between two vertices u and v in X if uv is an edge of H or if there exists a connected component C in $H \setminus X$ that is adjacent to both u and v. Given a class Δ of conjunctive queries, we write $\mathsf{contract}(\Delta)$ for the set of all of its contracts.*

▶ **Definition 9** (Dominating star size). *Let (H, X) be graphical conjunctive query and let $Y_1, \ldots, Y_\ell$ be the connected components of the subgraph $H[V(H) \setminus X]$ induced by the quantified variables. Further, let k_i be the number of vertices $x \in X$ for which there exists a vertex $y \in Y_i$ that is adjacent to x. The* dominating star size *of (H, X) is defined via*

$$\mathsf{dss}(H, X) = \max\{\, k_i \mid i \in \ell \,\}.$$

We are now in position to state our main result, the full classification for counting answers to conjunctive queries. Note that Theorem 2 is subsumed by the full classification in the case of graphs. The general version, that is, the case of arbitrary logical signatures with bounded arity, is stated and proved in the full version.

▶ **Theorem 10.** *Let Δ be a recursively enumerable class of minimal conjunctive queries.*

1. *If the treewidth of Δ and $\mathsf{contract}(\Delta)$ is bounded, then $\#\mathsf{Hom}(\Delta)$ can be computed in polynomial time.*
2. *If the treewidth of Δ is unbounded and the treewidth of $\mathsf{contract}(\Delta)$ is bounded, then $\#\mathsf{Hom}(\Delta)$ is $\mathrm{W}[1]$-equivalent.*
3. *If the treewidth of $\mathsf{contract}(\Delta)$ is unbounded and the dominating star size of Δ is bounded, then $\#\mathsf{Hom}(\Delta)$ is $\#\mathrm{W}[1]$-equivalent.*
4. *If the dominating star size of Δ is unbounded, then $\#\mathsf{Hom}(\Delta)$ is $\#\mathrm{W}[2]$-hard. Moreover, for any fixed query δ with $\mathsf{dss}(\delta) \geq 3$, the problem $\#\mathsf{Hom}(\delta \to \star)$ cannot be computed in time $O(n^{\mathsf{dss}(\delta)-\varepsilon})$ for any $\varepsilon > 0$ unless SETH fails.*
5. *If the linked matching number of Δ is unbounded, then $\#\mathsf{Hom}(\Delta)$ is $\#\mathrm{A}[2]$-equivalent.*

In proving the last case of Theorem 10, we establish the following generalization of the Excluded-Grid-Theorem which applies for conjunctive queries with a large linked matching number and is essentially equivalent to Theorem 4; consult Figure 3 for the notion of a *grate*.

▶ **Theorem 11.** *Let Δ be a class of graphical conjunctive queries. If the linked matching number of Δ is unbounded, then Δ contains arbitrarily large grates as minors.*

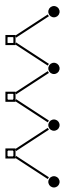
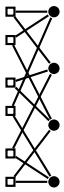
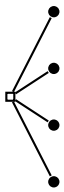
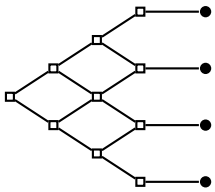
Query Classes	Δ_{poly}	$\Delta_{\mathrm{W}[1]}$	$\Delta_{\#\mathrm{W}[1]}$	$\Delta_{\#\mathrm{W}[2]}$	$\Delta_{\#\mathrm{A}[2]}$
Query for $k = 4$					
contract for $k = 4$		$\emptyset$			
tw	$O(1)$	∞	∞	$O(1)$	∞
tw(contract)	$O(1)$	$O(1)$	∞	∞	∞
dss	$O(1)$	$O(1)$	$O(1)$	∞	∞
lmn	$O(1)$	$O(1)$	$O(1)$	$O(1)$	∞
Complexity	P	W[1]-eq.	#W[1]-eq.	#W[2]-hard	#A[2]-eq.$^{(*)}$

Figure 3 Example problems for each case of the complexity classification (Theorem 10):

$$\Delta_{\mathrm{poly}} = \{\, \varphi_k \mid k \in \mathbb{N} \,\}, \quad \text{where } \varphi_k := x_1 \dots x_k \; \exists y_1 \dots \exists y_{k-1} : \bigwedge_{1 \le i < k} Ex_iy_i \wedge Ey_ix_{i+1}$$

$$\Delta_{\mathrm{W}[1]} = \{\, \psi_k \mid k \in \mathbb{N} \,\}, \quad \text{where } \psi_k := \exists y_1 \dots \exists y_k : \bigwedge_{1 \le i < j \le k} Ey_iy_j$$

$$\Delta_{\#\mathrm{W}[1]} = \{\, \upsilon_k \mid k \in \mathbb{N} \,\}, \quad \text{where } \upsilon_k := x_1 \dots x_k : \bigwedge_{1 \le i < j \le k} \exists y_{ij} : Ex_iy_{ij} \wedge Ey_{ij}x_j$$

$$\Delta_{\#\mathrm{W}[2]} = \{\, \delta_k \mid k \in \mathbb{N} \,\}, \quad \text{where } \delta_k := x_1 \dots x_k \exists y : \bigwedge_{1 \le i \le k} Ex_iy$$

Furthermore, $\Delta_{\#\mathrm{A}[2]}$ is the set of all *grates*. Here, a k-grate is the conjunctive query whose quantified variables constitute half of a $k \times k$ grid whose diagonal is connected to k free variables by a matching of size k. The formal definition is given in the full version.
Depicted is the query (H, X) for $k = 4$, where free variables (i.e., vertices in X) are drawn as solid discs and quantified variables (i.e., vertices in $V(G) \setminus X$) are drawn as hollow squares. We also display the contract (see Definition 8) of each query. We write $O(1)$ whenever a parameter is bounded by a constant in the entire query class, and ∞ whenever it is unbounded. Finally, we show the complexity of counting answers to conjunctive queries in each of the classes in terms of polynomial-time tractability (P) and equivalence (-eq.) or hardness for one of the parameterized complexity classes.

$(*)$ The observant reader might have noticed that a k-grate is not a minimal conjunctive query. For this reason, #A[2]-equivalence in the last column refers to minimal conjunctive queries that contain arbitrary large grates as minors. Alternatively, #A[2]-equivalence is shown to hold for k-grates in the color-prescribed case. Details are given in the full version.

Building upon Theorem 10, we invoke the complexity monotonicity property of linear combinations of conjunctive queries to classify the complexity of counting answers to queries of the form (2), that is, existential or universal first-order queries for which we allow inequalities and non-monotone constraints, both over the free variables. Specifically, we prove the classification as given by Theorem 3. Again, we refer to the full version for the general case of arbitrary logical structures of bounded arity. We also discuss conjunctive queries that may contain inequalities over the free variables. Note that in that case, we are able to give explicit criteria for the five different cases in Theorem 3.

4 Conclusions

We established a comprehensive classification of the complexity of counting answers to conjunctive queries and linear combinations thereof. Depending on the structural parameters of the class of allowed queries, the problem is either polynomial-time solvable, W[1]-equivalent, #W[1]-equivalent, #W[2]-hard or #A[2]-equivalent. This classification, however, leaves out a gap between the latter two cases. More precisely, the following question remains open:

Does a class of conjunctive queries Δ exist for which $\#\mathsf{Hom}(\Delta)$ is $\#\mathrm{W}[2]$-hard but neither equivalent for $\#\mathrm{W}[2]$ nor for $\#\mathrm{A}[2]$?

We conjecture a positive answer; the interested reader is encouraged to make themself familiar with the parameterized complexity class $\mathsf{W}_{\mathsf{func}}[2]$ (see e.g. [14, Chapter 8.8]). This class has a canonical counting version which we call $\#\mathsf{W}_{\mathsf{func}}[2]$ and which interpolates between $\#\mathrm{W}[2]$ and $\#\mathrm{A}[2]$. In particular, we conjecture that there exists a class of conjunctive queries Δ for which $\#\mathsf{Hom}(\Delta)$ is $\#\mathsf{W}_{\mathsf{func}}[2]$-equivalent. Consequently, a negative answer to the previous question would imply that either $\#\mathsf{W}_{\mathsf{func}}[2] = \#\mathrm{W}[2]$ or $\#\mathsf{W}_{\mathsf{func}}[2] = \#\mathrm{A}[2]$, which seems to be very unlikely (see e.g. the discussion of $\mathsf{W}_{\mathsf{func}}[2]$ in [14, Chapter 8.8]).

A further question that remains open, and which should be considered a stronger version of the previous question, reads as follows:

Does a class of conjunctive queries Δ exist such that Δ has bounded linked matching number and the problem $\#\mathsf{Hom}(\Delta)$ is $\#\mathrm{A}[2]$-equivalent?

In other words, the above question asks whether the absence of a bound on the linked matching number is not only sufficient, but also necessary for $\#\mathrm{A}[2]$-equivalence. In contrast to the previous question, we conjecture a negative answer. Let us provide some intuition for the latter conjecture: It seems that a constant bound on the linked matching number of a class of conjunctive queries Δ yields a separator decomposition of the quantified variables of queries in Δ in components that have either small treewidth or a small matching number to the free variables. We conjecture that such a decomposition implies the existence of what is called a κ-restricted nondeterministic Turing machine $\mathbb{M}$ such that the number of accepting paths of $\mathbb{M}$ on input $(H, X) \in \Delta$ and a graph G is precisely $\#\mathsf{Hom}(H, X \to G)$ (see e.g. [14, Definition 14.15]). If additionally $\#\mathsf{Hom}(\Delta)$ is $\#\mathrm{A}[2]$-equivalent, this would imply that the set of $\#\mathrm{A}[2]$-equivalent problems is a subset of the set of $\#\mathsf{W}[\mathsf{P}]$-equivalent problems; consult e.g. [14, Chapter 3 and 14.2] for a treatment of the class $\#\mathsf{W}[\mathsf{P}]$. However, the latter inclusion seems to be unlikely and we refer the interested reader to [14, Chapter 8] for a detailed treatment of the corresponding question whether $\mathrm{A}[2] \subseteq \mathsf{W}[\mathsf{P}]$ in the decision world. We conclude with the remark that even a proof of $\mathrm{A}[2] \subseteq \#\mathsf{W}[\mathsf{P}]$ would be a major breakthrough as it constitutes the first step of a parameterized analogue of Toda's theorem [31], which is one of the fundamental open problems in (structural) parameterized counting complexity.

References

1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995. URL: http://webdam.inria.fr/Alice/.

2 Ashok K. Chandra and Philip M. Merlin. Optimal Implementation of Conjunctive Queries in Relational Data Bases. In *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*, pages 77–90, 1977. doi:10.1145/800105.803397.

3 Chandra Chekuri and Anand Rajaraman. Conjunctive query containment revisited. *Theoretical Computer Science*, 239(2):211–229, 2000. doi:10.1016/S0304-3975(99)00220-0.

4 Hubie Chen and Stefan Mengel. A Trichotomy in the Complexity of Counting Answers to Conjunctive Queries. In *18th International Conference on Database Theory, ICDT 2015, March 23-27, 2015, Brussels, Belgium*, pages 110–126, 2015. doi:10.4230/LIPIcs.ICDT.2015.110.

5 Hubie Chen and Stefan Mengel. Counting Answers to Existential Positive Queries: A Complexity Classification. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 315–326, 2016. doi:10.1145/2902251.2902279.

6 Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms are a good basis for counting small subgraphs. In *Proceedings of the 49th ACM Symposium on Theory of Computing, STOC*, pages 210–223, 2017. doi:10.1145/3055399.3055502.

7 Radu Curticapean and Dániel Marx. Complexity of Counting Subgraphs: Only the Boundedness of the Vertex-Cover Number Counts. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 130–139, 2014. doi:10.1109/FOCS.2014.22.

8 Víctor Dalmau and Peter Jonsson. The complexity of counting homomorphisms seen from the other side. *Theoretical Computer Science*, 329(1):315–323, 2004. doi:10.1016/j.tcs.2004.08.008.

9 Víctor Dalmau, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint Satisfaction, Bounded Treewidth, and Finite-Variable Logics. In Pascal Van Hentenryck, editor, *Principles and Practice of Constraint Programming - CP 2002, 8th International Conference, CP 2002, Ithaca, NY, USA, September 9-13, 2002, Proceedings*, volume 2470 of *Lecture Notes in Computer Science*, pages 310–326. Springer, 2002. doi:10.1007/3-540-46135-3_21.

10 Reinhard Diestel, Tommy R. Jensen, Konstantin Yu. Gorbunov, and Carsten Thomassen. Highly Connected Sets and the Excluded Grid Theorem. *Journal of Combinatorial Theory, Series B*, 75(1):61–73, 1999. doi:10.1006/jctb.1998.1862.

11 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

12 Arnaud Durand and Stefan Mengel. Structural Tractability of Counting of Solutions to Conjunctive Queries. *Theory of Computing Systems*, 57(4):1202–1249, 2015. doi:10.1007/s00224-014-9543-y.

13 Jörg Flum and Martin Grohe. The Parameterized Complexity of Counting Problems. *SIAM Journal of Computing*, 33(4):892–922, 2004. doi:10.1137/S0097539703427203.

14 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. doi:10.1007/3-540-29953-X.

15 Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, Saket Saurabh, and B. V. Raghavendra Rao. Faster algorithms for finding and counting subgraphs. *jcss*, 78(3):698–706, 2012. doi:10.1016/j.jcss.2011.10.001.

16 Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and Ryan Williams. Completeness for First-Order Properties on Sparse Structures with Algorithmic Applications. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2162–2181, 2017. doi:10.1137/1.9781611974782.141.

17 Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *Journal of the ACM*, 54(1):1, 2007. doi:10.1145/1206035.1206036.

18 Martin Grohe, Thomas Schwentick, and Luc Segoufin. When is the evaluation of conjunctive queries tractable? In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 657–666, 2001. doi:10.1145/380752.380867.

19 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

20 Anthony C. Klug. On conjunctive queries containing inequalities. *Journal of the ACM*, 35(1):146–160, 1988. doi:10.1145/42267.42273.

21 Miroslaw Kowaluk, Andrzej Lingas, and Eva-Marta Lundell. Counting and Detecting Small Subgraphs via Equations. *SIAM Journal on Discrete Mathematics*, 27(2):892–909, 2013. doi:10.1137/110859798.

22 László Lovász. Operations with structures. *Acta Mathematica Hungarica*, 18(3-4):321–328, 1967.

23 László Lovász. *Large networks and graph limits*, volume 60. American Mathematical Society Providence, 2012.

24 Dániel Marx. Can You Beat Treewidth? *Theory of Computing*, 6(1):85–112, 2010. doi:10.4086/toc.2010.v006a005.

25 Stefan Mengel. *Conjunctive queries, arithmetic circuits and counting complexity*. PhD thesis, University of Paderborn, 2013. URL: http://nbn-resolving.de/urn:nbn:de:hbz:466:2-11944.

26 Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.

27 Reinhard Pichler and Sebastian Skritek. Tractable counting of the answers to conjunctive queries. *Journal of Computer and System Sciences*, 79(6):984–1001, 2013. doi:10.1016/j.jcss.2013.01.012.

28 Mihai Pătraşcu and Ryan Williams. On the Possibility of Faster SAT Algorithms. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1065–1075, 2010. doi:10.1137/1.9781611973075.86.

29 Gian-Carlo Rota. On the foundations of combinatorial theory I. Theory of Möbius functions. *Probability theory and related fields*, 2(4):340–368, 1964.

30 Marc Roth. Counting Restricted Homomorphisms via Möbius Inversion over Matroid Lattices. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, pages 63:1–63:14, 2017. doi:10.4230/LIPIcs.ESA.2017.63.

31 Seinosuke Toda. PP is as Hard as the Polynomial-Time Hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991. doi:10.1137/0220053.

32 Ryan Williams. Faster decision of first-order graph properties. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 80:1–80:6, 2014. doi:10.1145/2603088.2603121.

33 Virginia Vassilevska Williams and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM Journal of Computing*, 42(3):831–854, 2013. doi:10.1137/09076619X.

A Faster Deterministic Exponential Time Algorithm for Energy Games and Mean Payoff Games

Dani Dorfman
Blavatnik School of Computer Science, Tel Aviv University, Israel
dannatand@mail.tau.ac.il

Haim Kaplan
Blavatnik School of Computer Science, Tel Aviv University, Israel
haimk@post.tau.ac.il

Uri Zwick
Blavatnik School of Computer Science, Tel Aviv University, Israel
zwick@tau.ac.il

Abstract

We present an improved exponential time algorithm for Energy Games, and hence also for Mean Payoff Games. The running time of the new algorithm is $O\left(\min\left(mnW, mn2^{n/2}\log W\right)\right)$, where n is the number of vertices, m is the number of edges, and when the edge weights are integers of absolute value at most W. For small values of W, the algorithm matches the performance of the pseudopolynomial time algorithm of Brim et al. on which it is based. For $W \geq n2^{n/2}$, the new algorithm is faster than the algorithm of Brim et al. and is currently the fastest *deterministic* algorithm for Energy Games and Mean Payoff Games. The new algorithm is obtained by introducing a technique of forecasting repetitive actions performed by the algorithm of Brim et al., along with the use of an edge-weight *scaling* technique.

2012 ACM Subject Classification Computing methodologies → Stochastic games

Keywords and phrases Energy Games, Mean Payoff Games, Scaling

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.114

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Funding Haim Kaplan was partially supported by the Israel Science Foundation, grant 1841/14, by the German-Israeli Foundation for Scientific Research and Development, grant 1367/2017, and by the Blavatnik Research Foundation. Part of this research was carried out while Uri Zwick was visiting BARC, Copenhagen, funded by the VILLUM Foundation, grant 16582, and while he was visiting IRIF, Paris, with support from the FSMP.

1 Introduction

Energy Games (EGs) and Mean Payoff Games (MPGs) are simple and natural infinite-duration games played on graphs that can be used to model quantitative properties of interactive systems. They are also interesting as they are perhaps the most natural combinatorial problems that are in $NP \cap \text{co-}NP$ and yet not known to be in P or in BPP. Mean Payoff Games (MPGs) were introduced by Ehrenfeucht and Mycielski [9]. Energy Games (EGs) were introduced by Chakrabarti et al. [7] and later by Bouyer et al. [4] who also showed their equivalence to MPGs.

Energy Games are games played by two players, player 0 and player 1, on a weighted directed graph whose vertices are partitioned among the two players. The two players construct an infinite path, that starts at a designated start vertex, in the following way. The player controlling the end-point u of the path constructed so far extends the path by

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 114; pp. 114:1–114:14


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

choosing an edge emanating from u. Let $w_1, w_2, \ldots$ be the weights of the edges on the path constructed. Player 0 wins this play if $\liminf_{n\to\infty} \sum_{i=1}^{n} w_i > -\infty$, i.e., if there exists an initial finite *energy level* c such that $c + \sum_{i=1}^{n} w_i \geq 0$, for every $n \geq 1$. Player 1 wins otherwise. Player 0 wins the game from an initial vertex u if she can ensure a winning play from u, no matter what player 1 does. It is known that if player 0 can win from a certain vertex, then she can also do it using a *positional strategy*, i.e., a deterministic strategy in which the edge chosen depends only on the current vertex. Furthermore, she has a single positional strategy using which she wins from all the vertices from which she can win. Solving an EGs amounts to finding the winner from each vertex, and possibly an optimal positional strategy and the minimal energy level required from every winning vertex.

Parity Games (PGs) form a very special sub-class of MPGs. In a recent breakthrough, Calude et al. [6] obtained a deterministic quasipolynomial $n^{O(\log n)}$-time algorithm for PGs, where n is the number of vertices. (Variants of their algorithm were obtained by [3, 10, 13, 18, 21].) Unfortunately, these techniques do not seem applicable to MPGs and EGs. (See [11].) The currently fastest algorithm for these games, as well as the more general (turn-based) Stochastic Games (SGs), is a sub-exponential $2^{\tilde{O}(\sqrt{n})}$ ([1, 2, 16, 17, 23]). These sub-exponential algorithms are based on randomized pivoting rules for the simplex algorithm devised by Kalai [19, 20] and Matoušek, Sharir and Welzl [24]. The fastest known deterministic algorithms for EGs and MPGs are the exponential $O(mn2^n \log W)$-time algorithm of Lifshits and Pavlov [22],[1] and a pseudo-polynomial $O(mnW)$-time algorithm of Brim et al. [5].[2] Polynomial time algorithms for EGs with very special weight structures were obtained by Chatterjee et al. [8].

The simple and elegant $O(mnW)$-time algorithm of Brim et al. [5], henceforth referred to as the BCDGR algorithm, is a *progress measure lifting* algorithm for solving EGs. MPGs are essentially equivalent to EGs ([4]), hence the algorithm can also be used to solve MPGs. The lifting technique used by Brim et al. is similar to the *value iteration* technique used by Zwick and Paterson [26] on MPGs.

We present an improvement of the BCDGR algorithm that runs in $O(\min\{mnW, mn2^{n/2} \log W\})$-time. The new algorithm is always as fast as the BCDGR algorithm and strictly faster when $W = \omega(n2^{n/2})$. The running time of the new algorithm can be made to be $O(poly(n)2^{n/2})$, without any dependence on W, assuming that arithmetic operations on integers of absolute value $O(nW)$ take constant time. (Details will appear in the full version of the paper.) The new algorithm is currently the fastest *deterministic* algorithm for EGs and MPGs when $W \geq n2^{n/2}$.

The new algorithm uses two new ideas. The first is a technique for predicting sequences of update steps that are performed repetitively by the BCDGR algorithm, and achieving the net effect of these repetitions much more quickly. To make this approach work, a second idea, that of *scaling*, needs to be used. Scaling is a well-known technique used in various combinatorial optimization problems such as shortest paths, flow problems, matching problems etc. (See, e.g., [12, 14, 15].) A scaling algorithm first divides all edge weights by 2, rounds them up so that they remain integers, solves the reduced problem recursively, and then converts the solution of the reduced problem to a solution of the original algorithm. It is quite natural to try to use the scaling technique on EGs or MPGs. However, naïve or

---

[1] For solving EGs, and for deciding whether the values of a MPG are non-negative, the $\log W$ factor in the running time of [22] is not needed.

[2] Recently, Fijalkow et al. [11] gave an $O(mn(nW)^{1-1/n})$-time algorithm for solving MPGs. This, however, is never asymptotically better than $O(\min\{mnW, mn2^n\})$, as $W^{1-1/n} < \frac{1}{2}W$ only if $W \geq 2^n$.

direct approaches do not seem to give any improvement. To the best of our knowledge, the algorithm presented here is the first algorithm that successfully uses scaling for speeding up the solution of EGs and MPGs.

An EG, MPG or SG is said to be *binary* if the outdegree of each vertex is 2. It is known that binary EGs, MPGs and SGs can be modeled as *Acyclic Unique Sink Orientations* (AUSOs) (see, e.g., [23, 25]). Solving a game is then equivalent to finding the *sink* of the associated AUSO. The fastest deterministic sink-finding algorithm runs in $O(1.606^n)$-time. Our new algorithm is faster than this algorithm and works for all games, not only binary.

The rest of the paper is organized as follows. In the next section we provide some definitions and basic results and briefly review the algorithm of Brim et al. [5] on which our new algorithm is based. In Section 3 we present our new algorithm. In the full version of the paper we describe energy games on which the new algorithm requires $\Omega(2^{n/2})$ time, showing that our analysis is essentially tight. We end in Section 4 with concluding remarks and open problems.

2 Preliminaries

A **game graph** is a tuple $\Gamma = (V_0, V_1, E, w)$, where $V = V_0 \cup V_1$ is the set of vertices, $E \subseteq V \times V$ is the set of edges, and $w : E \to \mathbb{Z}$ is a weight function. We assume that $V_0 \cap V_1 = \emptyset$ and that each vertex has at least one outgoing edge. The sets V_0 and V_1 are the sets of vertices controlled by player 0 and player 1. A *positional strategy* of player i is a mapping $\sigma : V_i \to E$ such that for every $v \in V_i$ we have $(v, \sigma(v)) \in E$. Given *positional strategies* σ_0, σ_1 of player 0 and player 1 and an initial vertex v_0, $play(v_0, \sigma_0, \sigma_1) = v_0, v_1, \ldots, v_i, \ldots$ is the infinite walk resulting from σ_0 and σ_1 starting at v_0.

An **Energy-Game** is an infinite game on a game graph Γ. Player 0 wins from an initial vertex $v_0 \in V$ if and only if there exists a positional strategy σ_0, and a finite *energy level* $c = c(v_0)$, such that for every positional strategy σ_1 of player 1, we have $c + \sum_{i=0}^{n-1} w(v_i, v_{i+1}) \geq 0$, for every $n \geq 1$, where $play(v, \sigma_0, \sigma_1) = v_0, v_1, \ldots$.

We shall refer to a function $f : V \to \mathbb{N} \cup \{\infty\}$ as a *potential* function.

▶ **Definition 2.1.** *Let $\Gamma = (V_0, V_1, E, w)$ be an energy-game. A function $f : V \to \mathbb{N} \cup \{\infty\}$ is a feasible potential iff for every $v \in V$:*

- *if $v \in V_0$, then $f(v) + w(v, v') \geq f(v')$ for some $(v, v') \in E$.*
- *if $v \in V_1$, then $f(v) + w(v, v') \geq f(v')$ for all $(v, v') \in E$.*

We call the potential function $g(v) = \min\{f(v) \mid f \text{ feasible potential}\}$ the solution *of Γ.*

Brim et al. [5] proved that g is a *feasible* potential and that player 0 wins from v if and only if $g(v) < \infty$, in which case $g(v)$ is the minimal required initial energy.

Let $\Gamma = (V_0, V_1, E, w)$ be an energy-game and let $f : V \to \mathbb{N} \cup \{\infty\}$ be a *potential* function. We denote by $w_f(u, v) = w(u, v) + f(u) - f(v)$ the *modified* weight of (u, v). An edge (u, v) is *valid* with respect to f if $w_f(u, v) \geq 0$. A vertex $v \in V_0$ (V_1) is *valid* with respect to f if (v, v') is *valid* with respect to f for some (all) $(v, v') \in E$, otherwise we say that v is *invalid* with respect to f (we say just *valid* when f is clear from the context). An edge (v, v') is *tight* if $w_f(v, v') = 0$. A path p is *tight* if all its edges are tight. A vertex $v \in V_0$ is *tight* if it is *valid* and $w_f(v, v') \leq 0$ for all $(v, v') \in E$. A vertex $v \in V_1$ is *tight* if it is *valid* and $w_f(v, v') = 0$ for some $(v, v') \in E$. We denote by $in(u)$ and $out(u)$ the sets of incoming and outgoing edges from u, respectively.

2.1 The algorithm of Brim et al.

Brim et al. [5] suggested the following algorithm: maintain $f : V \to \mathbb{N} \cup \{\infty\}$, starting with $f \equiv 0$. As long as there are *invalid* vertices, pick some *invalid* vertex v and increase $f(v)$ to the minimal value that would make v *valid*. It is known that if player 0 can win from a certain vertex, then she can win with an initial energy of at most nW. Thus, if $f(v)$ reaches nW, we know that v is a losing vertex for player 0, and we can let $f(v) \leftarrow \infty$.

To efficiently find an invalid vertex, the algorithm maintains a list L of *invalid* vertices. When $f(v)$ of some *invalid* vertex $v \in V$ is updated, the algorithm checks for every edge $(v', v) \in in(v)$ that became *invalid* whether v' is now also *invalid*. If $v' \in V_1$, then this is the case, and v' is added to L, if it is not already there. If $v' \in V_0$, then increasing $f(v)$ does not necessarily make v' *invalid*, as v' may have had other valid edges. The algorithm maintains $count[v']$, the number of valid edges in $out(v')$. If (v', v) was valid, then $count[v']$ is decremented. If $count[v']$ becomes 0, then v' is now *invalid* and it is added to L. It is not hard to check that the running time of the resulting algorithm is $O(mnW)$, which is also known to be tight.

2.2 A reduction to games with finite values

The description and the correctness proof of algorithms for solving EGs are often simplified if it assumed that all vertices have finite values, i.e., are all winning for player 0. (This does not trivialize the problem, as we still want to find the minimum energy level needed from each vertex, and corresponding optimal positional strategies for the two players.) We describe a simple reduction, inspired by a reduction of Björklund et al. [2], that shows that the solution of a general EG can be reduced to the solution of an EG with finite values.

Let $\Gamma = (V_0, V_1, E, w)$ be an EG, and let $n = |V|$ and $W = \max_{e \in E} |w(e)|$. Let f be the solution of Γ. For every $v \in V$, we know that either $0 \leq f(v) < nW$, or $f(v) = \infty$. To convert Γ into a game Γ' in which all values are finite, we add a *sink* vertex s, with a self-loop of weight 0, and add an edge (v, s) of weight $-2nW$ for every $v \in V_0$. This ensures that the values of all vertices in V_0 are finite. (In particular, their value is at most $2nW$.)

To ensure that the values of all vertices in V_1 are also finite, we need to perform a simple preprocessing step. If $u \in V_1$ and player 0 has a strategy for reaching a vertex of V_0, starting at u, then the value of u is also finite. We are thus left with vertices of V_1 from which player 1 can win the game, i.e., reach a negative cycle, without leaving V_1. It is easy to identify these vertices and remove them from the game. The value of all remaining vertices is now finite.

If it is to player 0's advantage to escape to the sink, she might as well do it without closing any cycles. Player 0 can therefore gain at most $(n-1)W$ units of energy by following original edges before deciding to take an edge to the sink. The energy needed in such a case is therefore at least nW. We thus have:

▶ **Lemma 2.2.** *Let $\Gamma = (V_0, V_1, E, w)$ be an EG and let Γ' be the EG obtained by the reduction above. Let f and f' be the solutions of Γ and Γ'. Then, for every $u \in V = V_0 \cup V_1$, we have*

$$f(u) = \begin{cases} f'(u) & \text{if } f'(u) < nW, \\ \infty & \text{otherwise.} \end{cases}$$

Note that the reduction introduces only one new vertex which is important if we want to use it in conjunction with exponential time algorithms. The maximal edge weight is increased from W to $2nW$, but this is not an issue if the running time of the algorithm depends only logarithmically on W.

```
COMPUTE-ENERGY (V_0, V_1, E, w).
1  if w ≥ 0 then
2      return f ≡ 0
3  w' ← ⌈w/2⌉
4  f ← COMPUTE-ENERGY(V_0, V_1, E, w')
5  f, w' ← 2f, 2w'
6  foreach v ∈ V do
7      foreach e ∈ out(v) do
8          if w'(e) > w(e) then w'(e) ← w'(e) − 1
9      UPDATE-ENERGY (V_0, V_1, E, w', f, v)
10 return f
```

```
UPDATE-ENERGY (V_0, V_1, E, w, f, v).
1  if v ∈ V_0 and ∀(v,u) ∈ E : w_f(v,u) < 0 then L ← {v}
2  if v ∈ V_1 and ∃(v,u) ∈ E : w_f(v,u) < 0 then L ← {v}
3  foreach u ∈ V_0 do
4      count[u] ← |{u' | (u,u') ∈ E, w_f(u,u') ≥ 0}|
5  while L = {v} do
6      B ← {v}
7      UPDATE(v, L, B)
8      while L \ {v} ≠ ∅ do
9          pick u ∈ L \ {v}
10         UPDATE(u, L, B)
11     Δ ← DELTA(B)
12     foreach u ∈ B do f(u) ← f(u) + Δ
```

Figure 1 The main two functions of the new $O\left(\min\left(mnW, mn2^{n/2}\log W\right)\right)$-time algorithm.

3 The new algorithm

We now describe our new algorithm. For simplicity, we assume that all the values in the input game are finite. This can be achieved, for example, using the simple reduction above. In the full version we show that the algorithm presented actually works, as is, even if some values are infinite, but the correctness proof becomes slightly more complicated.

Given an EG $\Gamma = (V_0, V_1, E, w)$, we construct a scaled down version $\Gamma' = (V_0, V_1, E, w' = \lceil \frac{w}{2} \rceil)$, where $\lceil \frac{w}{2} \rceil(e) = \lceil \frac{w(e)}{2} \rceil$, for every $e \in E$, and solve it recursively, obtaining the solution f' of Γ'. (Note that Γ' is "easier" for player 0 because of the rounding up. In particular, if all values in Γ are finite, so are all the values in Γ'.) We now scale Γ' back up to $\Gamma'' = (V_0, V_1, E, 2\lceil \frac{w}{2} \rceil)$. Note that $f'' \equiv 2f'$ is the solution of Γ'' and that Γ'' is very close to Γ: $2\lceil \frac{w}{2} \rceil$ and w only differ, by 1, on edges e for which $w(e)$ is *odd*. To convert f'' into a solution of Γ, we consider each vertex $v \in V$ with odd outgoing edges, decrease the corresponding edge weights in $2\lceil \frac{w}{2} \rceil$ by 1, and update the solution f'' accordingly. (This is, of course, the hardest part of the algorithm.) These operations are carried out by algorithms COMPUTE-ENERGY and UPDATE-ENERGY given in Figure 1.

UPDATE(u, L, B).

```
1 L ← L \ {u}
2 f(u) ← f(u) + 1
3 if u ∈ V_0 then count[u] ← |{(u, u') ∈ E | w_f(u, u') ≥ 0}|
4 foreach u' ∈ in(u) such that w_f(u', u) < 0 do
5     if u' ∈ V_0 then
6         if w_f(u', u) = −1 then count[u'] ← count[u'] − 1
7         if count[u'] = 0 then L ← L ∪ {u'}, B ← B ∪ {u'}
8     if u' ∈ V_1 then L ← L ∪ {u'}, B ← B ∪ {u'}
```

DELTA(B).

```
1 p_1 ← min{−w_f(u, u') | (u, u') ∈ E(B ∩ V_0, B̄)}
2 p_2 ← min{γ(u) | u ∈ B̄ ∩ V_0, ∀u' ∈ B̄  w_f(u, u') < 0}
3 p_3 ← min{w_f(u, u') | (u, u') ∈ E(B̄ ∩ V_1, B)}
4 return min{p_1, p_2, p_3}
```

Figure 2 The remaining two function of the new $O\left(\min\left(mnW, mn2^{n/2}\log W\right)\right)$-time algorithm.

UPDATE-ENERGY updates the solution f after the decrease of the weights of some of the edges emanating from v by 1. As in [5], UPDATE-ENERGY maintains a list L of all vertices that are currently *invalid*. Initially only v may be *invalid*. To quickly determine whether a vertex $u \in V_0$ becomes *invalid*, we maintain in $count[u]$ the number of *valid* edges from u. A vertex $u \in V_0$ is thus *invalid* iff $count[u] = 0$.

Lines 1–2 of UPDATE-ENERGY determine whether v is *invalid*. If v is *valid*, we are done. Otherwise, we add v to L and *fix* v by increasing $f(v)$ by 1. This makes v *valid*, but vertices u with edges (u, v) may become *invalid* and need to be added to L. Updating v's potential and checking for new *invalid* vertices is carried out by UPDATE given in Figure 2. If we fix *invalid* vertices from L in an arbitrary order, as done by [5], we get the running time of [5].

The main point in which our algorithm differs from the algorithm of [5], in addition to the use of scaling, is that we fix at first only *invalid* vertices *different* from v, delaying an additional fixing of v, if required, by as much as possible. Lemma 3.2 below shows that no vertex needs to be fixed twice, before v is fixed again.

If v does not become *invalid* again, then UPDATE-ENERGY is done. Otherwise, let B be the set of vertices, including v, updated until v is the only *invalid* vertex. (Lemma 3.2 shows that this must eventually happen.) When we update v again, it could be that the *same* set of vertices B will eventually become *invalid*, and hence updated, again. Furthermore, in worst-case examples of [5], the same sequences of update operations may be repeated many times. Instead of carrying out these updates again and again, we *predict* how many times the same sequence of updates will be repeated and perform all these updates at once. This approach seems to work only when the weights of edges are decreased by 1, which is why the scaling idea needs to be used.

The computation of Δ, the number of repetitions of the current sequence, carried out by DELTA(B) in Figure 2, is based on the following observation. Let B be the set of vertices that became *invalid* after updating v and let B' be the set of vertices that became *invalid* after updating v again. Assume $B' \neq B$. If $B' \setminus B \neq \emptyset$, let $u \in B' \setminus B$ be the first vertex in $B' \setminus B$ that became *invalid*. It must be the case that at least one of u's *valid* edges to B became *invalid*. If $u \in V_1$ this could be any edge from u, and if $u \in V_0$ this edge is the edge with maximal *modified* weight from u and u had no *valid* edges to $\bar{B} \equiv V \setminus B$.

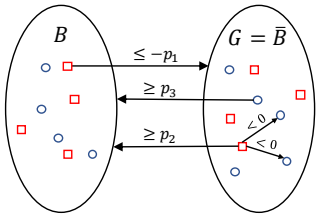


Figure 3 Calculating DELTA(B): Vertices in V_0 are red squares; Vertices in V_1 are blue circles.

If $B \setminus B' \neq \emptyset$, then $(B \setminus B') \cap V_0 \neq \emptyset$. To see this, let $v_0 = v, v_1, \ldots$ be the vertices of B in the order in which they were updated. Let v_j be the first vertex in this order which is not in B'. Vertex v_j must have an *invalid* edge (v_j, v_ℓ) such that $\ell < j$. This edge became *invalid* when we updated v_ℓ and caused the addition of v_j to B. Since v_j is the first vertex which is not in B', $v_\ell \in B'$ and therefore the edge (v_j, v_ℓ) becomes *invalid* when we collect B' following the second update of v. So $v_j \in V_0$, as otherwise it should have been added to B'. Vertex v_j is not added to L since it has an edge (v_j, w), $w \notin B$ that had a *modified* weight of -1 that became 0 (i.e., *valid*) following the update of v_j.

Therefore, to compute Δ, we must consider all *valid* edges from $\bar{B}$ to B (to detect new vertices that might become *invalid*) and all *invalid* edges from vertices in $V_0 \cap B$ to $\bar{B}$, see Figure 3. We refer to minimum and maximum of an empty set as ∞ and $-\infty$, respectively. We let $\Delta = \min\{p_1, p_2, p_3\}$ were p_1, p_2 and p_3 are defined as follows. The value p_1 is minus the maximum modified edge weight of an edge from $B \cap V_0$ to $\bar{B}$. Note that $p_1 \geq 0$. To define p_2 consider every vertex $u \in V_0 \cap \bar{B}$ that does not have a valid edge (u, w) to $w \in \bar{B}$. For every such u let $\gamma(u)$ be the maximum modified weight of an edge (u, w), $w \in B$. Note that $\gamma(u) \geq 0$. We define p_2 to be the minimum value of $\gamma(u)$ over all such vertices u. The value p_3 is the minimum modified edge weight of an edge from $V_1 \cap \bar{B}$ to B. Note that p_2 and p_3 are nonnegative. Pseudo-code of DELTA(B) is given in Figure 2.

3.1 Correctness

As explained, we assume for simplicity that all values are finite. This assumption is removed in the full version of the paper. The correctness of the new algorithm follows from the fact that the potential function kept by the algorithm is always a lower bound on the values of the vertices, and hence the updates performed are justified, as in the correctness proof of the BCDGR algorithm. As the new algorithm predicts sequence of updates that are going to be performed repeatedly, and performs all these repetitions at once, what remains to be shown is that the predictions of the algorithm are correct.

▶ **Lemma 3.1.** *Let* $\Gamma^1 = (V_0, V_1, E, w^1)$, $\Gamma^2 = (V_0, V_1, E, w^2)$ *be two games graphs with solutions* f^1 *and* f^2*, respectively. If* $w^1 \leq w^2$ *then* $f^1 \geq f^2$ *(coordinate-wise).*

It follows Lemma 3.1 that the solution of Γ'' is a lower bound on the solution of the original game Γ. All that remains, therefore, is to show that the updates performed by UPDATE-ENERGY are justified.

Let UPDATE($v_1 \equiv v$), UPDATE(v_2), ..., UPDATE(v_k) be the sequence of vertex updates performed by UPDATE-ENERGY($\ldots, f, v$). A *round* is one iteration of the outer while loop of UPDATE-ENERGY, i.e., all vertex updates starting with UPDATE(v) until and not including the next UPDATE(v). We number the rounds starting from 1 and let f^r be f at the end of round r. For convenience, we define $f^0 \equiv f$. We let B^r be the set of vertices that were updated during round r ("bad" vertices). Thus, B^r is B at the end of round r of the outer while loop of UPDATE-ENERGY. Let $G^r = \overline{B^r} = V \setminus B^r$ be the set of "good" vertices.

▶ **Lemma 3.2.** *In each round, each vertex is updated at most once.*

Proof. By contradiction, let u be the first vertex that joined L for the second time during a round. We have that $u \neq v$ by the definition of a *round*. Assume $u \in V_0$. Since $u \neq v$ (i.e., u is *valid* at the beginning of the round), $w_f(u, u') \geq 0$ for some vertex u' at the beginning of the round. From the beginning of the round until u's second update, u' is updated at most once and u is updated exactly once so we have that $w_f(u, u') \geq 0$ right before u's second update which is a contradiction (u is *valid*). A similar argument works when $u \in V_1$. ◀

▶ **Lemma 3.3.** *During* Update-Energy, $u \in L$ *if and only if* u *is* invalid.

Proof. By induction on the iterations of the algorithm. ◀

Note that vertices u that were never updated (in any call to Update-Energy) are those with $f(u) = 0$. Also, note that every *tight* vertex u has at least one *tight* edge.

▶ **Lemma 3.4.** *During* Update-Energy, *if* $u \notin L$ *and* $f(u) > 0$, *then* u *is tight.*

Proof. Following an update, u becomes *tight* and it remains *tight* as long as it is *valid*. Since $f(u) > 0$, u was updated and since $u \notin L$, u is *valid* (Lemma 3.3). ◀

▶ **Lemma 3.5.** *At the end of round* r, *every* $u \in (G^r \setminus \{v \mid f(v) = 0\}) \cap V_0$ *is tight and for every tight edge* (u, u') *it holds that* $u' \in G^r$. *Also, there exists* $u' \in G^r$ *such that* $(u, u') \in E$ *is* tight *during round* r.

Proof. We prove the first part by contradiction. Let (u, u') be a *tight* edge at the end of round r such that $u' \in B^r$. Since $u' \in B^r$, u' was updated during round r and therefore at the beginning of round r it holds that $w_f(u, u') > 0$, so u was not *tight* at the beginning of the round. This contradicts Lemma 3.4 that implies that u is always *tight* during round r.

We now prove the second part. Since u is *tight* during the round and in particular at the end of the round, $(u, u') \in E$ is *tight* for some u' at the end of the round. By the first part of the Lemma $u' \in G^r$. Thus, (u, u') is *tight* during the entire round (since both $f(u)$ and $f(u')$ remain unchanged during the round). ◀

▶ **Lemma 3.6.** *At the end of round* r, *if* $u \in (B^r \setminus \{v\}) \cap V_1$ *and* $(u, u') \in E$ *is* tight, *then* $u' \in B^r$.

Proof. If $u' \in G^r$ then (u, u') was *invalid* at the beginning of round r (since $f(u)$ but not $f(u')$ was increased during the round), and therefore u was *invalid* at the beginning of round r, but only v is invalid at the beginning of each round, a contradiction. ◀

▶ Remark. Note that we cannot guarantee that u has a *tight* edge during the round (as in Lemma 3.5). This is because when u becomes *invalid*, it might be the case that all of its edges became *invalid* (u is ensured to have a *tight* edge only when it is *valid*).

The proof of the following lemma is given in the full version of the paper.

▶ **Lemma 3.7.** *Consider round* r. *If we would have performed* Update-Energy *without lines* 11–12, *then in the following* Δ *rounds* $r + 1, ..., r + \Delta$ *we would have* $B^r = B^{r+i}$, *for* $1 \leq i \leq \Delta$, *where* Δ *is the value returned by* Delta(B). *Furthermore, if* $r + \Delta$ *is not the last round then* $B^{r+\Delta+1} \neq B^r$.

As a consequence we get:

▶ **Theorem 3.8.** Update-Energy (V_0, V_1, E, w, f, v) *updates* f *correctly.*

The rest of the lemmas in this section are used in the next section to bound the complexity of the algorithm.

Let u be a *valid* vertex such that $f^r(u) > f^0(u)$ (i.e., the potential of u was changed at least once before the end of round r). We let $t_r(u)$ be the time right after the last UPDATE(u) that occurred before the end of round r. We remove the subscript r when it is clear from the context.

▶ **Lemma 3.9.** *At the end of round r, for any $u \in G^r$ with $f^r(u) > f^0(u)$:*
1. *If $u \in V_0$ then for any* tight *edge (u, u') with $u' \in G^r$, either $f^r(u') = f^0(u')$ or $t_r(u') < t_r(u)$.*
2. *If $u \in V_1$ then there exists a* tight *edge (u, u') such that $u' \in G^r$ and $t_r(u') < t_r(u)$.*

Proof. Note that since $u \in G^r$, $t_r(u)$ is before round r begins. We begin with the first part. Let (u, u') be such an edge. If $f^r(u') = f^0(u')$ then we are done. Otherwise, u' must have been updated at least once (and therefore $t(u')$ is defined). Assume by contradiction that $t(u') > t(u)$. Since (u, u') is *tight* at the end of round r then $w_f(u, u') > 0$ at $t_r(u)$. This contradicts Lemma 3.4 since at $t_r(u)$ it holds that u is *valid* and not *tight*.

We now prove the second part. Since u must be *tight* at $t(u)$ there exists $u' \in V$ such that (u, u') is *tight* at $t(u)$. Note that u' must be *valid* from $t(u)$ until the end of round r, since otherwise u will become *invalid* after $t(u')$ which is a contradiction. Thus, $u' \in G^r$ and $t(u') < t(u)$. Since u and u' remain *valid* from $t(u)$ until the end of round r, (u, u') is *tight* at the end of round r. ◀

▶ **Lemma 3.10.** *At the end of round r the following holds for all $u \in V$:*
1. *If $u \in B^r$ then u has a* tight *path of vertices in B^r to v.*
2. *If $u \in G^r$ then u has a* tight *path of vertices in G^r to a vertex u' with $f^r(u') = f^0(u')$.*

Proof. We begin by proving the first claim. Every vertex $u \in B^r$, $u \neq v$ joins L because of some edge (u, u') which is *invalid* after we update u'. This edge must be *tight* at the end of the round. So each vertex u has a *tight* edge to a vertex u' which was updated before u during round r. This implies the first part.

We now prove the second claim. If $f^r(u) = f^0(u)$ then we are done. Otherwise, assume $f^r(u) > f^0(u)$. We continue the proof by induction on $t(u)$. Base case, $t(u)$ is minimal (i.e., u was updated first). By Lemma 3.9, u has a *tight* edge (u, u') with $u' \in G^r$ such that $f^r(u') = f^0(u')$ (since $t(u)$ is minimal) and we are done. Assume that the claim follows for all vertices u' with $t(u') < t(u)$. By Lemma 3.9, u has a *tight* edge (u, u') with $u' \in G^r$ such that either $f^r(u') = f^0(u')$ or $t(u') < t(u)$. If $f^r(u') = f^0(u')$ then we are done. Otherwise, $t(u') < t(u)$ and therefore by the induction hypothesis, u' has a *tight* path to some vertex u'' with $f^r(u'') = f^0(u'')$. ◀

3.2 Complexity

Recall that $|V| = n, |E| = m$ and W is the maximal absolute value weight.

▶ **Theorem 3.11.** *The running time of compute–energy is* $O\big(\min\big(mnW, mn \cdot 2^{n/2} \log W\big)\big)$.

The $O(mnW)$ bound follows immediately since each vertex u is updated at most $O(|V| \cdot W)$ times and each such update takes $O(|in(u)| + |out(u)|)$ time, where $in(u)$ and $out(u)$ are the sets of ingoing and outgoing edges of u, respectively. To prove the latter bound we must have a better understanding of the relation between B^r and G^r. In the rest of this section we prove the $O\left(mn \cdot 2^{n/2} \log W\right)$ bound.

For this we define a potential function that maps rounds (as defined in Section 3.1) into integers. The *good anchor* of round r is defined as the set $GA^r = V \setminus \bigcup_{i=1}^{r} B^i$, i.e., vertices whose potential was not changed yet ($f^r(u) = f^0(u)$). Following each round the *good anchor* can only lose vertices. The *bad anchor* of round r is defined as the set $BA^r = \{u \in B^r \mid \exists \textit{tight}$ path of vertices in $V_0 \cap B^r$ from u to $v\}$, i.e., BA^r contains v and all vertices in $V_0 \cap B^r$ that have a *tight* path of vertices in $V_0 \cap B^r$ to v. Note that if $GA^r = \emptyset$, then no vertex is winning for player 0: To see this, note that in this case $f(u) > 0$ for all $u \in V$. Therefore, the potential $f'(v) \equiv f(v) - 1$, for all $v \in V$ is *feasible*, a contradiction (since f is the *solution*).

We partition B^r and G^r into layers BL_i^r, GL_i^r, $i = 0, 1 \ldots$, respectively, see Figure 4. The layer BL_i^r/GL_i^r is called the i'th layer of B^r/G^r, respectively. The 0'th layers are the anchors, i.e., $BL_0^r = BA^r, GL_0^r = GA^r$. The layers are defined inductively as follows.

$$\begin{aligned} BL_i^r &= \{u \in B^r \cap V_p \mid \ u \text{ has a tight path of vertices in } u \in B^r \cap V_p \text{ to } BL_{i-1}^r\} \\ GL_i^r &= \{u \in G^r \cap V_{1-p} \mid \ u \text{ has a tight path of vertices in } u \in G^r \cap V_{1-p} \text{ to } GL_{i-1}^r\}. \end{aligned} \tag{1}$$

where $p = i \ (mod\ 2)$.

The following lemmas prove that only the first layers have tight edges to the anchors.

▶ **Lemma 3.12.** *At the end of round r, if $(u, u') \in E$ is a tight edge s.t $u \in G^r \setminus GA^r$ and $u' \in GA^r$, then $u \in GL_1^r$.*

Proof. It suffices to show that $u \in V_0$. By contradiction, assume that $u \in V_1$. Since $u \notin GA^r$, u was in $B^{r'}$ at some round $r' < r$. Let UPDATE(z) be the update that added u into L in round r' and let f_1 be f right before this UPDATE(z). Clearly, $z \neq u'$ (since $u' \in GA^r$). Since u is was *valid* before UPDATE(z) in round r', and $u \in V_1$, we have that $f_1(u) + w(u, u') \geq f_1(u')$. Therefore, since $f(u')$ did not change until the end of round r and u must have been updated following the update of z and before the end of round r, we have that $f(u) + w(u, u') > f_1(u) + w(u, u') \geq f_1(u') = f(u')$ at the end of round r, a contradiction to the assumption that (u, u') is *tight* at the end of round r. ◀

▶ **Lemma 3.13.** *At the end of round r, if $(u, u') \in E$ is a tight edge s.t $u \in B^r \setminus BA^r$ and $u' \in BA^r$, then $u \in BL_1^r$.*

Proof. Immediate from the definition of BA^r. ◀

The following lemma, which follows immediately from Lemma 3.10, shows that every vertex belongs either to an *anchor* or to some layer of B^r or G^r.

▶ **Lemma 3.14.** *For any round r, $B^r = \bigcup_{i=0}^{n} BL_i^r$, $G^r = \bigcup_{i=0}^{n} GL_i^r$.*

We associate with B^r and G^r binary numbers b^r and g^r, respectively of length $n+1$ defined as follows. Let k be maximal such that $|BL_k^r| > 0$. Then, b^r is:

$$b^r = \begin{cases} \underbrace{1.....1}_{|BL_1^r|}\underbrace{0.....0}_{|BL_2^r|}.....\underbrace{1.....1}_{|BL_k^r|}\ \underbrace{10.........0}_{n+1-|\bigcup_i BL_i^r|} & \text{if } k \text{ is odd} \\ \underbrace{1.....1}_{|BL_1^r|}\underbrace{0.....0}_{|BL_2^r|}.....\underbrace{0.....0}_{|BL_k^r|}\ \underbrace{10.........0}_{n+1-|\bigcup_i BL_i^r|} & \text{if } k \text{ is even.} \end{cases}$$

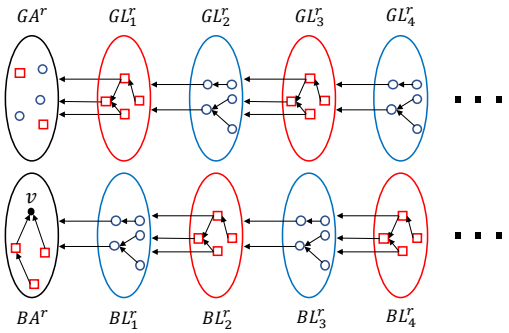


Figure 4 The layer graph of round r. Red vertices refer to V_0 and blue vertices refer to V_1. All drawn edges are *tight* at the end of round r. By Definition (1), each layer is either contained in V_0 or in V_1.

That is, for an odd layer we add a sequence of 1's whose length is the size of the layer. Similarly, for even layers we add sequences of 0's. At the end we pad the number with a single 1 followed by zeros. The number g^r is defined similarly with respect to the layers of G^r. Finally, the potential ϕ^r of round r is defined as $\phi^r = b^r + g^r$. Clearly $\phi^r \leq 2 \cdot 2^{n+1}$. In Lemma 3.18 we prove that for every round r, under certain conditions (that can be violated in at most $|V|^2$ rounds), $\phi^{r+1} \geq \phi^r + 2^{n/2}$, yielding the desired runtime.

The following lemmas consider UPDATE-ENERGY at the end of round r.

▶ **Lemma 3.15.** *For every r, $GA^{r+1} \subseteq GA^r$ and if $GA^{r+1} = GA^r$, then $BA^{r+1} \subseteq BA^r$.*

Proof. Since f only grows the first claim follows directly from the definition of the algorithm. We prove the second claim by contradiction. Assume $GA^{r+1} = GA^r$ and let $u \in BA^{r+1} \setminus BA^r$. By definition of *bad anchor* $u \in V_0$ and at the end of round $r+1$ there exists a *tight* path $p = v_0 v_1 ... v_k$ from $u = v_0$ to $v = v_k$ such that $v_i \in V_0 \cap B^{r+1}$ for all $i < k$. Let j be maximal such that $v_j \in BA^{r+1} \setminus BA^r$ (therefore $v_{j+1} \in BA^{r+1} \cap BA^r$, j is well defined since $u \in BA^{r+1} \setminus BA^r$). If $v_j \in B^r$ then (v_j, v_{j+1}) was tight also at the end of round r (since both $v_j, v_{j+1} \in B^r \cap B^{r+1}$) and thus $v_j \in BA^r$, a contradiction. So we have that $v_j \in G^r$. Therefore, at the beginning of round r it must hold that $f(v_j) + w(v_j, v_{j+1}) > f(v_{j+1})$ (i.e., v_j is not *tight* at the beginning of round r). By Lemma 3.3, $v_j \in GA^r$, a contradiction to the assumption that $GA^r = GA^{r+1}$. ◀

The following lemma is similar to Lemma 3.7. Its proof is given in the full version of the paper.

▶ **Lemma 3.16.** *For every r, $B^{r+1} \neq B^r$.*

▶ **Lemma 3.17.** *Suppose that $GA^{r+1} = GA^r$ and $BA^{r+1} = BA^r$.*

1. *Let i be the smallest such that $GL_i^{r+1} \neq GL_i^r$. Then, if i is odd then $GL_i^r \subset GL_i^{r+1}$, and if i is even then $GL_i^{r+1} \subset GL_i^r$.*
2. *Let i be the smallest such that $BL_i^{r+1} \neq BL_i^r$. Then, if i is odd then $BL_i^r \subset BL_i^{r+1}$, and if i is even then $BL_i^{r+1} \subset BL_i^r$.*

Proof. We prove only the first claim as the latter is similar. We divide the proof into cases according to the parity of i.

i **is *odd***: We show that $GL_i^r \subset GL_i^{r+1}$. By contradiction, assume that $\exists u \in GL_i^r \setminus GL_i^{r+1}$. By Definition (1), $u \in V_0$ and at the end of round r there exists a tight path $p = u_0, u_1, ... u_k$ from $u_0 = u \in GL_i^r$ to $u_k \in GA^r = GL_0^r$ that traverses the "good" layers in non-increasing

order. Let j be the maximal such that $u_j \in GL_i^r \setminus GL_i^{r+1}$. Thus, either $u_{j+1} \in GL_{i-1}^r$ or $u_{j+1} \in GL_i^r \cap GL_i^{r+1}$. Note that in both cases $u_{j+1} \in G^{r+1}$ and therefore we have that also $u_j \in G^{r+1}$ (since (u_j, u_{j+1}) was tight at the end of round r and remains tight during round $r+1$). Assume $u_{j+1} \in GL_{i-1}^r$. Since $GL_{i-1}^r = GL_{i-1}^{r+1}$ and since (u_j, u_{j+1}) is *tight* at the end of round $r+1$, we have that $u_j \in GL_\ell^{r+1}$ for some $\ell \leq i$, a contradiction (since $u_j \notin GL_i^{r+1}$ and because lower layers are equal in both rounds by our assumption). Assume now that $u_{j+1} \in GL_i^r \cap GL_i^{r+1}$. We get a contradiction since $u_j \in GL_\ell^{r+1}$ for some $\ell \leq i$.

i **is *even***: We that show $GL_i^{r+1} \subset GL_i^r$. By contradiction, assume that $\exists u \in GL_i^{r+1} \setminus GL_i^r$. By Definition (1), $u \in V_1$ and at the end of round $r+1$ there exists a tight path $p = u_0, u_1, ...u_k$ from $u_0 = u \in GL_i^{r+1}$ to $u_k \in GA^{r+1} = GL_0^{r+1}$ that traverses the "good" layers in non-increasing order. Assume $u \in B^r$. By Lemma 3.6, at the end of round r, all of u's *tight* edges are directed to B^r. Let ℓ be minimal such that $u_\ell \in V_0$. Hence, by Lemma 3.6 $u_\ell \in B^r$. Therefore $u_\ell \in GL_m^{r+1}$ for some $m < i$ (since $u_\ell \in V_0$ and $GL_i^{r+1} \subset V_1$), this contradicts our assumption $GL_m^{r+1} = GL_m^r$. Thus, $u \in G^r$.

Let j be maximal such that $u_j \in GL_i^{r+1} \setminus GL_i^r$. Hence, either $u_{j+1} \in GL_{i-1}^r$ or $u_{j+1} \in GL_i^r \cap GL_i^{r+1}$. In both cases $u_j, u_{j+1} \in G^r \cap G^{r+1}$ and thus (u_j, u_{j+1}) is *tight* in both rounds. Therefore $u_j \in GL_\ell^r$ for some $\ell \leq i$. Note that $\ell > i-1$ since otherwise $GL_\ell^r \neq GL_\ell^{r+1}$ which contradicts our assumption. Therefore $\ell = i$ and this contradict the assumption $u_j \in GL_i^{r+1} \setminus GL_i^r$. ◀

Note that if the conditions of Lemma 3.17 are satisfied, then by Lemma 3.17 and by the definition of b^r and g^r we have that $b^{r+1} \geq b^r$ and $g^{r+1} \geq g^r$.

▶ **Lemma 3.18.** *For every r, If $GA^{r+1} = GA^r$ and $BA^{r+1} = BA^r$, then $\phi^{r+1} \geq \phi^r + 2^{(n+k)/2}$, where $k = |GA^r| + |BA^r|$.*

Proof. Assume $|B^r \setminus BA^r| \geq |G^r \setminus GA^r|$, so $|G^r \setminus GA^r| \leq (n-k)/2$ and therefore g^r contains at least $(n+k)/2$ "padding bits". Hence, by Lemma 3.17 we have that $g^{r+1} \geq g^r + 2^{(n+k)/2}$ and $b^{r+1} \geq b^r$. Thus, $\phi^{r+1} \geq \phi^r + 2^{(n+k)/2}$ and we are done.

The case $|B^r \setminus BA^r| < |G^r \setminus GA^r|$ is identical. ◀

We are now ready to present the proof of our main result.

Proof of Theorem 3.11. By Lemma 3.18, there can be at most $2^{(n-k)/2}$ consecutive rounds satisfying $GA^{r+1} = GA^r$ and $BA^{r+1} = BA^r$, where $k = |GA^r| + |BA^r|$. Thus, by Lemma 3.15 we get that the following bounds the number of rounds during Update-Energy

$$\sum_{i_1=1}^{n} \sum_{i_1=1}^{n-i_1} 2^{(n-(i_1+i_2))/2} = O(2^{n/2}) \,,$$

where i_1 and i_2 represent $|GA^r|$ and $|BA^r|$, respectively. Hence, since a round takes $O(m)$ time and Compute-Energy calls Update-Energy at most $n \cdot \log W$ times, we get that Compute-Energy terminates in $O\left(mn \cdot 2^{n/2} \log W\right)$ time. ◀

4 Concluding remarks and open problems

We presented an $O\left(\min\left(mnW, mn2^{n/2} \log W\right)\right)$-time algorithm for solving EGs and MPGs. The algorithm is always at least as fast as the algorithm of Brim et al. [5], and is the fastest known deterministic algorithm when $W \geq n2^{n/2}$. (As mentioned the $\log W$ factor can be replaced by a $poly(n)$ factor.) The exponential running time of the new algorithm is

still far from what we would wish for. We hope, however, that the techniques used in our paper may lead to further improvements. The ultimate goal of using scaling is to obtain an algorithm whose running time is $O(poly(n) \log W)$. We are, of course, still extremely far from achieving this goal.

Many open problems remain: (1) Improve the pseudopolynomial running time to $O(mnf(W))$, where $f(W) = o(W)$. A more ambitious open problem is: (2) Obtain a deterministic sub-exponential time algorithm for solving EGs and MPGs, matching the running time of the fastest randomized algorithms. Even more ambitious open problem is: (3) obtain a quasipolynomial time algorithm for EGs and MPGs, matching the running time of the fastest algorithm for solving PGs. The most ambitious problem, of course, is: (4) obtain a polynomial time algorithm for PGs, EGs and MPGs.

References

1. Henrik Björklund and Sergei G. Vorobyov. Combinatorial structure and randomized subexponential algorithms for infinite games. *Theor. Comput. Sci.*, 349(3):347–360, 2005. doi:10.1016/j.tcs.2005.07.041.
2. Henrik Björklund and Sergei G. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. *Discrete Applied Mathematics*, 155(2):210–229, 2007.
3. Mikołaj Bojanczyk and Wojciech Czerwiński. An Automata Toolbox, February 2018. URL: https://www.mimuw.edu.pl/~bojan/papers/toolbox.pdf.
4. Patricia Bouyer, Uli Fahrenberg, Kim G Larsen, Nicolas Markey, and Jiří Srba. Infinite runs in weighted timed automata with energy constraints. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 33–47. Springer, 2008.
5. Lubos Brim, Jakub Chaloupka, Laurent Doyen, Raffaella Gentilini, and Jean-François Raskin. Faster algorithms for mean-payoff games. *Formal methods in system design*, 38(2):97–118, 2011.
6. Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In *Proc. of 49th STOC*, pages 252–263, 2017. doi:10.1145/3055399.3055409.
7. Arindam Chakrabarti, Luca De Alfaro, Thomas A Henzinger, and Mariëlle Stoelinga. Resource interfaces. In *International Workshop on Embedded Software*, pages 117–133. Springer, 2003.
8. Krishnendu Chatterjee, Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Polynomial-time algorithms for energy games with special weight structures. *Algorithmica*, 70(3):457–492, 2014.
9. A. Ehrenfeucht and J. Mycielski. Positional Strategies for Mean Payoff Games. *International Journal of Game Theory*, 8:109–113, 1979.
10. John Fearnley, Sanjay Jain, Sven Schewe, Frank Stephan, and Dominik Wojtczak. An ordered approach to solving parity games in quasi polynomial time and quasi linear space. In *Proceedings of the 24th ACM SIGSOFT International SPIN Symposium on Model Checking of Software*, pages 112–121. ACM, 2017.
11. Nathanaël Fijalkow, Paweł Gawrychowski, and Pierre Ohlmann. The complexity of mean payoff games using universal graphs. *CoRR*, abs/1812.07072, 2018. arXiv:1812.07072.
12. Harold N. Gabow and Robert Endre Tarjan. Faster Scaling Algorithms for General Graph-Matching Problems. *J. ACM*, 38(4):815–853, 1991. doi:10.1145/115234.115366.
13. Hugo Gimbert and Rasmus Ibsen-Jensen. A short proof of correctness of the quasi-polynomial time algorithm for parity games. *CoRR*, abs/1702.01953, 2017. arXiv:1702.01953.
14. Andrew V. Goldberg. Scaling Algorithms for the Shortest Paths Problem. *SIAM J. Comput.*, 24(3):494–504, 1995. doi:10.1137/S0097539792231179.
15. Andrew V. Goldberg and Satish Rao. Beyond the Flow Decomposition Barrier. *J. ACM*, 45(5):783–797, 1998. doi:10.1145/290179.290181.

16 Nir Halman. Simple stochastic games, parity games, mean payoff games and discounted payoff games are all LP-type problems. *Algorithmica*, 49(1):37–50, 2007.

17 Thomas Dueholm Hansen and Uri Zwick. An Improved Version of the Random-Facet Pivoting Rule for the Simplex Algorithm. In *Proc. of 47th STOC*, pages 209–218, 2015. doi:10.1145/2746539.2746557.

18 Marcin Jurdziński and Ranko Lazić. Succinct progress measures for solving parity games. In *Proc. of 32nd LICS*, pages 1–9, 2017. doi:10.1109/LICS.2017.8005092.

19 Gil Kalai. A Subexponential Randomized Simplex Algorithm (Extended Abstract). In *Proc. of 24th STOC*, pages 475–482, 1992. doi:10.1145/129712.129759.

20 Gil Kalai. Linear programming, the simplex algorithm and simple polytopes. *Math. Program.*, 79:217–233, 1997. doi:10.1007/BF02614318.

21 Karoliina Lehtinen. A modal μ perspective on solving parity games in quasi-polynomial time. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 639–648. ACM, 2018.

22 Yuri M Lifshits and Dmitri S Pavlov. Potential theory for mean payoff games. *Journal of Mathematical Sciences*, 145(3):4967–4974, 2007.

23 Walter Ludwig. A Subexponential Randomized Algorithm for the Simple Stochastic Game Problem. *Inf. Comput.*, 117(1):151–155, 1995. doi:10.1006/inco.1995.1035.

24 Jiří Matoušek, Micha Sharir, and Emo Welzl. A Subexponential Bound for Linear Programming. *Algorithmica*, 16(4/5):498–516, 1996. doi:10.1007/BF01940877.

25 Tibor Szabó and Emo Welzl. Unique Sink Orientations of Cubes. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 547–555, 2001. doi:10.1109/SFCS.2001.959931.

26 Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1-2):343–359, 1996.

Reachability for Branching Concurrent Stochastic Games

Kousha Etessami
School of Informatics, University of Edinburgh, UK
kousha@inf.ed.ac.uk

Emanuel Martinov
School of Informatics, University of Edinburgh, UK
eo.martinov@gmail.com

Alistair Stewart
Department of Computer Science, University of Southern California, Los Angeles, CA, USA
stewart.al@gmail.com

Mihalis Yannakakis
Department of Computer Science, Columbia University, New York City, NY, USA
mihalis@cs.columbia.edu

Abstract

We give polynomial time algorithms for deciding almost-sure and limit-sure reachability in Branching Concurrent Stochastic Games (BCSGs). These are a class of infinite-state imperfect-information stochastic games that generalize both finite-state concurrent stochastic reachability games ([8]) and branching simple stochastic reachability games ([13]).

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases stochastic games, multi-type branching processes, concurrent games, minimax-polynomial equations, reachability, almost-sure, limit-sure

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.115

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version A full preprint is available on arXiv at `https://arxiv.org/abs/1806.03907`.

Funding Research partially supported by an EPSRC studentship EP/N509644/1-1789473 (Martinov), and by NSF Grants CCF-1703925,CCF-1763970 (Yannakakis).

1 Introduction

Branching Processes (BP) are infinite-state stochastic processes that model the stochastic evolution of a population of entities of distinct types. In each generation, every entity of each type t produces, as offsprings, a set of entities of various types in the next generation, according to a given probability distribution on offsprings associated with type t. BPs are fundamental stochastic models that have been used to model phenomena in many fields, including biology (see, e.g., [24]), population genetics ([20]), physics and chemistry (e.g., particle systems, chemical chain reactions), medicine (e.g. cancer growth [2, 28]), marketing, and others. In many cases, the process is not purely stochastic but there is the possibility of taking actions (for example, adjusting the conditions of reactions, applying drug treatments in medicine, advertising in marketing, etc.) which can influence the probabilistic evolution of the process to bias it towards achieving desirable objectives. Some of the factors that affect the reproduction may be controllable (to some extent) while others are not and also may not be sufficiently well-understood to be modeled accurately by specific probability distributions, and thus it may be more appropriate to consider their effect in an adversarial (worst-case)

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 115; pp. 115:1–115:14


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

sense. *Branching Concurrent Stochastic Games* (BCSG) are a natural model to represent such settings. There are two players and, for each type t, each player has a set of available actions, which can affect the reproduction of entities of type t. The evolution of the process starts with some initial population of entities. For each entity of type t, the two players each select (concurrently and independently) an action from their available set (possibly in a randomized manner) and their join choices of actions determine the probability distribution for the offsprings of that entity. The first player represents a controller who can control some of the parameters of the reproduction and the second player represents other parameters that are not controlled and are treated adversarially. The first player wants to select a strategy that optimizes some objective. In this paper we focus on *reachability objectives*, a basic and natural class of objectives. Some types are designated as undesirable (for example, malignant cells), in which case we want to minimize the probability of ever reaching any entity of such a type. Or conversely, some types may be designated as desirable, in which case we want to maximize the probability of reaching an entity of such a type.

BCSGs generalize purely stochastic multi-type BPs as well as Branching Markov Decision Processes (BMDP) and Branching Simple Stochastic Games (BSSG) which were studied for reachability objectives in [13]. In BMDPs there is only one player who aims to maximize or minimize a reachability objective. In BSSGs there are two opposing players, but they control different types. These models were studied previously also under another basic objective, namely optimization of *extinction probability*, i.e., the probability that the process will eventually become extinct, that is, that the population will become empty [11, 15]. We will later discuss in detail the prior results and compare them with the results in this paper.

BCSGs can also be seen as a generalization of finite-state concurrent (stochastic) games [8] (see also [18]); namely they extend such finite-state games with branching. Concurrent games have been used in the verification area to model the dynamics of open systems, where one player represents the system and the other player the environment. Such a system moves sequentially from state to state depending on the actions of the two players (the system and the environment). Branching concurrent games model the more general setting in which processes can spawn new processes that then proceed independently in parallel (e.g., new threads are created and terminated). We note incidentally that even if there are no probabilities in the system itself, in the case of concurrent games, probabilities arise naturally from the fact that the optimal strategies are in general randomized; as a consequence it can be shown that branching concurrent stochastic games are expressively and computationally equivalent to the non-stochastic version (see [15]).

We now summarize our main results and compare and contrast them with previous results on related models. First, we show that a Branching concurrent stochastic game (BCSG), G, with a reachability objective has a well-defined *value*, i.e., given an initial (finite) population μ of entities of various types and a target type t^*, if the sets of all possible (mixed) strategies of the two players are respectively Ψ_1, Ψ_2, and if $\Upsilon_{\sigma,\tau}(\mu, t^*)$ denotes the probability of eventually reaching an entity of type t^* when starting from initial population μ under strategy $\sigma \in \Psi_1$ for player 1 and strategy $\tau \in \Psi_2$ for player 2, then $\inf_{\sigma\in\Psi_1} \sup_{\tau\in\Psi_2} \Upsilon_{\sigma,\tau}(\mu, t^*)$ $= \sup_{\tau\in\Psi_2} \inf_{\sigma\in\Psi_1} \Upsilon_{\sigma,\tau}(\mu, t^*)$, which is the *value*, v^*, of the game. Furthermore, we show that the player who wants to minimize the reachability probability always has an optimal (mixed) *static strategy* that achieves the value, i.e., a strategy σ^* which uses, for all entities of each type t generated over the entire history of the game, the same probability distribution on the available actions for type t, independent of the past history, and which has the property that $v^* = \sup_{\tau\in\Psi_2} \Upsilon_{\sigma^*,\tau}(\mu, t^*)$. The optimal strategy in general has to be mixed (randomized); this is the case even for finite-state concurrent games [8]. On the other hand,

the player that wants to maximize the reachability probability of a BCSG need not have any optimal strategy (whether static or not), and it was known that this holds even for BMDPs, i.e., even when there is only one player [13]. The same holds for finite-state CSGs: the player maximizing reachability probability need not have any optimal strategy [8].

To analyze BCSGs reachability games, we model them by a system of equations $x = P(x)$, called a *minimax Probabilistic Polynomial System* (minimax-PPS for short), where x is a tuple of variables corresponding to the types of the BCSG. There is one equation $x_i = P_i(x)$ for each type t_i, where $P_i(x)$ is the value of a (one-shot) two-player zero-sum matrix game, whose payoff for every pair of actions is given by a polynomial in x whose coefficients are positive and sum to at most 1 (a probabilistic polynomial). The function $P(x)$ defines a monotone operator from $[0,1]^n$ to itself, and thus it has, in particular, a *greatest fixed point* (GFP) g^* in $[0,1]^n$. We show that the coordinates g_i^* of the GFP give the optimal *non-reachability* probabilities for the BCSG game when started with a population that consists of a single entity of type t_i. The value of the game for any initial population μ can be derived easily from the GFP g^* of the minimax-PPS. This generalizes a result in [13], which established an analogous result for the special case of BSSGs. It also follows from our minimax-PPS equational characterization that *quantitative* decision problems for BCSGs, such as deciding whether the reachability game value is $\geq p$ for a given $p \in (0,1)$ are all solvable in PSPACE.

Our main algorithmic results concern the *qualitative* analysis of the reachability problem, that is, the problem of determining whether one of the players can win the game with probability 1, i.e., if the value of the game is 0 or 1. We provide the first polynomial-time algorithms for qualitative reachability analysis for branching concurrent stochastic games. For the value=0 problem, the algorithm and its analysis are very simple. If the value is 0, the algorithm computes an optimal strategy σ^* for the player that wants to minimize the reachability probability, where σ^* is in fact static and deterministic, i.e., it selects for each type, deterministically, a single available action, and guarantees $\Upsilon_{\sigma^*,\tau}(\mu, t^*) = 0$ for all $\tau \in \Psi_2$. If the value is positive then the algorithm computes a static mixed strategy τ for the player maximizing reachability probability that guarantees $\inf_{\sigma\in\Psi_1} \Upsilon_{\sigma,\tau}(\mu, t^*) > 0$.

The value=1 problem is much more complicated. There are two versions of it, because it is possible that the game value is 1 but no strategy for the maximizing player guarantees reachability with probability 1. Thus, we have two versions of the problem. In the first version, called the *almost-sure problem*, we want to decide whether there exists a strategy τ^* for player 2 that guarantees that the target type t^* is reached with probability 1 regardless of the strategy of player 1, i.e., such that $\Upsilon_{\sigma,\tau^*}(\mu, t^*) = 1$ for all $\sigma \in \Psi_1$. In the second version, called the *limit-sure problem*, we want to decide if the value $v^* = \sup_{\tau\in\Psi_2} \inf_{\sigma\in\Psi_1} \Upsilon_{\sigma,\tau}(\mu, t^*)$ is 1, i.e., if for every $\epsilon > 0$ there is a strategy τ_ϵ of player 2 that guarantees that the probability of reaching the target type is at least $1 - \epsilon$ regardless of the strategy σ of player 1; such a strategy τ_ϵ is called *ϵ-optimal*. We provide polynomial-time algorithms for both versions of the problem. The algorithms non-trivially generalize the algorithms of both [8] and [13], both of which address different special subcases of qualitative BCSG reachability. Our positive results on qualitative reachability for BCSG are surprising especially in view of the known major obstacles in resolving the analogous questions for the extinction problem for BCSG, as explained below in the review of related work.

In the almost-sure problem, if the answer is positive, our algorithm constructs (a compact description of) a strategy τ^* of player 2 that achieves value 1; the strategy is a randomized non-static strategy, and this is inherent (i.e., there may not exist a static strategy that achieves value 1). If the answer is negative, our algorithm constructs a (non-static, randomized)

strategy σ for the opposing player 1 such that $\Upsilon_{\sigma,\tau}(\mu, t^*) < 1$ for all strategies τ of player 2. In the limit-sure problem, if the answer is positive, i.e., the value is 1, our algorithm constructs for any given $\epsilon > 0$, a static, randomized ϵ-optimal strategy, i.e., a strategy τ_ϵ such that $\Upsilon_{\sigma,\tau_\epsilon}(\mu, t^*) \geq 1 - \epsilon$ for all $\sigma \in \Psi_1$. If the answer is negative, i.e., the value is < 1, our algorithm constructs a static randomized strategy σ' for player 1 such that $\sup_{\tau \in \Psi_2} \Upsilon_{\sigma',\tau} < 1$. Due to space limits, most proofs are omitted; see the full version.

Related Work. As mentioned, the two works most closely related to ours are [8] and [13]. Our results generalize both. de Alfaro, Henzinger, and Kupferman [8] studied finite-state concurrent (stochastic) games (CSGs) with reachability objectives and provided polynomial time algorithms for their qualitative analysis, both for the almost-sure and the limit-sure reachability problem. See also [22, 19, 21, 5] for more recent results on finite-state CSG reachability; in particular, there are finite-state CSGs with value$= 1$ for which (near-)optimal strategies need to have some action probabilities that are doubly-exponentially small [22, 5] (unlike simple, turned-based stochastic games), and thus must be represented succinctly to ensure polynomial space. This is of course the case also for branching CSGs, and the optimal or ϵ-optimal strategies constructed by our algorithms are represented compactly so that the algorithms run in polynomial time.

BMDPs and BSSGs with reachability objectives were studied in [13], which provided polynomial-time algorithms for their qualitative analysis, and also gave polynomial time algorithms for approximate quantitative analysis of BMDPs, i.e., approximate computation of the optimal reachability probability for maximizing and minimizing BMDPs, and it showed that this problem for BSSGs is in TFNP. Note that even for finite-state simple stochastic games the question of whether the value of the game can be computed in polynomial time is a well-known long-standing open problem [7]. It was also shown in [13] that the optimal non-reachability probabilities of maximizing or minimizing BMDPs and BSSGs are captured by the GFP of a system of min/max-PPS equations, $x = P(x)$, where each right-hand side $P_i(x)$ is the maximum or minimum of a set of probabilistic polynomials in x; note that these types of equation systems are special cases of minimax-PPSs and are much simpler.

Another important objective, the probability of *extinction*, has been studied previously for Branching Concurrent Stochastic Games, as well as BMDPs and BSSGs, and the purely stochastic model of Branching Processes (BPs). These branching models under the extinction objective are equivalent to corresponding subclasses of recursive Markov models, called respectively, 1-exit Recursive Concurrent Stochastic Games (1-RCSG), Markov Decision Processes (1-RMDP), and Markov Chains (1-RMC), and related subclasses of probabilistic pushdown processes under a termination objective [16, 12, 17, 11, 15, 10]. The extinction probabilities for these models are captured by the *least fixed point* (LFP) solutions of similar systems of probabilistic polynomial equations; for example, the optimal extinction probabilities of a BCSG are given by the LFP of a minimax-PPS. Polynomial time-algorithms for qualitative analysis, as well as for the approximate computation of the optimal extinction probabilities of Branching MDPs (and 1-RMDPs) were given in [17, 11]. However, negative results were shown also which indicate that the problem is much harder for branching concurrent (or even simple) stochastic games, even for the qualitative extinction problem. Specifically, it was shown in [17] that the qualitative extinction (termination) problem for BSSG (equivalently, 1-RSSG) is at least as hard as the well-known open problem of computing the value of a finite-state simple stochastic game [7]. Furthermore, it was shown in [15] that (both the almost-sure and limit-sure) qualitative extinction problems for BCSGs (equivalently 1-RCSGs) are at least as hard as the square-root sum problem, which is not even known to be

in NP. Thus, *the qualitative reachability problem for BCSGs seems to be very different than the extinction problem for BCSGs*: obtaining analogous results for the qualitative extinction to those of the present paper for qualitative reachability would resolve two major open problems. This is despite the fact that there is a natural "duality" between the extinction and reachability objectives (see [13]).

The equivalence between branching models (like BMDPs, BCSGs) and recursive Markov models (like 1-RMDPs, 1-RCSGs) with respect to extinction does *not* hold for the reachability objective. For example, almost-sure and limit-sure reachability coincide for a BMDP, i.e., if the supremum probability of reaching the target is 1 then there exists a strategy that ensures reachability with probability 1. However, this is not the case for 1-RMDPs. Furthermore, almost-sure reachability for 1-RMDPs can be decided in polynomial time [3, 4], but limit-sure reachability for 1-RMDPs is not even known to be decidable. The qualitative reachability problem for 1-RMDPs and 1-RSSGs (and equivalent probabilistic pushdown models) was studied in [4, 1]. These results do not apply to the corresponding branching models (BMDP, BSSG). Another objective considered in prior work is the *expected total reward* objective for 1-RSSGs ([14]) and 1-RCSGs ([31]) with positive rewards. None of these prior results have any implications for BCSGs with reachability objectives.

For richer objectives beyond reachability or extinction, Chen et. al. [6] studied model checking of purely stochastic branching processes (BPs) with respect to properties expressed by deterministic parity tree automata, and showed that the qualitative problem is in P-time (hence this holds in particular for reachability probability in BPs), and that the quantitative problem of comparing the probability with a rational is in PSPACE. Michalevski and Mio [25] extended this to properties of BPs expressed by "game automata", a subclass of alternating parity tree automata. More recently, Przybyłko and Skrzypczak [27] considered existence and complexity of game values of Branching turn-based (i.e., simple) stochastic games, with regular objectives, where the two players aim to maximize/minimize the probability that the generated labeled tree belongs to a regular language (given by a tree automaton). They showed that (unlike our case of simpler reachability games) already for some basic regular properties these games are not even determined, meaning they do not have a value. They furthermore showed that for a probabilistic turn-based branching game, with a regular tree objective, it is undecidable to compare the value that a given player can force to 1/2; whereas for deterministic turn-based branching games they showed it is decidable and 2-EXPTIME-complete (respectively, EXPTIME-complete), to determine whether the player aiming to satisfy (respectively, falsify) a given regular tree objective has a pure winning strategy. Other past research includes work in operations research on (one-player) Branching MDPs [26, 29, 9]. None of these prior works bear on any of the results on BCSG reachability problems established in this paper.

2 Background

A *Branching Concurrent Stochastic Game* (BCSG) consists of a finite set $V = \{T_1, \ldots T_n\}$ of types, two finite non-empty sets $\Gamma^i_{max}, \Gamma^i_{min} \subseteq \Sigma$ of actions (one for each player) for each type T_i (Σ is a finite action alphabet), and a finite set $R(T_i, a_{max}, a_{min})$ of probabilistic rules associated with each tuple (T_i, a_{max}, a_{min}), where $i \in [n]$, $a_{max} \in \Gamma^i_{max}$, & $a_{min} \in \Gamma^i_{min}$. Each rule $r \in R(T_i, a_{max}, a_{min})$ is a triple (T_i, p_r, α_r), which we can denote by $T_i \xrightarrow{p_r} \alpha_r$, where $\alpha_r \in \mathbb{N}^n$ is a n-vector of natural numbers that denotes a finite multi-set over the set V, and where $p_r \in (0, 1] \cap \mathbb{Q}$ is the probability of the rule r (which we assume to

be a rational number, for computational purposes), where we assume that for all $T_i \in V$ and $a_{max} \in \Gamma^i_{max}$, $a_{min} \in \Gamma^i_{min}$, the rule probabilities in $R(T_i, a_{max}, a_{min})$ sum to 1, i.e., $\sum_{r \in R(T_i, a_{max}, a_{min})} p_r = 1$.

If for all types $T_i \in V$, either $|\Gamma^i_{max}| = 1$ or $|\Gamma^i_{min}| = 1$, then the model is a "turn-based" perfect-information game and is called a *Branching Simple Stochastic Game* (BSSG). If for all $T_i \in V$, $|\Gamma^i_{max}| = 1$ (respectively, $|\Gamma^i_{min}| = 1$), then it is called a *minimizing Branching Markov Decision Process* (BMDP) (respectively, a *maximizing* BMDP). If both $|\Gamma^i_{min}| = 1 = |\Gamma^i_{max}|$ for all $i \in [n]$, then the process is a classic, purely stochastic, *multi-type Branching Process* (BP) ([23]).

A *play* of a BCSG defines a (possibly infinite) node-labeled forest, whose nodes are labeled by the type of the object they represent. A play contains a sequence of "generations", $X_0, X_1, X_2, \ldots$ (one for each integer time $t \geq 0$, corresponding to nodes at depth/level t in the forest). For each $t \in \mathbb{N}$, X_t consists of the population (set of objects of given types), at time t. X_0 is the initial population at generation 0 (these are the roots of the forest). X_{k+1} is obtained from X_k in the following way: for each object e in the set X_k, assuming e has type T_i, both players select simultaneously and independently actions $a_{max} \in \Gamma^i_{max}$, and $a_{min} \in \Gamma^i_{min}$ (or distributions on such actions), according to their strategies; thereafter a rule $r \in R(T_i, a_{max}, a_{min})$ is chosen randomly and independently (for object e) with probability p_r; each such object e in X_k is then replaced by the set of objects specified by the multi-set α_r associated with the corresponding randomly chosen rule r. This process is repeated in each generation, as long as the current generation is not empty, and if for some $k \geq 0$, $X_k = \emptyset$ then we say the process *terminates* or becomes *extinct*.

The strategies of players can in general be arbitrary functions from any finite *history* tree, to (distributions on) actions, for each object in the current population. The *history* of the process up to time k is a forest of depth k that includes not only the populations $X_0, X_1, \ldots, X_k$, but also all the information regarding past actions and rules applied at each object, and all the parent-child relationships between objects up to generation k. The history can be represented by a forest of depth k, with internal nodes labeled by rules and actions, and whose leaves at level k form the current population X_k. Thus, formally, a strategy of player 1 (player 2, respectively) is a function that maps every finite history (i.e., a labelled forest of some finite depth, k, as above) and each object e in the current population X_k (leaf at depth k) to a probability distribution on the actions Γ^i_{max} (to the actions Γ^i_{min}, respectively), assuming that object e has type T_i.[1]

Let Ψ_1, Ψ_2 be the set of all strategies of players 1, 2. We say that a strategy is *deterministic* if for every history it maps each object e in the current population to a single action with probability 1 (in other words, it does not randomize on actions). We say that a strategy is *static* if for each type $T_i \in V$, and for any object e of type T_i, the player always chooses the same distribution on actions, irrespective of the history.

Different objectives can be considered for BCSGs. This paper considers (existential) *reachability*, where the aim of the players is to maximize/minimize the probability of reaching a generation that contains at least one object of a given target type T_{f^*}. The BCSG reachability game can of course also be viewed as a "non-reachability" ("safety") game, by just reversing the role of the players. We will exploit this alternative view in crucial ways (and this was also exploited in [13] for BSSGs). Given an initial population $\mu \in \mathbb{N}^n$,

[1] Note: this very general notion of a "strategy" permits the action (or distribution on actions) chosen for a given object e to depend not only on e's "ancestors" in the history forest, but also on siblings, cousins, etc., in the entire forest, up to and including the generation of the population that e belongs to.

with $\mu_{f^*} = 0$, and given strategies $\sigma \in \Psi_1, \tau \in \Psi_2$, let $g^*_{\sigma,\tau}(\mu)$ denote the probability that $(X_l)_{f^*} = 0$ for all $l \geq 0$. Let $g^*_{\sigma,*}(\mu) = \inf_{\tau\in\Psi_2} g^*_{\sigma,\tau}(\mu)$, let $g^*_{*,\tau}(\mu) = \sup_{\sigma\in\Psi_1} g^*_{\sigma,\tau}(\mu)$, and let $g^*(\mu) = \sup_{\sigma\in\Psi_1} \inf_{\tau\in\Psi_2} g^*_{\sigma,\tau}(\mu)$ denote the *value* of the game under the non-reachability objective and for the initial population μ. We will show that these games do have a *value*, meaning $g^*(\mu) = \sup_{\sigma\in\Psi_1} \inf_{\tau\in\Psi_2} g^*_{\sigma,\tau}(\mu) = \inf_{\tau\in\Psi_2} \sup_{\sigma\in\Psi_1} g^*_{\sigma,\tau}(\mu)$.

In the case where the initial population μ is a single object of some type T_i, then for the value of the game we write g^*_i, and similarly, when one or both of the strategies σ, τ are fixed, we write $(g^*_{\sigma,*})_i$, $(g^*_{*,\tau})_i$, or $(g^*_{\sigma,\tau})_i$. The vector g^* of g^*_i's, is called the vector of the non-reachability values of the game. We will see that, having the vector of g^*_i's, the non-reachability value for any starting population μ can be computed simply as $g^*(\mu) = \prod_i (g^*_i)^{\mu_i}$. So given a BCSG, the goal is to compute the vector g^* of non-reachability values. As our original objective is reachability, we point out that the vector of reachability values is $r^* = \mathbf{1} - g^*$ (where $\mathbf{1}$ is the all-1 vector), and hence the reachability value $r^*(\mu)$ of the game starting with population μ is $r^*(\mu) = 1 - g^*(\mu)$.

We will associate with any given BCSG a system of *minimax probabilistic polynomial equations* (minimax-PPS), $x = P(x)$, for the non-reachability objective. This system will have one variable x_i, and one equation $x_i = P_i(x)$, for each type T_i other than the target type T_{f^*}. We will show that the vector of non-reachability values g^* for different starting types is precisely the Greatest Fixed Point (GFP) solution of the system $x = P(x)$ in $[0,1]^n$. To define these equations, some shorthand notation will be useful. We use x^v to denote the monomial $x_1^{v_1} x_2^{v_2} \cdots x_n^{v_n}$ for an n-vector of variables $x = (x_1, \cdots, x_n)$ and a vector $v \in \mathbb{N}^n$. Considering a multi-variate polynomial $P_i(x) = \sum_{r\in R} p_r x^{\alpha_r}$ for some rational coefficients $p_r, r \in R$, we will call $P_i(x)$ a *probabilistic polynomial*, if $p_r \geq 0$ for all $r \in R$ and $\sum_{r\in R} p_r \leq 1$. A *minimax probabilistic polynomial system of equations* (minimax-PPS), $x = P(x)$, is a system of n equations in n variables $x = (x_1, \ldots, x_n)$, where for each $i \in \{1, \ldots, n\}$, $P_i(x) := Val(A_i(x))$, where $A_i(x)$ is a matrix whose entries are probabilistic polynomials, and $Val(A_i(x))$ is the minimax value of the finite two-player zero-sum game with payoff matrix $A_i(x)$ for each $x \in \mathbb{R}^n$. Note that as special cases, when $A_i(x)$ has only one row or only one column, then $Val(A_i(x))$ is the maximum or minimum of a set of probabilistic polynomials, and when it has only one row and column, then $Val(A_i(x))$ is simply a probabilistic polynomial.

For computational purposes, we assume that all coefficients are rational and that there are no zero terms in the probabilistic polynomials, and we assume the coefficients and non-zero exponents of each term are given in binary. We denote by $|P|$ the total bit encoding length of a system $x = P(x)$ under this representation. Since $P(x)$ defines a monotone function $P : [0,1]^n \to [0,1]^n$, it follows from Tarski's theorem ([30])) that any such system has both a *Least Fixed Point* (LFP) solution $q^* \in [0,1]^n$, and a *Greatest Fixed Point* (GFP) solution, $g^* \in [0,1]^n$. In other words, $q^* = P(q^*)$ and $g^* = P(g^*)$ and moreover, for all $s^* \in [0,1]^n$ such that $s^* = P(s^*)$, we have $q^* \leq s^* \leq g^*$ (coordinate-wise inequality).

For convenience in proofs and algorithms throughout the paper and to simplify the structure of the matrices involved, we shall observe that minimax-PPSs can always be cast in the following normal form. A minimax-PPS in *simple normal form* (SNF), $x = P(x)$, is a system of n equations in n variables $\{x_1, \cdots, x_n\}$, where each $P_i(x)$ for $i = 1, 2, \ldots, n$ is one of three forms:

- Form L: $P_i(x) = a_{i,0} + \sum_{j=1}^n a_{i,j} x_j$, where for all j, $a_{i,j} \geq 0$, and $\sum_{j=0}^n a_{i,j} \leq 1$
- Form Q: $P_i(x) = x_j x_k$ for some j, k
- Form M: $P_i(x) = Val(A_i(x))$, where $A_i(x)$ is a $(n_i \times m_i)$ matrix, such that for all $j \in [n_i]$ and $k \in [m_i]$, the entry $(A_i(x))_{j,k} \in \{x_1, \ldots, x_n\} \cup \{1\}$.

We shall often assume a minimax-PPS in its SNF form, and say that a variable x_i is "of form/type" L, Q, or M, meaning that $P_i(x)$ has the corresponding form.

▶ **Proposition 1** (informal statement; see full version for formal statement and proof)**.** *Every minimax-PPS, $x = P(x)$, can be transformed in P-time to an "equivalent" minimax-PPS, $y = Q(y)$ in SNF form, such that $|Q| \in O(|P|)$.*

Thus, for the rest of this paper *we may assume, without loss of generality, that all minimax-PPSs are in SNF normal form.*

3 Non-reachability values for BCSGs and the Greatest Fixed Point

We show that for a given BCSG with a target type T_{f^*}, a minimax-PPS, $x = P(x)$, can be constructed such that its *greatest fixed point* (GFP) $g^* \in [0,1]^n$ is precisely the vector g^* of non-reachability values for the BCSG. For simplicity, from now on let us call a *maximizer* (respectively, a *minimizer*) the player that aims to *maximize* (respectively, *minimize*) the probability of *not* reaching the target type. That is, we swap the roles of the players for the benefit of less confusion in analysing the minimax-PPS which captures non-reachability values in its GFP.

For each type $T_i \neq T_{f^*}$, the minimax-PPS will have an associated variable x_i and an equation $x_i = P_i(x)$, and the $P_i(x)$ is defined as follows. For each action $a_{max} \in \Gamma^i_{max}$ of the maximizer (i.e., the player aiming to maximize the probability of *not* reaching the target) and action $a_{min} \in \Gamma^i_{min}$ of the minimizer, in T_i, let $R'(T_i, a_{max}, a_{min}) = \{r \in R(T_i, a_{max}, a_{min}) \mid (\alpha_r)_{f^*} = 0\}$ be the set of probabilistic rules r for type T_i and players' action pair (a_{max}, a_{min}) that generate a multi-set α_r which does not contain an object of the target type. For each action pair for T_i, there is a probabilistic polynomial $q_{i,a_{max},a_{min}}(x) := \sum_{r \in R'(T_i,a_{max},a_{min})} p_r x^{\alpha_r}$. Now we let $P_i(x) \equiv Val(A_i(x))$ be the value of a finite zero-sum game with matrix $A_i(x)$, where the matrix is constructed as follows: (1) rows belong to the max player in the minimax-PPS (i.e., the player trying to maximize the non-reachability probability), and columns belong to the min player; (2) for each row and column (i.e., pair of actions (a_{max}, a_{min})) the matrix entry $A_i(x)_{a_{max},a_{min}}$ is the corresponding probabilistic polynomial $q_{i,a_{max},a_{min}}(x)$.

The following theorem captures the fact that the optimal *non-reachability values* g^* in the BCSG game exist (meaning these game do have a value) and correspond to the GFP of the minimax-PPS $x = P(x)$ that was just defined.

▶ **Theorem 2.** *The non-reachability game values $g^* \in [0,1]^n$ of a BCSG reachability game exist, and correspond to the GFP of the minimax-PPS, $x = P(x)$, in $[0,1]^n$. That is, $g^* = P(g^*)$, and for all other fixed points $g' = P(g')$ in $[0,1]^n$, it holds that $g' \leq g^*$. Moreover, for an initial population μ, the optimal non-reachability value is $g^*(\mu) = \prod_i (g^*_i)^{\mu_i}$ and the game is determined, i.e., $g^*(\mu) = \sup_{\sigma \in \Psi_1} \inf_{\tau \in \Psi_2} g^*_{\sigma,\tau}(\mu) = \inf_{\tau \in \Psi_2} \sup_{\sigma \in \Psi_1} g^*_{\sigma,\tau}(\mu)$. Finally, the player maximizing non-reachability probability in the BCSG has a (mixed) static optimal strategy.*

▶ **Corollary 3.** *Given a BCSG reachability game, and a probability $p \in (0,1)$, deciding whether the game value is $\geq p$ is in PSPACE.*

The PSPACE upper bound follows from Theorem 2, by appealing to decision procedures for the (existential) theory of reals to answer quantitative questions about the GFP of the corresponding minimax-PPS equations. This is entirely analogous to very similar arguments in [15, 13, 16], so we do not elaborate. Any substantial improvement on PSPACE for such quantitative decision problems would require a major breakthrough on exact numerical computation, even for BPs or BMDPs (see [16, 13, 15]).

4 P-time algorithm for almost-sure reachability for BCSGs

Before considering almost-sure reachability, we first show there is a simple P-time algorithm for computing the variables x_i with value $q_i^* = 1$ for the GFP in a given minimax-PPS, $x = P(x)$, or in other words, for a given BCSG, deciding whether the reachability value, starting with an object of a given type T_i, is 0. The algorithm is easy, amounts to an AND-OR graph reachability analysis, and is very similar to the algorithm given for deciding $q_i^* = 1$ for BSSGs in [13]. Let $W = \{x_1, \ldots, x_n\}$ denote the set of all variables of the minimax-PPS. The algorithm is given in Figure 1.

1. Initialize $S := \{x_i \in W \mid P_i(\mathbf{1}) < 1\}$.
2. Repeat until no change has occurred:
   a. if there is a variable $x_i \notin S$ of form L or Q such that $P_i(x)$ contains a variable already in S, then add x_i to S.
   b. if there is a variable $x_i \notin S$ of form M such that for every action $a_{max} \in \Gamma^i_{max}$, there exists an action $a_{min} \in \Gamma^i_{min}$, such that $A_i(x)_{(a_{max},a_{min})} \in S$, then add x_i to S.
3. Output the set $\bar{S} := W - S$.

Figure 1 Simple P-time algorithm for computing the set of types with reachability value 0 in a given BCSG, or equivalently the set of variables $\{x_i \mid q_i^* = 1\}$ of the associated minimax-PPS.

▶ **Proposition 4.** *There is a P-time algorithm that, given a BCSG or equivalently a corresponding minimax-PPS, $x = P(x)$, with n variables and with GFP $q^* \in [0,1]^n$, and given $i \in [n]$, determines whether $q_i^* = 1$ or $q_i^* < 1$. In case $q_i^* = 1$, the algorithm can produce a deterministic static strategy σ for the max player (maximizing non-reachability) that forces $q_i^* = 1$. Otherwise, if $q_i^* < 1$, the algorithm can produce a mixed static strategy τ for the min player (minimizing non-reachability) that guarantees $q_i^* < 1$.*

We are now ready to present a P-time algorithm for almost-sure reachability for BCSGs. First, as a preprocessing step, we apply the algorithm of Proposition 4, which identifies in P-time all the variables x_i where $q_i^* = 1$. We then remove these variables from the system, substituting the value 1 in their place. We then simplify and reduce the resulting SNF-form minimax-PPS into a reduced form, with GFP $q^* < 1$. Note that the resulting reduced SNF-form minimax-PPS may contain some variables x_j of form M, whose corresponding matrix $A_j(x)$ has some entries that contain the value 1 rather than a variable (because we substituted 1 for removed variables x_j, where $q_j^* = 1$). Note also that in the reduced SNF-form minimax-PPS each variable x_i of form Q has an associated quadratic equation $x_i = x_j x_k$, because if one of the variables (say x_k) on the right-hand side was set to 1 during preprocessing, the resulting equation ($x_i = x_j$) would have been declared to have form L in the reduced minimax-PPS. We henceforth assume that the minimax-PPS is in SNF-form, with $q^* < 1$, and we let X be its set of (remaining) variables. We now apply the algorithm of Figure 2 to the minimax-PPS with $q^* < 1$, which identifies the variables x_i in the minimax-PPS (equivalently, the types in the BCSG), from which we can almost-surely reach the target type T_{f^*} (i.e., $q_i^* = 0$ *and* there is a strategy τ^* for the player minimizing non-reachability probability that achieves this value, no matter what the other player does).

▶ **Theorem 5.** *Given a BCSG with minimax-PPS, $x = P(x)$, such that the GFP $q^* < \mathbf{1}$, the algorithm in Figure 2 terminates in polynomial time and returns the set of variables $\{x_i \in X \mid \exists \tau \in \Psi_2\ (q^*_{*,\tau})_i = 0\}$.*

1. Initialize $S := \{x_i \in X \mid P_i(\mathbf{0}) > 0$, that is $P_i(x)$ has a constant term $\}$.
   Let $\gamma_0^i := \Gamma_{min}^i$ for every variable $x_i \in X - S$. Let $t := 1$.
2. Repeat until no change has occurred to S:
   a. if there is a variable $x_i \in X - S$ of form L where $P_i(x)$ contains a variable already in S, then add x_i to S.
   b. if there is a variable $x_i \in X - S$ of form Q where both variables in $P_i(x)$ are already in S, then add x_i to S.
   c. if there is a variable $x_i \in X - S$ of form M and if for all $a_{min} \in \Gamma_{min}^i$, there exists a $a_{max} \in \Gamma_{max}^i$ such that $A_i(x)_{(a_{max},a_{min})} \in S \cup \{1\}$, then add x_i to S.
3. For each $x_i \in X - S$ of form M, let:
   $\gamma_t^i := \{a_{min} \in \gamma_{t-1}^i \mid \forall a_{max} \in \Gamma_{max}^i,\ A_i(x)_{(a_{max},a_{min})} \notin S \cup \{1\}\}$. (Note that $\gamma_t^i \subseteq \gamma_{t-1}^i$.)
4. Let $F := \{x_i \in X - S \mid P_i(\mathbf{1}) < 1$, or $P_i(x)$ is of form Q $\}$
5. Repeat until no change has occurred to F:
   a. if there is a variable $x_i \in X - (S \cup F)$ of form L where $P_i(x)$ contains a variable already in F, then add x_i to F.
   b. if there is a variable $x_i \in X - (S \cup F)$ of form M such that for $\forall a_{max} \in \Gamma_{max}^i$, there is a min player's action $a_{min} \in \gamma_t^i$ such that $A_i(x)_{(a_{max},a_{min})} \in F$, then add x_i to F.
6. If $X = S \cup F$, **return** F, and halt.
7. Else, let $S := X - F$, $t := t + 1$, and go to step 2.

Figure 2 P-time algorithm for computing almost-sure reachability types $\{x_i \mid \exists \tau \in \Psi_2\ (g_{*,\tau}^*)_i = 0\}$ for a minimax-PPS (in SNF), associated with a given BCSG.

The proof is in the full version. Here we give very brief intuition for why the algorithm works. The set S will accumulate variables $x_i \in X$, such that regardless of the strategy $\tau \in \Psi_2$ of the player minimizing non-reachability (i.e., maximizing reachability), the probability of reaching the target type is < 1. The loop in Step (2.) is a basic "attractor set" construction that adds to S any variable x_i that should be in S by virtue of prior membership in S of variables (types) occurring in $P_i(x)$. In step (3.), for each variable $x_i \in X - S$ we maintain the remaining "useful" set of actions $\gamma_t^i \subseteq \Gamma_{min}^i$ that can avoid the set S (or extinction). The loop in Step (5.) accumulates a set $F \subseteq X - S$, such that for every $x_i \in F$ there is a strategy $\tau \in \Psi_2$ to either reach the target or a branching (quadratic) type in F, with positive probability, regardless of the opponent's strategy. The key assertion is this: if in step (6.) we find all variables are already either in S or in F, we are done; F must be the set of types from which we can force almost-sure reachability of the target type; otherwise, all variables in $X - (F \cup S)$ can be added to S. The reason this assertion holds is not obvious (see the detailed proof). The proof of the theorem also yields the following:

▶ **Corollary 6.** *Let F be the set of variables output by the algorithm in Figure 2.*

1. *Let $S = X - F$. There is a randomized non-static strategy $\hat{\sigma}$ for the max player (maximizing non-reachability) such that for all $x_i \in S$, and for all strategies τ of the min player (minimizing non-reachability), starting with one object of type T_i the probability of reaching the target type is < 1.*
2. *There is a randomized non-static strategy $\hat{\tau}$ for the min player (minimizing non-reachability) such that for all strategies σ of the max player (maximizing non-reachability), and for all $x_i \in F$, starting at one object of type T_i the probability of reaching the target type is 1.*

The non-static strategies $\hat{\sigma}$ and $\hat{\tau}$ mentioned above both have suitably compact descriptions (as functions that map finite histories to distributions over actions for entities in the current population) and can be described in such a compact form in polynomial time, as a function of the encoding size of the input BCSG.

The strategies need to be represented compactly because some of the actions probabilities may be doubly-exponentially small (or doubly-exponentially close to 1), and this is inherent.

5 P-time algorithm for limit-sure reachability for BCSGs

In this section we focus on the limit-sure reachability problem, i.e., starting with one object of a type T_i, decide whether the reachability value is 1. Since we translate reachability into non-reachability when analysing the corresponding minimax-PPS, we are asking whether there exists a sequence of strategies $\langle \tau^*_{\epsilon_j} \mid j \in \mathbb{N} \rangle$ for the min player, such that $\forall j \in \mathbb{N}$, $\epsilon_j > \epsilon_{j+1} > 0$, and where $\lim_{j\to\infty} \epsilon_j = 0$, such that the strategy $\tau^*_{\epsilon_j}$ forces non-reachability probability to be at most ϵ_j, regardless of the strategy σ used by the max player. In other words, for a given starting object of type T_i, we ask whether $\inf_{\tau\in\Psi_2}(g^*_{*,\tau})_i = 0$.

1. Initialize $S := \{x_i \in X \mid P_i(\mathbf{0}) > 0$, that is $P_i(x)$ has a constant term $\}$.
2. Repeat until no change has occurred to S:
   a. if there is a variable $x_i \in X - S$ of form L where $P_i(x)$ contains a variable already in S, then add x_i to S.
   b. if there is a variable $x_i \in X - S$ of form Q where both variables in $P_i(x)$ are already in S, then add x_i to S.
   c. if there is a variable $x_i \in X - S$ of form M and if for all $a_{min} \in \Gamma^i_{min}$, there exists $a_{max} \in \Gamma^i_{max}$ such that $A_i(x)_{(a_{max},a_{min})} \in S \cup \{1\}$, then add x_i to S.
3. Let $F := \{x_i \in X - S \mid P_i(\mathbf{1}) < 1$, or $P_i(x)$ is of form Q $\}$
4. Repeat until no change has occurred to F:
   a. if there is a variable $x_i \in X - (S \cup F)$ of form L where $P_i(x)$ contains a variable already in F, then add x_i to F.
   b. if there is a variable $x_i \in X - (S \cup F)$ of form M and if the following procedure returns "Yes", then add x_i to F.
      i. Set $L_0 := \emptyset$, $B_0 := \emptyset$, $k := 0$. Let $O := X - (S \cup F)$.
      ii. Repeat:
         - $k := k + 1$.
         - $L_k := \{a_{min} \in \Gamma^i_{min} - \bigcup_{j=0}^{k-1} L_j \mid \forall a_{max} \in \Gamma^i_{max} - B_{k-1},\ A_i(x)_{(a_{max},a_{min})} \in F \cup O\}$.
         - $B_k := B_{k-1} \cup \{a_{max} \in \Gamma^i_{max} - B_{k-1} \mid \exists a_{min} \in L_k \ s.t.\ A_i(x)_{(a_{max},a_{min})} \in F\}$.

         Until $B_k = B_{k-1}$.
      iii. Return: "Yes" if $B_k = \Gamma^i_{max}$, and "No" otherwise.
5. If $X = S \cup F$, **return** F, and halt.
6. Else, let $S := X - F$, and go to step 2.

Figure 3 P-time algorithm for computing the set of types that satisfy limit-sure reachability in a given BCSG, i.e., the set of variables $\{x_i \mid g^*_i = 0\}$ in the associated minimax-PPS.

Again, as in the almost-sure case, we first, as a preprocessing step, use the P-time algorithm from Proposition 4 to remove all variables x_i such that $g^*_i = 1$, and we substitute 1 for these variables in the remaining equations. We hence obtain a reduced SNF-form

minimax-PPS, for which we can assume $g^* < 1$. The set of all remaining variables in the SNF-form minimax-PPS is again denoted by X. Thereafter, we apply the algorithm in Figure 3, which computes the set of variables, x_i, such that $g^*_i = 0$. In other words, we compute the set of types, such that starting from one object of that type the value of the reachability game is 1.

▶ **Theorem 7.** *Given a BCSG with minimax-PPS, $x = P(x)$, with GFP $g^* < \mathbf{1}$, the algorithm in Figure 3 terminates in polynomial time, and returns the set of variables $\{x_i \in X \mid g^*_i = 0\}$.*

The proof is in the full version. We again give very brief intuition for why the algorithm works. The algorithm is similar in structure to that of almost-sure reachability, but differs in crucial aspects. Firstly, step (2.) now accumulates the variables x_i for which we know $g^*_i > 0$. In step (3.), as before, the set F is initialized to variables $x_i \in X - S$ (types) where either $P_i(x)$ is quadratic, or the target type is generated with positive probability in the next step, i.e., $(P_i(\mathbf{1}) < 1)$. In step (4.) the algorithm makes crucial use of a "limit-escape" set construction (inside inner loop 4.(b)), a variant of which was used by de Alfaro, Henzinger, and Kupferman in [8], in the context of finite-state CSGs, but which has been adapted here to the context of BCSGs. This loop adds a variable x_i to the set F whenever it is the case that, for any $\epsilon > 0$, the player minimizing non-reachability can play a distribution on actions (depending on ϵ) at any object e of type T_i such that (regardless of the action distribution of the other player) the probability that e produces an offspring in the next generation whose type is in S is at most ϵ times the probability that it produces an offspring that is already in F. Unlike the almost-sure case (where we maintain remaining sets of "useful" actions $\gamma^i_t \subset \Gamma^i_{min}$ that can avoid the set S), in the limit-sure case the player minimizing non-reachability may not have any (distribution on) actions that entirely avoid the set S, but it nevertheless may have a series of distributions on actions that make the ratio of the probability of the "bad" event of generating an offspring in S, compared to the probability of the "good" event of generating an offspring in F, arbitrarily small. Similar to the case of almost-sure reachability, a key assertion is that if in step (5.) all variables in X are already in $S \cup F$ then we are done: F consists of precisely those variables (types) that satisfy limit-sure reachability; otherwise, we can add $X - (S \cup F)$ to the set S. The reason why this holds is again subtle (see the detailed proof in the full version).

The proof of the theorem also yields the following corollary:

▶ **Corollary 8.** *Suppose the algorithm in Figure 3 outputs the set F when it terminates. Let $S := X - F$.*

1. *There is a randomized static strategy $\hat{\sigma}$ for the max player (maximizing non-reachability) such that for all variables $x_i \in S$, we have $(g^*_{\hat{\sigma},*})_i > 0$.*
2. *For all $\epsilon > 0$, there is a randomized static strategy τ_ϵ, for the min player (minimizing non-reachability), such that for all variables $x_i \in F$, $(g^*_{*,\tau_\epsilon})_i \leq \epsilon$.*

The static strategies $\hat{\sigma}$ and τ_ϵ mentioned above can both be described, in a suitably compact form, in polynomial time, as a function of the encoding size of the input BCSG. However, these static strategies, specifically in the case of τ_ϵ, involve probabilities that are double-exponentially small (and double-exponentially close to 1), as a function of the encoding size of the BCSG, so these probabilities have to be encoded in a suitable succinct notation in order for the output to have polynomial encoding size.

References

1 R. Bonnet, S. Kiefer, and A. W. Lin. Analysis of Probabilistic Basic Parallel Processes. In *Proc. of FoSSaCS'14*, pages 43–57, 2014.

2 Bozic and et. al. Evolutionary dynamics of cancer in response to targeted combination therapy. *eLife*, 2:e00747, 2013.

3 T. Brázdil, V. Brozek, V. Forejt, and A. Kucera. Reachability in recursive markov decision processes. *Inf. Comput.*, 206(5):520–537, 2008.

4 T. Brázdil, V. Brozek, A. Kucera, and J. Obdrzálek. Qualitative reachability in stochastic BPA games. *Inf. Comput.*, 209(8):1160–1183, 2011.

5 K. Chatterjee, K. A. Hansen, and R. Ibsen-Jensen. Strategy complexity of concurrent safety games. In *Proc. of 42nd Inter. Symp. on Math. Found. of Computer Science (MFCS)*, volume 83 of *LIPIcs*, pages 55:1–55:13, 2017.

6 T. Chen, K. Dräger, and S. Kiefer. Model Checking Stochastic Branching Processes. In *Proc. of MFCS'12*, volume 7464 of *Springer LNCS*, pages 271–282, 2012.

7 A. Condon. The complexity of stochastic games. *Inf. & Comput.*, 96(2):203–224, 1992.

8 L. de Alfaro, T. A. Henzinger, and O. Kupferman. Concurrent reachability games. *Theoretical Computer Science*, 386(3):188–217, 2007. (Conference version in FOCS'98).

9 E. Denardo and U. Rothblum. Totally expanding multiplicative systems. *Linear Algebra Appl.*, 406:142–158, 2005.

10 J. Esparza, A. Kučera, and R. Mayr. Model checking probabilistic pushdown automata. *Logical Methods in Computer Science*, 2(1):1–31, 2006.

11 K. Etessami, A. Stewart, and M. Yannakakis. Polynomial-time algorithms for Branching Markov Decision Processes, and probabilistic min(max) polynomial Bellman equations. In *Proc. of 39th Int. Coll. on Automata, Languages and Programming (ICALP)*, 2012. (All references are to the full preprint Arxiv:1202.4789).

12 K. Etessami, A. Stewart, and M. Yannakakis. A polynomial-time algorithm for computing extinction probabilities of multitype branching processes. *SIAM J. Computing*, 46(5):1515–1553, 2017. (Conference version in STOC'12).

13 K. Etessami, A. Stewart, and M. Yannakakis. Greatest Fixed Points of Probabilistic Min/Max Polynomial Equations, and Reachability for Branching Markov Decision Processes. *Information and Computation*, 261:355–382, 2018. (special issue for ICALP'15).

14 K. Etessami, D. Wojtczak, and M. Yannakakis. Recursive stochastic games with positive rewards. In *Proc. of 35th ICALP (1)*, volume 5125 of *Springer LNCS*, pages 711–723, 2008.

15 K. Etessami and M. Yannakakis. Recursive Concurrent Stochastic Games. *Logical Methods in Computer Science*, 4(4), 2008.

16 K. Etessami and M. Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. *Journal of the ACM*, 56(1), 2009.

17 K. Etessami and M. Yannakakis. Recursive Markov decision processes and recursive stochastic games. *Journal of the ACM*, 62(2):1–69, 2015.

18 H. Everett. Recursive games. *Contributions to the Theory of Games*, 3(39):47–78, 1957.

19 S. K. S Frederiksen and P. B. Miltersen. Approximating the Value of a Concurrent Reachability Game in the Polynomial Time Hierarchy. In *Proc. of 24th ISAAC*, pages 457–467, 2013.

20 P. Haccou, P. Jagers, and V. A. Vatutin. *Branching Processes: Variation, Growth, and Extinction of Populations*. Cambridge U. Press, 2005.

21 K. A. Hansen, R. Ibsen-Jensen, and P. B. Miltersen. The Complexity of Solving Reachability Games using Value and Strategy Iteration. *Theory Comput. Syst.*, 55(2):380–403, 2014.

22 K. A. Hansen, M. Koucky, and P. B. Miltersen. Winning Concurrent Reachability Games Requires Doubly-Exponential Patience. In *Proc. of 24th Annual IEEE Symp. on Logic in Computer Science*, pages 332–341, 2009.

23 T. E. Harris. *The Theory of Branching Processes*. Springer-Verlag, 1963.

24 M. Kimmel and D. E. Axelrod. *Branching processes in biology*. Springer, 2002.

25 H. Michalewski and M. Mio. On the problem of computing the probability of regular sets of trees. In *Proc. of FSTTCS'15*, pages 489–502, 2015.

26 S. Pliska. Optimization of multitype branching processes. *Management Sci.*, 23(2):117–124, 1976/77.

27 M. Przybyłko and M. Skrzypczak. On the complexity of branching games with regular conditions. In *Proc. of MFCS'16*, volume 78 of *LIPIcs*, 2016.

28 G. Reiter, I. Bozic, K. Chatterjee, and M. A. Nowak. TTP: Tool for tumor progression. In *Proc. of CAV'2013*, volume 8044 of *Springer LNCS*, pages 101–106, 2013.

29 U. Rothblum and P. Whittle. Growth optimality for branching Markov decision chains. *Math. Oper. Res.*, 7(4):582–601, 1982.

30 A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.

31 D. Wojtczak. Expected termination time in BPA games. In *Proc of ATVA'2013*, pages 303–318, 2013.

FO = FO^3 for Linear Orders with Monotone Binary Relations

Marie Fortin
LSV, CNRS & ENS Paris-Saclay, Université Paris-Saclay, France
fortin@lsv.fr

Abstract

We show that over the class of linear orders with additional binary relations satisfying some monotonicity conditions, monadic first-order logic has the three-variable property. This generalizes (and gives a new proof of) several known results, including the fact that monadic first-order logic has the three-variable property over linear orders, as well as over $(\mathbb{R}, <, +1)$, and answers some open questions mentioned in a paper from Antonopoulos, Hunter, Raza and Worrell [FoSSaCS 2015]. Our proof is based on a translation of monadic first-order logic formulas into formulas of a star-free variant of Propositional Dynamic Logic, which are in turn easily expressible in monadic first-order logic with three variables.

2012 ACM Subject Classification Theory of computation → Logic

Keywords and phrases first-order logic, three-variable property, propositional dynamic logic

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.116

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version `https://arxiv.org/abs/1904.00189`

1 Introduction

Logics with a bounded number of variables have been extensively studied, in particular in the context of descriptive complexity [17, 18, 10, 21] and temporal logics [20, 7, 14, 16]. One recurring question of interest [7, 25, 19, 4, 26, 1] is to determine, in a given class $\mathcal{C}$ of structures, whether all properties expressible in monadic first-order logic (FO) can be defined in the fragment FO^k consisting of formulas which use at most k variables. (A same variable may be quantified over several times in a formula.) In fact, several non-equivalent versions of this question appear in the literature, many of which are compared in [15]. We say that $\mathcal{C}$ has the *k-variable property* if every formula of FO with at most k free variables is equivalent over $\mathcal{C}$ to a formula of FO^k. Note that this is strictly stronger than requiring that all *sentences* (without free variables) of FO are equivalent to some FO^k formulas. Indeed, Hodkinson and Simon gave an example of a class of structures where no sentence requires more than 3 variables, but which does not have the k-variable property for any k [15].

The problem of whether a given class of structures has the k-variable property is closely related to the question of the existence of an expressively complete temporal logic (with a finite set of FO-definable modalities). A temporal logic is called expressively complete if any first-order formula with a single free variable can be expressed in it. For instance, it is well-known that linear temporal logic (LTL) over Dedekind-complete time flows, or its extension with Stavi connectives over all time flows, are expressively complete for first-order logic [20, 8]. More recently, it was shown that over the real numbers equipped with binary relations $+q$ for all $q \in \mathbb{Q}$, metric temporal logic (MTL) is expressively complete [16]. However, the questions of having the k-variable property for some k or admitting an expressively complete temporal logic are incomparable in general: there exists a class of structures which admits a finite expressively complete set of temporal connectives but which does not have the k-variable

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 116; pp. 116:1–116:13


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

property for any k [15], and there exists one which has the 3-variable property but for which no temporal logic is expressively complete [14]. However, Gabbay established that having the k-variable property implies the existence of a *multi-dimensional* expressively complete temporal logic, with multiple reference points [7].

Another classical approach to proving or disproving that a class of structures has the k-variable property is through Ehrenfeucht-Fraïssé games, with a bounded number of pebbles [12, 25, 19, 1]. Immerman and Kozen applied this to linear orders and bounded-degree trees [19], and Antonopoulos et al. to real-time signals [1].

Natural candidates for classes $\mathcal{C}$ which might have the k-variable property are classes of linearly ordered structures. Indeed, a typical counter-example to unrestricted structures having the k-variable property is a formula such as "there exists $k+1$ distinct elements which satisfy some predicate P". It is in general not expressible in FO^k, but it is easily expressible in FO^2 if all models are equipped with a linear order $<$. For instance for $k=2$, we take the formula $\exists x.\ P(x) \wedge \exists y.(x < y \wedge P(y) \wedge \exists x.(y < x \wedge P(x)))$. As mentioned before, Immerman and Kozen showed that the class of linear orders has the 3-variable property [19]. However, adding a single binary relation suffices to obtain a class of linearly ordered structures which does not have the k-variable property for any k. Venema gave an example of a dense linear order with a single equivalence relation which does not have the k-variable property for any k [29]; this was adapted in [1] to give another example where the equivalence relation is replaced with a bijection. In fact, even for finite linear orders, Rossman [26] proved that the class of linearly ordered graphs does not have the k-variable property for any k, resolving a problem which had been open for more than 25 years [17]. Therefore, adding binary relations to linear orders while keeping the k-variable property requires some restrictions on the interpretation of the relation symbols.

On the positive side, Antonopoulos et al. proved that the class of structures over $(\mathbb{R}, <, +1)$ (or signals) has the 3-variable property [1]. Such structures have been studied in the context of real-time verification. As a corollary, they also showed that $(\mathbb{R}, <, f)$ has the 3-variable property for any linear function $f : x \mapsto ax + b$.

Contribution. We consider the class of linearly ordered structures with an additional (finite or infinite) number of binary *interval-preserving* relations. These are binary relations $\mathcal{R}$ such that, for all intervals I, any point which is in between two points of $\mathcal{R}(I)$ and has a preimage by $\mathcal{R}$ must have one in I. (We also require a symmetric condition of the converse relation $\mathcal{R}^{-1}$.) We show that FO over this class of structures also has the 3-variable property.

This generalizes results from [19] and [1] described above. Moreover, this answers some open questions mentioned in the conclusion of [1], which asked if the result could be extended from linear functions to polynomials over the reals, or other linear orders and families of monotone functions. In fact, all increasing or decreasing partial functions (over arbitrary linear orders) are special cases of interval-preserving relations, and thus covered by our result.

Our proof relies on different techniques than [19, 1], which were based on Ehrenfeucht-Fraïssé games. We give an effective translation from FO to FO^3 which goes through a star-free variant of Propositional Dynamic Logic (PDL) with converse. Propositional dynamic logic was introduced by Fischer and Ladner [6] to reason about program schemes, and has now found a large range of applications in artificial intelligence and verification [11, 5, 23, 22, 9]. It combines local formulas containing modal operators, and path formulas using the concatenation, union and Kleene star operations. Several extensions have been studied, including PDL with converse [27], intersection [3], or negation of atomic programs

[24]. The particular *star-free* variant of PDL we use here is in fact very similar to Tarski's relation algebras [28], which was used as a basis for formalizing set theory. It also corresponds to a *two-dimensional* temporal logic in the sense of Gabbay [7].

We applied similar proof techniques in [2], where we introduced a star-free variant of PDL and proved that it is equivalent to FO over *message sequence charts* (MSCs) (and thus obtained a 3-variable property result for MSCs as a corollary). MSCs are discrete partial orders which represent behaviors of concurrent message passing systems. They consist of a fixed, finite number of linear orders called process orders (one for each process in the system), together with FIFO binary message relations connecting matching send and receive actions. Having a (fixed) finite number of total orders instead of a single one is not an important difference, as we could always put them one after the other to extend them into a single linear order. FIFO relations are a special case of interval-preserving relations, thus the result of the present paper can in fact be seen as a strict generalization of our previous result in [2]. More importantly, a major difference between MSCs studied in [2] and the setting we consider here is that MSCs are *discrete* structures, whereas here we allow arbitrary linear orders. In fact, [2] relied on the definition of formulas describing the minimum or the maximum of some binary relations. As such, it is interesting to see that the same kind of techniques can still be applied to a priori very different linear orders.

Outline. In Section 2, we introduce interval-preserving relations and monadic first-order logic. In Section 3, we define star-free PDL, and prove some properties of its formulas. In Section 4, we give an effective translation from FO to star-free PDL, and explain its consequences. We conclude in Section 5.

2 Interval-preserving relations and first-order logic

In this section, we define the class of structures covered by our results, and recall the syntax of first-order logic.

Interval-preserving binary relations. Let $\mathcal{R} \subseteq A \times B$ be a binary relation between sets A and B. We write $a\ \mathcal{R}\ b$ if $(a,b) \in \mathcal{R}$, and $\mathcal{R}(a) = \{b \in B \mid a\ \mathcal{R}\ b\}$. For a subset $A' \subseteq A$, we also write $\mathcal{R}(A') = \bigcup_{a \in A'} \mathcal{R}(a)$. We define the *converse* of a relation $\mathcal{R}$ as $\mathcal{R}^{-1} = \{(b,a) \in B \times A \mid (a,b) \in \mathcal{R}\}$, and the *composition* of two binary relations $\mathcal{R}_1 \subseteq A \times B$ and $\mathcal{R}_2 \subseteq B \times C$ as $\mathcal{R}_1 \cdot \mathcal{R}_2 = \{(a,c) \in A \times C \mid \exists b \in B.\ (a,b) \in \mathcal{R}_1 \wedge (b,c) \in \mathcal{R}_2\}$. Finally, we write $\mathcal{R}^{\mathsf{c}} = (A \times B) \setminus \mathcal{R}$ for the *complement* of $\mathcal{R}$. Note that we have the following identities:

$$(\mathcal{R}_1 \cdot \mathcal{R}_2)^{-1} = {\mathcal{R}_2}^{-1} \cdot {\mathcal{R}_1}^{-1} \qquad (\mathcal{R}^{\mathsf{c}})^{-1} = \left(\mathcal{R}^{-1}\right)^{\mathsf{c}} \qquad (\mathcal{R}_1 \cap \mathcal{R}_2)^{-1} = \mathcal{R}_1^{-1} \cap \mathcal{R}_2^{-1}\,.$$

A linear order $\leq$ over a set A is a reflexive, transitive and antisymmetric relation $\leq \subseteq A \times A$ such that for all $a, b \in A$, we have $a \leq b$ or $b \leq a$. Let $(A, \leq)$ be a linearly ordered set. For $A' \subseteq A$, we also denote by $\leq$ the restriction of $\leq$ to A', so that $(A', \leq)$ is still a linearly ordered set. Moreover, for $a \in A$, we write $a < A'$ if for all $a' \in A'$, $a < a'$, and $A' < a$ if for all $a' \in A'$, $a' < a$.

An *interval* of $(A, \leq)$ is a set $I \subseteq A$ such that for all $a \leq b \leq c$ with $a, c \in I$, we have $b \in I$. For $a, b \in A$, we denote by $[a,b)$ the interval $\{c \in A \mid a \leq c < b\}$, and similarly for the intervals $[a,b]$, $(a,b]$, (a,b). We call a relation $\mathcal{R} \subseteq A \times B$ between two linearly ordered sets $(A, \leq_A)$ and $(B, \leq_B)$ *interval-preserving* if:

- For all intervals I of $(A, \leq_A)$, $\mathcal{R}(I)$ is an interval of $(\mathcal{R}(A), \leq_B)$.
- For all intervals J of $(B, \leq_B)$, $\mathcal{R}^{-1}(J)$ is an interval of $(\mathcal{R}^{-1}(B), \leq_A)$.

Figure 1 Definition of interval-preserving relations.

In other terms, for all $a_1 \mathcal{R} b_1$ and $a_2 \mathcal{R} b_2$ with $a_1, a_2 \in I$, for all $b_1 \leq_B b \leq_B b_2$, if there exists some $a \in A$ such that $a \mathcal{R} b$, then there exists one in I (cf. Figure 1). Note that we do not require that *all* elements between b_1 and b_2 are in $\mathcal{R}(I)$, but only those which are in the image of $\mathcal{R}$. The second condition is symmetric: for all $a_1 \mathcal{R} b_1$ and $a_2 \mathcal{R} b_2$ with $b_1, b_2 \in J$, for all $a_1 \leq_A a \leq_A a_2$, if there exists some $b \in B$ such that $a \mathcal{R} b$, then there exists one in J.

▶ **Example 1.** For any linear order $(A, \leq)$ and partial function $f : A \to A$, if f is increasing or decreasing then the relation $\{(a, f(a)) \mid a \in \mathrm{dom}(f)\}$ is interval-preserving.

As another example, consider a temporal structure $(A, \leq, \lambda)$ over a set of atomic propositions AP, where $\lambda : A \to 2^{\mathrm{AP}}$ indicates the set of propositions which are true at a given point. For $P, Q \in \mathrm{AP}$, we let $\mathsf{until}_{P,Q} = \{(a, b) \in A \times A \mid a < b \wedge Q \in \lambda(b) \wedge \forall a < c < b, P \in \lambda(c)\}$. Then $\mathsf{until}_{P,Q}$ is interval-preserving.

The following lemma states some simple closure properties of interval-preserving relations.

▶ **Lemma 2.** *Let $(A, \leq_A)$, $(B, \leq_B)$, $(C, \leq_C)$ be linearly ordered sets.*

1. *For all interval-preserving relation $\mathcal{R} \subseteq A \times B$, $\mathcal{R}^{-1}$ is interval-preserving.*
2. *For all interval-preserving relations $\mathcal{R}_1, \mathcal{R}_2 \subseteq A \times B$, $\mathcal{R}_1 \cap \mathcal{R}_2$ is interval-preserving.*
3. *For all interval-preserving relations $\mathcal{R}_1 \subseteq A \times B$ and $\mathcal{R}_2 \subseteq B \times C$, $\mathcal{R}_1 \cdot \mathcal{R}_2$ is interval-preserving.*

Proof. Part 1 follows from the fact that $(\mathcal{R}^{-1})^{-1} = \mathcal{R}$.

Let us prove 2. Since $(\mathcal{R}_1 \cap \mathcal{R}_2)^{-1} = \mathcal{R}_1^{-1} \cap \mathcal{R}_2^{-1}$, by symmetry, it suffices to prove that for all interval I of $(A, \leq)$, $(\mathcal{R}_1 \cap \mathcal{R}_2)(I)$ is an interval of $((\mathcal{R}_1 \cap \mathcal{R}_2)(A), \leq)$. Let $a_1, a_2 \in I$ and $b_1 \leq b \leq b_2$ such that $(a_1, b_1), (a_2, b_2) \in (\mathcal{R}_1 \cap \mathcal{R}_2)$ and $(a, b) \in (\mathcal{R}_1 \cap \mathcal{R}_2)$ for some $a \in A$. If $a \in I$, then we are done. Otherwise, suppose for instance that $a < a_1 \leq a_2$ (the other cases are similar). Since $\mathcal{R}_1$ is interval-preserving, there exists $a_1 \leq a' \leq a_2$ such that $a' \mathcal{R}_1 b$. Then, since $a < a_1 \leq a'$ and $\mathcal{R}_1^{-1}(b)$ is an interval of $(\mathcal{R}_1^{-1}(B), \leq_A)$, we obtain $a_1 \mathcal{R}_1 b$. Similarly, $a_1 \mathcal{R}_2 b$. Hence $a_1 (\mathcal{R}_1 \cap \mathcal{R}_2) b$.

Let us show that 2 implies 3. Again, by symmetry, it suffices to prove that for all interval I of $(A, \leq_A)$, $(\mathcal{R}_1 \cdot \mathcal{R}_2)(I)$ is an interval of $((\mathcal{R}_1 \cdot \mathcal{R}_2)(A), \leq_C)$. Let $\mathcal{R}_3 \subseteq B \times C$ denote the relation $\mathcal{R}_1(A) \times C$. It is an interval-preserving relation between $(B, \leq_B)$ and $(C, \leq_C)$. Moreover, we have $(\mathcal{R}_1 \cdot \mathcal{R}_2)(A) = (\mathcal{R}_2 \cap \mathcal{R}_3)(B)$. Now, let I be some interval of $(A, \leq_A)$, and $J = \{b \in B \mid \exists b_1, b_2 \in \mathcal{R}_1(I), b_1 \leq b \leq b_2\}$. Then J is an interval of $(B, \leq_B)$. Moreover, since $\mathcal{R}_1$ is interval-preserving, we have $\mathcal{R}_1(I) = J \cap \mathcal{R}_1(A)$, hence

$$(\mathcal{R}_1 \cdot \mathcal{R}_2)(I) = \mathcal{R}_2(\mathcal{R}_1(I)) = \mathcal{R}_2(J \cap \mathcal{R}_1(A)) = (\mathcal{R}_2 \cap \mathcal{R}_3)(J)\,.$$

Then, according to 2, $(\mathcal{R}_1 \cdot \mathcal{R}_2)(I)$ is an interval of $((\mathcal{R}_2 \cap \mathcal{R}_3)(B), \leq_C)$, i.e., an interval of $((\mathcal{R}_1 \cdot \mathcal{R}_2)(A), \leq_C)$. ◀

Models. Let $\mathcal{P} = \{P, Q, \ldots\}$ be an infinite set of monadic predicates, and $\Gamma = \{\alpha, \beta, \ldots\}$ be a finite or infinite set of binary relation symbols. Throughout the paper, $\mathcal{M}$ will denote a structure $\mathcal{M} = (A, \leq, (\alpha^{\mathcal{M}})_{\alpha \in \Gamma}, (P^{\mathcal{M}})_{P \in \mathcal{P}})$ where $\leq$ is a linear order over A, $\alpha^{\mathcal{M}} \subseteq A \times A$ is an interval-preserving relation for all $\alpha \in \Gamma$, and $P^{\mathcal{M}} \subseteq A$ for all $P \in \mathcal{P}$.

Monadic first-order logic. We assume an infinite supply of variables $\mathcal{X} = \{x, y, \ldots\}$. The set $\mathrm{FO}[\Gamma, \leq]$ of monadic first-order logic formulas over Γ is defined as follows:

$$\Phi ::= P(x) \mid x \leq y \mid x = y \mid \alpha(x, y) \mid \Phi \vee \Phi \mid \neg\Phi \mid \exists x.\Phi\,, \qquad \text{where } x, y \in \mathcal{X}, P \in \mathcal{P}, \alpha \in \Gamma\,.$$

We assume that all formulas are interpreted over structures $\mathcal{M}$ defined as above. Given an $\mathrm{FO}[\Gamma, \leq]$ formula Φ, we denote by $\mathsf{Free}(\Phi)$ its set of free variables. We define the satisfaction relation $\mathcal{M}, \nu \models \Phi$ as usual, where $\mathcal{M} = (A, \leq, (\alpha^{\mathcal{M}})_{\alpha\in\Gamma}, (P^{\mathcal{M}})_{P\in\mathcal{P}})$ and $\nu : \mathsf{Free}(\Phi) \to A$ is an interpretation of the free variables of Φ. We say that two formulas $\Phi, \Psi \in \mathrm{FO}[\Gamma, \leq]$ are *equivalent*, written $\Phi \equiv \Psi$, if for all $\mathcal{M} = (A, \leq, (\alpha^{\mathcal{M}})_{\alpha\in\Gamma}, (P^{\mathcal{M}})_{P\in\mathcal{P}})$ and $\nu : \mathsf{Free}(\Phi) \cup \mathsf{Free}(\Psi) \to A$, we have $\mathcal{M}, \nu|_{\mathsf{Free}(\Phi)} \models \Phi$ if and only if $\mathcal{M}, \nu|_{\mathsf{Free}(\Psi)} \models \Psi$.

For $k \in \mathbb{N}$, we denote by $\mathrm{FO}^k[\Gamma, \leq]$ the set of first-order formulas with at most k variables. Note that a same variable may be quantified over several times in the formula.

▶ **Example 3.** Let $p : \mathbb{R} \to \mathbb{R}$ be a polynomial function, and $m_1 < \cdots < m_n$ its local extrema (we suppose that $n \geq 1$). Fix $\Gamma = \{p\}$. For convenience, we write $p(x) = y$ instead of $p(x, y)$ in $\mathrm{FO}[\Gamma, \leq]$ formulas. We focus on models of the form $\mathcal{M} = (\mathbb{R}, \leq, p^{\mathcal{M}}, (P^{\mathcal{M}})_{P\in\mathcal{P}})$ where $\leq$ is the usual ordering of the reals, and $p^{\mathcal{M}} = \{(x, p(x)) \mid x \in \mathbb{R}\}$. Let us describe an $\mathrm{FO}^3[\Gamma, \leq]$ formula $m_i \leq x$ such that for all $\mathcal{M}$ and $r \in \mathbb{R}$, we have $\mathcal{M}, [x \mapsto r] \models m_i \leq x$ if and only if $m_i \leq r$. First, we write $p(x) \leq p(y)$ for the $\mathrm{FO}^3[\Gamma, \leq]$ formula

$$\exists z.\, p(x) = z \wedge \exists x.\, (p(y) = x \wedge z \leq x)\,.$$

We can then define formulas $\min(x) \in \mathrm{FO}^3[\Gamma, \leq]$ and $\max(x) \in \mathrm{FO}^3[\Gamma, \leq]$ which state that x is a local minimum (resp. maximum) of p, for instance:

$$\begin{aligned}\min(x) = {} & (\exists z.\, z < x \wedge \forall y.\, (z < y \leq x \implies p(x) \leq p(y))) \wedge \\ & (\exists z.\, x < z \wedge \forall y.\, (x \leq y < z \implies p(x) \leq p(y)))\,.\end{aligned}$$

The formula $m_i \leq x$ then states that there exist at least i local extrema before x, alternating existential quantifications over y and z to identify them; for instance, $m_3 \leq x$ is the formula

$$\exists y.\, y \leq x \wedge (\min(y) \vee \max(y)) \wedge \exists z.\, z < y \wedge (\min(z) \vee \max(z)) \wedge \exists y.\, y < z \wedge (\min(y) \vee \max(y))\,.$$

3 Star-free Propositional Dynamic Logic

Star-free Propositional Dynamic Logic. Propositional dynamic logic (PDL) [6] consists of two sorts of formulas: state formulas which are evaluated at single elements, and path formulas which are evaluated at pairs of elements and allow to navigate inside the model. Here we consider a star-free variant of PDL (with converse). The syntax of *star-free propositional dynamic logic* over Γ, written $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq]$, is given below:

$$\begin{aligned}&\varphi ::= P \mid \varphi \vee \varphi \mid \neg\varphi \mid \langle\pi\rangle\,\varphi && \text{(state formulas)}\\ &\pi ::= \alpha \mid \leq \mid \{\varphi\}? \mid \pi^{-1} \mid \pi \cdot \pi \mid \pi \cup \pi \mid \pi \cap \pi \mid \pi^{\mathsf{c}} && \text{(path formulas)}\end{aligned}$$

where $P \in \mathcal{P}$ and $\alpha \in \Gamma$.

Compared to classical PDL, star-free PDL uses the operators $(\cdot, \cup, \cap, \mathsf{c})$ of star-free expressions, instead of the rational operators $(\cdot, \cup, *)$.

Let $\mathcal{M} = (A, \leq, (\alpha^{\mathcal{M}})_{\alpha\in\Gamma}, (P^{\mathcal{M}})_{P\in\mathcal{P}})$. The semantics $[\![\varphi]\!]^{\mathcal{M}} \subseteq A$ or $[\![\pi]\!]^{\mathcal{M}} \subseteq A \times A$ of a state or path formula in $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq]$ is defined below. The state formula $\langle\pi\rangle\,\varphi$ is true at a point $a \in A$ in $\mathcal{M}$ (that is, $a \in [\![\langle\pi\rangle\,\varphi]\!]^{\mathcal{M}}$) if there exists some $b \in A$ such that (a, b) satisfies

π and φ is true at b. The path formula $\{\varphi\}?$ is stationary and tests if the state formula φ is true. The semantics of other formulas is straightforward:

$$\begin{aligned}
[\![P]\!]^{\mathcal{M}} &:= P^{\mathcal{M}} & [\![\varphi_1 \vee \varphi_2]\!]^{\mathcal{M}} &:= [\![\varphi_1]\!]^{\mathcal{M}} \cup [\![\varphi_2]\!]^{\mathcal{M}} \\
[\![\neg\varphi]\!]^{\mathcal{M}} &:= A \setminus [\![\varphi]\!]^{\mathcal{M}} & [\![\langle\pi\rangle\,\varphi]\!]^{\mathcal{M}} &:= \{a \in A \mid \exists b \in [\![\varphi]\!]^{\mathcal{M}},\ (a,b) \in [\![\pi]\!]^{\mathcal{M}}\} \\
[\![\alpha]\!]^{\mathcal{M}} &:= \alpha^{\mathcal{M}} & [\![\{\varphi\}?]\!]^{\mathcal{M}} &:= \{(a,a) \mid a \in [\![\varphi]\!]^{\mathcal{M}}\} \\
[\![\leq]\!]^{\mathcal{M}} &:= \leq & [\![\pi^{-1}]\!]^{\mathcal{M}} &:= ([\![\pi]\!]^{\mathcal{M}})^{-1} \\
[\![\pi_1 \cup \pi_2]\!]^{\mathcal{M}} &:= [\![\pi_1]\!]^{\mathcal{M}} \cup [\![\pi_2]\!]^{\mathcal{M}} & [\![\pi_1 \cap \pi_2]\!]^{\mathcal{M}} &:= [\![\pi_1]\!]^{\mathcal{M}} \cap [\![\pi_2]\!]^{\mathcal{M}} \\
[\![\pi^{\mathsf{c}}]\!]^{\mathcal{M}} &:= (A \times A) \setminus [\![\pi]\!]^{\mathcal{M}} & [\![\pi_1 \cdot \pi_2]\!]^{\mathcal{M}} &:= [\![\pi_1]\!]^{\mathcal{M}} \cdot [\![\pi_2]\!]^{\mathcal{M}}\,.
\end{aligned}$$

For simplicity, we often write $[\![\varphi]\!]$ or $[\![\pi]\!]$ instead of $[\![\varphi]\!]^{\mathcal{M}}$ and $[\![\pi]\!]^{\mathcal{M}}$. We also write $\mathcal{M}, a \models \varphi$ if $a \in [\![\varphi]\!]^{\mathcal{M}}$, and $\mathcal{M}, a, b \models \pi$ if $(a,b) \in [\![\pi]\!]^{\mathcal{M}}$.

We use the abbreviations $\mathit{true} := P \vee \neg P$, $\mathit{false} := \neg\mathit{true}$, $\geq := (\leq)^{-1}$, $< := \geq^{\mathsf{c}}$, $> := \leq^{\mathsf{c}}$ and $\langle\pi\rangle := \langle\pi\rangle\,\mathit{true}$. For all $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq]$ formulas π, we also define a state formula $\mathsf{loop}(\pi) := \langle\pi \cap \{\mathit{true}\}?\rangle$ which holds at a if and only if $(a,a) \in [\![\pi]\!]$.

▶ **Example 4.** Suppose that $\Gamma = \{+q \mid q \in \mathbb{Q}\}$, and that we consider only models over $\mathbb{R}$ and with $[\![+q]\!] = \{(r, r+q) \mid r \in \mathbb{R}\}$. Let $q, r \in \mathbb{Q}_{\geq 0}$ and $P, Q \in \mathcal{P}$. The formula $P\,\mathsf{U}_{(q,r)}\,Q$ of metric temporal logic, which holds at time $t \in \mathbb{R}$ if there exists $t+q < t' < t+r$ such that $t' \in [\![Q]\!]$ and for all $t < t'' < t'$, $t'' \in [\![P]\!]$, can be expressed in $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq]$ as follows:

$$P\,\mathsf{U}_{(q,r)}\,Q \equiv \big\langle (+q \cdot <) \cap (+r \cdot >) \cap (< \cdot \{\neg P\}? \cdot <)^{\mathsf{c}} \big\rangle\, Q\,.$$

An interval-preserving fragment of star-free PDL. We say that a path formula $\pi \in \mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq]$ is *interval-preserving* if for all $\mathcal{M}$, $[\![\pi]\!]^{\mathcal{M}}$ is interval-preserving. Notice that $\leq$ and $\{\varphi\}?$ (for all φ) are interval-preserving. By Lemma 2 (and assumption on $[\![\alpha]\!]$), all $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq]$ formulas constructed without the boolean operators $\cup$ and c are interval-preserving. However, the complement or the union of interval-preserving relations are not in general interval-preserving. We define below a fragment of $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq]$ where all path formulas are interval-preserving, and which will turn out to be as expressive as $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq]$ (and in fact, $\mathrm{FO}[\Gamma, \leq]$) when it comes to *state* formulas. To do so, we introduce several restrictions of π^{c} which are interval-preserving, and which suffice to characterize π^{c}.

Let us first look at the different reasons for which we may have $(a,b) \in [\![\pi^{\mathsf{c}}]\!]$, assuming that π is interval-preserving. To begin with, we focus on a. One first sufficient condition for having $b \notin [\![\pi]\!](a)$ is that $[\![\pi]\!](a) = \emptyset$. Now, suppose $[\![\pi]\!](a) \neq \emptyset$. If π is interval-preserving, there are only three possible cases in which $b \notin [\![\pi]\!](a)$: $b < [\![\pi]\!](a)$, or $[\![\pi]\!](a) < b$, or $[\![\pi^{-1}]\!](b) = \emptyset$. We define formulas $\mathsf{left}\ \pi$ and $\mathsf{right}\ \pi$ corresponding respectively to the first two cases. We let

$$\begin{aligned}
\mathsf{left}\ \pi = \{\langle\pi\rangle\}? \cdot (\pi \cdot \leq)^{\mathsf{c}}, \quad \text{i.e.} \qquad & (a,b) \in [\![\mathsf{left}\ \pi]\!] \quad \text{iff} \quad b < [\![\pi]\!](a) \neq \emptyset \\
\mathsf{right}\ \pi = \{\langle\pi\rangle\}? \cdot (\pi \cdot \geq)^{\mathsf{c}}, \quad \text{i.e.} \qquad & (a,b) \in [\![\mathsf{right}\ \pi]\!] \quad \text{iff} \quad b > [\![\pi]\!](a) \neq \emptyset\,.
\end{aligned}$$

Now, if we look at $[\![\pi^{-1}]\!](b)$ instead of $[\![\pi]\!](a)$, we can make the same observations, by symmetry: we have $(a,b) \in [\![\pi^{\mathsf{c}}]\!]$ if and only if $a \notin [\![\pi^{-1}]\!](b)$, and if π is interval-preserving, there are again only four possible cases: $[\![\pi^{-1}]\!](b) = \emptyset$, or $a < [\![\pi^{-1}]\!](b)$, or $a > [\![\pi^{-1}]\!](b)$, or $[\![\pi]\!](a) = \emptyset$.

Unfortunately, the formulas $\mathsf{left}\ \pi$ and $\mathsf{right}\ \pi$ are still not interval-preserving in general. However, if we take a more symmetric restriction of π^{c}, where we look at all the possible positions of b and a relatively to $[\![\pi]\!](a)$ and $[\![\pi^{-1}]\!](b)$, we obtain four cases, illustrated in Figure 2, which we will later show correspond to interval-preserving restrictions of π^{c}.

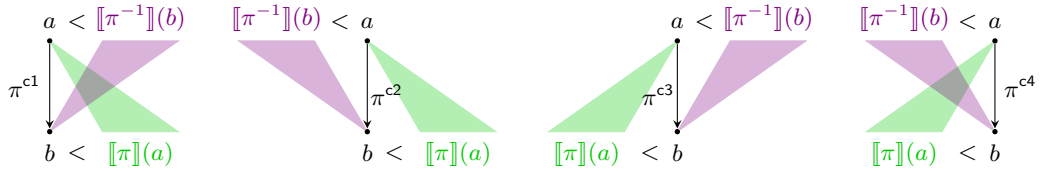


Figure 2 Definition of π^{c1}, π^{c2}, π^{c3} and π^{c4}, from left to right.

More precisely, let

$$\pi^{c1} := \mathsf{left}\ \pi \cap \left(\mathsf{left}\ \left(\pi^{-1}\right)\right)^{-1}, \text{i.e.} \qquad (a,b) \in [\![\pi^{c1}]\!] \quad \text{iff} \quad \begin{cases} b < [\![\pi]\!](a) \neq \emptyset & \text{and} \\ a < [\![\pi^{-1}]\!](b) \neq \emptyset \end{cases}$$

$$\pi^{c2} := \mathsf{left}\ \pi \cap \left(\mathsf{right}\ \left(\pi^{-1}\right)\right)^{-1}, \text{i.e.} \qquad (a,b) \in [\![\pi^{c2}]\!] \quad \text{iff} \quad \begin{cases} b < [\![\pi]\!](a) \neq \emptyset & \text{and} \\ a > [\![\pi^{-1}]\!](b) \neq \emptyset \end{cases}$$

$$\pi^{c3} := \mathsf{right}\ \pi \cap \left(\mathsf{left}\ \left(\pi^{-1}\right)\right)^{-1}, \text{i.e.} \qquad (a,b) \in [\![\pi^{c3}]\!] \quad \text{iff} \quad \begin{cases} b > [\![\pi]\!](a) \neq \emptyset & \text{and} \\ a < [\![\pi^{-1}]\!](b) \neq \emptyset \end{cases}$$

$$\pi^{c4} := \mathsf{right}\ \pi \cap \left(\mathsf{right}\ \left(\pi^{-1}\right)\right)^{-1}, \text{i.e.} \qquad (a,b) \in [\![\pi^{c4}]\!] \quad \text{iff} \quad \begin{cases} b > [\![\pi]\!](a) \neq \emptyset & \text{and} \\ a > [\![\pi^{-1}]\!](b) \neq \emptyset\,. \end{cases}$$

Notice that $\pi^{c3} \equiv ((\pi^{-1})^{c2})^{-1}$.

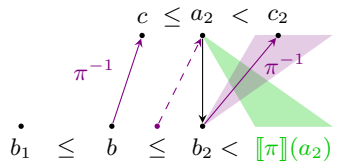
Let $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq, \cap, \mathsf{c1}, \mathsf{c2}, \mathsf{c3}, \mathsf{c4}]$ be the following restriction of $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq]$:

$$\varphi ::= P \mid \varphi \vee \varphi \mid \neg\varphi \mid \langle\pi\rangle\,\varphi$$
$$\pi ::= \alpha \mid \leq \mid \{\varphi\}? \mid \pi^{-1} \mid \pi \cdot \pi \mid \pi \cap \pi \mid \pi^{c1} \mid \pi^{c2} \mid \pi^{c3} \mid \pi^{c4}\,.$$

▶ **Lemma 5.** *All* $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq, \cap, \mathsf{c1}, \mathsf{c2}, \mathsf{c3}, \mathsf{c4}]$ *formulas are interval-preserving.*

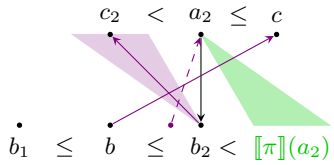
Proof. We proceed by induction on the formula. By assumption, α is interval-preserving for all $\alpha \in \Gamma$. Moreover, $\leq$ and $\{\varphi\}?$ are interval-preserving. For π^{-1}, $\pi_1 \cdot \pi_2$ and $\pi_1 \cap \pi_2$, we apply Lemma 2.

Suppose that π is interval-preserving. Let us show that π^{c1} is interval-preserving. Notice that $(\pi^{c1})^{-1} \equiv (\pi^{-1})^{c1}$. So we only need to show that for all intervals I, for all $b_1, b_2 \in [\![\pi^{c1}]\!](I)$ and $b_1 \leq b \leq b_2$ such that $[\![(\pi^{c1})^{-1}]\!](b) \neq \emptyset$, there exists $a \in I$ such that $(a,b) \in [\![\pi^{c1}]\!]$. Let $a_2 \in I$ such that $(a_2, b_2) \in [\![\pi^{c1}]\!]$. Let us show that we can in fact take $a = a_2$. The proof is illustrated in the picture below.



First, we have $b \leq b_2 < [\![\pi]\!](a_2) \neq \emptyset$. Moreover, $[\![\pi^{-1}]\!](b) \neq \emptyset$ (since $[\![(\pi^{c1})^{-1}]\!](b) \neq \emptyset$). Now, suppose towards a contradiction that $a_2 \not< [\![\pi^{-1}]\!](b)$. Let $c \in [\![\pi^{-1}]\!](b)$ such that $c \leq a_2$. Since $(a_2, b_2) \in [\![\pi^{c1}]\!]$, there exists $c_2 > a_2$ such that $(b_2, c_2) \in [\![\pi^{-1}]\!]$. We then have $c \leq a_2 < c_2$ and $[\![\pi]\!](a_2) \neq \emptyset$. Since π is interval-preserving, we obtain $a_2 \in [\![\pi^{-1}]\!]([b, b_2])$, a contradiction with the fact that $b_2 < [\![\pi]\!](a_2)$. Thus, $(a_2, b) \in [\![\pi^{c1}]\!]$.

Let us show that π^{c2} is also interval-preserving. Similarly to the previous case, we show that for all $(a_2, b_2) \in [\![\pi^{\mathsf{c2}}]\!]$ and $b \leq b_2$ such that $[\![(\pi^{\mathsf{c2}})^{-1}]\!](b) \neq \emptyset$, we have $(a_2, b) \in [\![\pi^{\mathsf{c2}}]\!]$.



First, $b \leq b_2 < [\![\pi]\!](a_2) \neq \emptyset$, and $[\![\pi^{-1}]\!](b) \neq \emptyset$. Suppose towards a contradiction that $[\![\pi^{-1}]\!](b) \not< a_2$. Let $c \in [\![\pi^{-1}]\!](b)$ such that $a_2 \leq c$, and $c_2 \in [\![\pi^{-1}]\!](b_2)$. We have $c_2 < a_2 \leq c$, and $[\![\pi]\!](a_2) \neq \emptyset$. Since π is interval-preserving, we obtain $a_2 \in [\![\pi^{-1}]\!]([b, b_2])$, a contradiction with the fact that $b_2 < [\![\pi]\!](a_2)$. Symmetrically, let J be an interval, $a_1, a_2 \in [\![(\pi^{\mathsf{c2}})^{-1}]\!](J)$, and $a_1 \leq a \leq a_2$ such that $[\![\pi^{\mathsf{c2}}]\!](a) \neq \emptyset$. Then for any $b_1 \in J$ such that $(a_1, b_1) \in [\![\pi^{\mathsf{c2}}]\!]$, we also have $(a, b_1) \in [\![\pi^{\mathsf{c2}}]\!]$, hence $a \in [\![(\pi^{\mathsf{c2}})^{-1}]\!](J)$.

Since $\pi^{\mathsf{c3}} \equiv ((\pi^{-1})^{\mathsf{c2}})^{-1}$, this also implies that π^{c3} is interval-preserving.

Finally, the case of π^{c4} is symmetric to the case of π^{c1}: for all $(a_1, b_1) \in [\![\pi^{\mathsf{c4}}]\!]$ and $b_1 \leq b$ such that $[\![(\pi^{\mathsf{c4}})^{-1}]\!](b) \neq \emptyset$, we have $(a_1, b) \in [\![\pi^{\mathsf{c4}}]\!]$. ◀

4 Star-free PDL is expressively equivalent to FO

Let φ be a state formula in $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq]$, and $\Phi(x)$ an $\mathrm{FO}[\Gamma, \leq]$ formula with a single free variable x. We say that φ and Φ are equivalent, written $\varphi \equiv \Phi(x)$, if for all $\mathcal{M}$ and elements a in $\mathcal{M}$, we have $\mathcal{M}, a \models \varphi$ if and only if $\mathcal{M}, [x \mapsto a] \models \Phi(x)$. Similarly, for a path formula $\pi \in \mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq]$ and an $\mathrm{FO}[\Gamma, \leq]$ formula $\Phi(x, y)$ with exactly two free variables x and y, we write $\pi \equiv \Phi(x, y)$ if for all $\mathcal{M}$ and elements a, b in $\mathcal{M}$, we have $\mathcal{M}, a, b \models \pi$ if and only if $\mathcal{M}, [x \mapsto a, y \mapsto b] \models \Phi(x, y)$.

From $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq]$ to $\mathrm{FO}^3[\Gamma, \leq]$. An easy induction shows that any formula in $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq]$ can be translated into an $\mathrm{FO}[\Gamma, \leq]$ formula which uses at most three distinct variables:

▶ **Lemma 6.** *For every state formula $\varphi \in \mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq]$, there exists a formula $\widetilde{\varphi}(x) \in \mathrm{FO}^3[\Gamma, \leq]$ such that $\varphi \equiv \widetilde{\varphi}(x)$. For every path formula $\pi \in \mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq]$, there exists a formula $\widetilde{\pi}(x, y) \in \mathrm{FO}^3[\Gamma, \leq]$ such that $\pi \equiv \widetilde{\pi}(x, y)$.*

For the other direction, we will see that the fragment $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq, \mathsf{loop}, \mathsf{c1}, \mathsf{c2}, \mathsf{c3}, \mathsf{c4}]$ of $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq]$ defined below is sufficient:

$$\varphi ::= P \mid \varphi \vee \varphi \mid \neg\varphi \mid \langle\pi\rangle\, \varphi \mid \mathsf{loop}(\pi)$$
$$\pi ::= \alpha \mid \leq \mid \{\varphi\}? \mid \pi^{-1} \mid \pi \cdot \pi \mid \pi^{\mathsf{c1}} \mid \pi^{\mathsf{c2}} \mid \pi^{\mathsf{c3}} \mid \pi^{\mathsf{c4}}\,.$$

This fragment is a restriction of $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq, \cap, \mathsf{c1}, \mathsf{c2}, \mathsf{c3}, \mathsf{c4}]$, where the intersection is only used for $\mathsf{loop}(\pi)$ formulas.

From $\mathrm{FO}[\Gamma, \leq]$ to $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq, \mathsf{loop}, \mathsf{c1}, \mathsf{c2}, \mathsf{c3}, \mathsf{c4}]$. The main result of the paper is an effective translation of $\mathrm{FO}[\Gamma, \leq]$ formulas into positive boolean combinations of formulas in $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq, \mathsf{loop}, \mathsf{c1}, \mathsf{c2}, \mathsf{c3}, \mathsf{c4}]$:

▶ **Theorem 7.** *Every formula $\Phi \in \mathrm{FO}[\Gamma, \leq]$ with at least one free variable is equivalent to a positive boolean combination of formulas of the form $\widetilde{\pi}(x, y)$, where $x, y \in \mathsf{Free}(\Phi)$ and $\pi \in \mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq, \mathsf{loop}, \mathsf{c1}, \mathsf{c2}, \mathsf{c3}, \mathsf{c4}]$.*

Note that the equivalent formula may also contain subformulas of the form $\widetilde{\pi}(x,x)$.

Before proving Theorem 7, we state some of its consequences.

▶ **Corollary 8.** *Every formula* $\Phi \in \mathrm{FO}[\Gamma,\leq]$ *with a single free variable is equivalent to some* $\mathrm{PDL}_{\mathsf{sf}}[\Gamma,\leq,\mathsf{loop},\mathsf{c1},\mathsf{c2},\mathsf{c3},\mathsf{c4}]$ *state formula. Every formula* $\Phi \in \mathrm{FO}[\Gamma,\leq]$ *with two free variables is equivalent to some* $\mathrm{PDL}_{\mathsf{sf}}[\Gamma,\leq]$ *path formula.*

Proof. If Φ has a single free variable x, it is equivalent to a positive boolean combination of formulas of the form $\widetilde{\pi}(x,x)$, which are themselves equivalent to the formulas $\mathsf{loop}(\pi)$. The combination of these $\mathsf{loop}(\pi)$ formulas is then a state formula of $\mathrm{PDL}_{\mathsf{sf}}[\Gamma,\leq,\mathsf{loop},\mathsf{c1},\mathsf{c2},\mathsf{c3},\mathsf{c4}]$.

If Φ has two free variables x and y, we obtain an equivalent positive boolean combination of formulas of the form $\widetilde{\pi}(x,y)$, $\widetilde{\pi}(y,x)$, $\widetilde{\pi}(x,x)$, or $\widetilde{\pi}(y,y)$. We can replace any subformula $\widetilde{\pi}(y,x)$ with $\widetilde{\pi^{-1}}(x,y)$, and any subformula $\widetilde{\pi}(x,x)$ with $\widetilde{\pi_1}(x,y)\vee\widetilde{\pi_2}(x,y)$, where $\pi_1 = (\{\mathsf{loop}(\pi)\}?\cdot\leq)$ and $\pi_2 = (\{\mathsf{loop}(\pi)\}?\cdot\geq)$, and similarly for formulas $\widetilde{\pi}(y,y)$. We obtain an equivalent positive boolean combination of formulas of the form $\widetilde{\pi}(x,y)$. Since $\mathrm{PDL}_{\mathsf{sf}}[\Gamma,\leq]$ allows union and intersection of path formulas, this is equivalent to a $\mathrm{PDL}_{\mathsf{sf}}[\Gamma,\leq]$ formula. ◀

Another consequence is that $\mathrm{FO}[\Gamma,\leq]$ over linear orders with interval-preserving relations has the three-variable property. More precisely:

▶ **Theorem 9.** *Any* $\mathrm{FO}[\Gamma,\leq]$ *formula is equivalent to a boolean combination of formulas in* $\mathrm{FO}^3[\Gamma,\leq]$.

This also allows us to answer an open question from [1], namely, whether structures over the real numbers with polynomial functions have the 3-variable property. Suppose that Γ is a set of polynomials $p:\mathbb{R}\to\mathbb{R}$. Let $\mathcal{M}_\Gamma = (\mathbb{R},\leq,(p^{\mathcal{M}_\Gamma})_{p\in\Gamma})$, where $\leq$ is the usual ordering of the real numbers, and $p^{\mathcal{M}_\Gamma} = \{(x,p(x)) \mid x\in\mathbb{R}\}$ for all $p\in\Gamma$. Given an interpretation $h:\mathcal{P}\to 2^{\mathbb{R}}$ of the monadic predicates, we denote by $(\mathcal{M}_\Gamma,h)$ the structure $(\mathbb{R},\leq,(p^{\mathcal{M}_\Gamma})_{p\in\Gamma},(h(P))_{P\in\mathcal{P}})$. We say that two formulas $\Phi,\Psi\in\mathrm{FO}[\Gamma,\leq]$ are equivalent over $\mathcal{M}_\Gamma$, written $\Phi\equiv_{\mathcal{M}_\Gamma}\Psi$, if for all $h:\mathcal{P}\to 2^{\mathbb{R}}$ and $\nu:\mathsf{Free}(\Phi)\cup\mathsf{Free}(\Psi)\to\mathbb{R}$, we have $(\mathcal{M}_\Gamma,h),\nu|_{\mathsf{Free}(\Phi)}\models\Phi$ if and only if $(\mathcal{M}_\Gamma,h),\nu|_{\mathsf{Free}(\Psi)}\models\Psi$.

▶ **Theorem 10.** *For all* $\Phi\in\mathrm{FO}[\Gamma,\leq]$, *there exists a boolean combination* Ψ *of formulas in* $\mathrm{FO}^3[\Gamma,\leq]$ *such that* $\Phi\equiv_{\mathcal{M}_\Gamma}\Psi$.

Proof. Let $p\in\Gamma$, and $m_1<\cdots<m_n$ its local extrema. We denote by $p_{(-\infty,m_1)}$, $p_{[m_1,m_2)},\ldots,p_{[m_n,+\infty)}$ the (monotone) restrictions of p to intervals delimitated by its local extrema, and Δ_p the set of these partial functions. Let $\Delta=\bigcup_{p\in\Gamma}\Delta_p$. As above, we let $\mathcal{M}_\Delta = (\mathbb{R},\leq,({p_I}^{\mathcal{M}_\Delta})_{p_I\in\Delta})$, where $\leq$ is the usual ordering of the real numbers, and ${p_I}^{\mathcal{M}_\Delta} = \{(x,p(x)) \mid x\in I\}$. Note that ${p_I}^{\mathcal{M}_\Delta}$ is interval-preserving (cf. Example 1). We say that two formulas $\Phi\in\mathrm{FO}[\Gamma,\leq]$ and $\Psi\in\mathrm{FO}[\Delta,\leq]$ are equivalent, written $\Phi\equiv\Psi$, when for all $h:\mathcal{P}\to 2^{\mathbb{R}}$ and $\nu:\mathsf{Free}(\Phi)\cup\mathsf{Free}(\Psi)\to\mathbb{R}$, we have $(\mathcal{M}_\Gamma,h),\nu|_{\mathsf{Free}(\Phi)}\models\Phi$ if and only if $(\mathcal{M}_\Delta,h),\nu|_{\mathsf{Free}(\Psi)}\models\Psi$.

Let $\Phi\in\mathrm{FO}[\Gamma,\leq]$. The formula $\Psi\in\mathrm{FO}[\Delta,\leq]$ obtained by replacing each atomic formula $p(x,y)$ by $\bigvee_{p_I\in\Delta_p}p_I(x,y)$ is equivalent to Φ. Applying Theorem 9 to Ψ, we obtain another formula $\Psi'\in\mathrm{FO}[\Delta,\leq]$ such that $\Psi'\equiv\Psi$ and Ψ' is a boolean combination of formulas in $\mathrm{FO}^3[\Delta,\leq]$.

Following Example 3, one can construct for each $p_I\in\Delta$ a formula "$x\in I$" of $\mathrm{FO}^3[\Gamma,\leq]$ such that $(\mathcal{M}_\Gamma,h),\nu\models x\in I$ if and only if $\nu(x)\in I$. Consider now the formula $\Phi'\in\mathrm{FO}[\Gamma,\leq]$ obtained by replacing each atomic formula $p_I(x,y)$ in Ψ' by $x\in I\wedge p(x,y)$. Then $\Phi'\equiv\Psi'$, hence $\Phi\equiv_{\mathcal{M}_\Gamma}\Phi'$. Moreover, Φ' is a boolean combination of formulas in $\mathrm{FO}^3[\Gamma,\leq]$. ◀

The remainder of the section is devoted to the proof of Theorem 7.

Eliminating negations. The fact that all $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq, \mathsf{loop}, \mathsf{c1}, \mathsf{c2}, \mathsf{c3}, \mathsf{c4}]$ path formulas are interval-preserving gives us a simple characterization of the complement of a path formula: we show below that an element b is in $[\![\pi^{\mathsf{c}}]\!](a)$ if it is to the left or to the right of all elements of $[\![\pi]\!](a)$, or if it does not satisfy $\langle \pi^{-1} \rangle$. We can then show that the complement of a path formula in $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq, \mathsf{loop}, \mathsf{c1}, \mathsf{c2}, \mathsf{c3}, \mathsf{c4}]$ is equivalent to a union of path formulas in $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq, \mathsf{loop}, \mathsf{c1}, \mathsf{c2}, \mathsf{c3}, \mathsf{c4}]$. This will allow us to deal with negation in the translation from $\mathrm{FO}[\Gamma, \leq]$ to $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq, \mathsf{loop}, \mathsf{c1}, \mathsf{c2}, \mathsf{c3}, \mathsf{c4}]$.

▶ **Lemma 11.** *For all path formulas $\pi \in \mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq, \mathsf{loop}, \mathsf{c1}, \mathsf{c2}, \mathsf{c3}, \mathsf{c4}]$, π^{c} is equivalent to a union of $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq, \mathsf{loop}, \mathsf{c1}, \mathsf{c2}, \mathsf{c3}, \mathsf{c4}]$ formulas.*

Proof. We show that

$$\begin{aligned}\pi^{\mathsf{c}} \equiv\ & (\{\neg \langle \pi \rangle\}? \cdot \leq) \cup (\{\neg \langle \pi \rangle\}? \cdot \geq) \cup \\ & (\leq \cdot \{\neg \langle \pi^{-1} \rangle\}?) \cup (\geq \cdot \{\neg \langle \pi^{-1} \rangle\}?) \cup \\ & (\pi^{\mathsf{c1}}) \cup (\pi^{\mathsf{c2}}) \cup (\pi^{\mathsf{c3}}) \cup (\pi^{\mathsf{c4}})\,.\end{aligned}$$

We denote by π' the right-hand-side formula. First, for all a, b such that $[\![\pi]\!](a) = \emptyset$ or $[\![\pi^{-1}]\!](b) = \emptyset$, we have $(a, b) \in [\![\pi^{\mathsf{c}}]\!]$ and $(a, b) \in [\![\pi']\!]$. Now, suppose that $[\![\pi]\!](a) \neq \emptyset$ and $[\![\pi^{-1}]\!](b) \neq \emptyset$. We have $(a, b) \in [\![\pi']\!]$ if and only if $(a, b) \in [\![\pi^{\mathsf{c1}} \cup \pi^{\mathsf{c2}} \cup \pi^{\mathsf{c3}} \cup \pi^{\mathsf{c4}}]\!]$. Clearly, if $(a, b) \in [\![\pi^{\mathsf{c1}} \cup \pi^{\mathsf{c2}} \cup \pi^{\mathsf{c3}} \cup \pi^{\mathsf{c4}}]\!]$, then $(a, b) \in [\![\pi^{\mathsf{c}}]\!]$. Conversely, let us show that if $(a, b) \notin [\![\pi^{\mathsf{c1}} \cup \pi^{\mathsf{c2}} \cup \pi^{\mathsf{c3}} \cup \pi^{\mathsf{c4}}]\!]$ then $(a, b) \in [\![\pi]\!]$. In that case, we have either $a_1 \leq a \leq a_2$ for some $a_1, a_2 \in [\![\pi^{-1}]\!](b)$, or $b_1 \leq b \leq b_2$ for some $b_1, b_2 \in [\![\pi]\!](a)$. Since π is interval-preserving, we obtain $(a, b) \in [\![\pi]\!]$. ◀

Existential quantification. The elimination of existential quantifiers relies on the simple lemma below:

▶ **Lemma 12.** *Let $(A, \leq)$ be a linearly ordered set, and $I_1, \ldots, I_n$ intervals of $(A, \leq)$. Then $\bigcap_{1 \leq i \leq n} I_i \neq \emptyset$ if and only if for all $1 \leq i, j \leq n$, $I_i \cap I_j \neq \emptyset$.*

Proof. We show that there exists k and ℓ such that $\bigcap_{1 \leq i \leq n} I_i = I_k \cap I_\ell$, which implies the result. We define relations $\sqsubseteq_{\text{left}}$ and $\sqsubseteq_{\text{right}}$ over $\{I_1, \ldots, I_n\}$ which, intuitively, compare respectively the left and right bounds of the intervals:

$$\begin{aligned} I \sqsubseteq_{\text{left}} J \quad & \text{if} \quad \forall a \in J, \exists b \in I, b \leq a \\ I \sqsubseteq_{\text{right}} J \quad & \text{if} \quad \forall a \in I, \exists b \in J, a \leq b\,. \end{aligned}$$

It is easy to check that $\sqsubseteq_{\text{left}}$ and $\sqsubseteq_{\text{right}}$ are transitive, an that for all I and J, we have $I \sqsubseteq_{\text{left}} J$ or $J \sqsubseteq_{\text{left}} I$ (or both), and similarly for $\sqsubseteq_{\text{right}}$. Thus, there exists k such that $I_i \sqsubseteq_{\text{left}} I_k$ for all i, and ℓ such that $I_\ell \sqsubseteq_{\text{right}} I_i$ for all i. Then for all $a \in I_k \cap I_\ell$, for all i, there exists $b, b' \in I_i$ such that $b \leq a \leq b'$. Since I_i is an interval, we obtain $a \in I_i$. Hence $I_k \cap I_\ell = \bigcap_{1 \leq i \leq n} I_i$. ◀

The next lemma follows from an application of Lemma 12 to intervals of the form $[\![\pi_i]\!](a_i)$.

▶ **Lemma 13.** *Let $n \geq 1$. For all path formulas $\pi_1, \ldots, \pi_n$ and all state formulas φ in $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq, \mathsf{loop}, \mathsf{c1}, \mathsf{c2}, \mathsf{c3}, \mathsf{c4}]$, the $\mathrm{FO}[\Gamma, \leq]$ formula*

$$\Phi(x_1, \ldots, x_n) = \exists x. \left(\widetilde{\varphi}(x) \wedge \bigwedge_{1 \leq i \leq n} \widetilde{\pi_i}(x_i, x) \right) \qquad (x_i \neq x \text{ for all } i)$$

is equivalent to a positive boolean combination of formulas of the form $\widetilde{\pi}(x_j, x_k)$, with $1 \leq j, k \leq n$ and $\pi \in \mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq, \mathsf{loop}, \mathsf{c1}, \mathsf{c2}, \mathsf{c3}, \mathsf{c4}]$.

Proof. Let $\psi = \varphi \wedge \bigwedge_{1\leq i\leq n} \langle \pi_i^{-1} \rangle$, and

$$\Psi(x_1,\dots,x_n) = \bigwedge_{1\leq i,j\leq n} (\pi_i \cdot \widetilde{\{\psi\}?} \cdot \pi_j^{-1})(x_i,x_j)\,.$$

Let us show that $\Phi(x_1,\dots,x_n) \equiv \Psi(x_1,\dots,x_n)$. Let $\mathcal{M} = (A, \leq, (\alpha^M)_{\alpha\in\Gamma}, (P^M)_{P\in\mathcal{P}})$, and $\nu : \{x_1,\dots,x_n\} \to A$. For all $1 \leq i \leq n$, let $I_i = [\![\pi_i]\!](\nu(x_i)) \cap [\![\psi]\!]$. Let us show that I_i is an interval of $([\![\psi]\!], \leq)$. First, since π_i is interval-preserving, $[\![\pi_i]\!](\nu(x_i))$ is an interval of $([\![\langle \pi_i^{-1} \rangle]\!], \leq)$. Thus, I_i is an interval of $([\![\langle \pi_i^{-1} \rangle]\!] \cap [\![\psi]\!], \leq)$. But since $[\![\langle \pi_i^{-1} \rangle]\!] \subseteq [\![\psi]\!]$, this is simply $([\![\psi]\!], \leq)$. Besides, it is easy to verify that

$$\mathcal{M}, \nu \models \Phi(x_1,\dots,x_n) \iff \bigcap_{1\leq i\leq n} I_i \neq \emptyset\,.$$

Applying Lemma 12, we obtain

$$\begin{aligned}
\mathcal{M}, \nu \models \Phi(x_1,\dots,x_n) &\iff \text{for all } 1 \leq i,j \leq n, I_i \cap I_j \neq \emptyset \\
&\iff \text{for all } 1 \leq i,j \leq n, (\nu(x_i),\nu(x_j)) \in [\![\pi_i \cdot \{\psi\}? \cdot \pi_j^{-1}]\!] \\
&\iff \mathcal{M}, \nu \models \Psi(x_1,\dots,x_n)\,.
\end{aligned}$$

◀

Translation from $\mathsf{FO}[\Gamma, \leq]$ to $\mathsf{PDL}_{\mathsf{sf}}[\Gamma, \leq, \mathsf{loop}, \mathsf{c1}, \mathsf{c2}, \mathsf{c3}, \mathsf{c4}]$. We are now ready to give the proof of Theorem 7.

Proof of Theorem 7. We assume that Φ is in prenex normal form, and prove the result by induction. The translation of atomic formulas $x \leq y$ or $\alpha(x,y)$ is straightforward; moreover, $P(x) \equiv \widetilde{\{P\}?}(x,x)$, and $(x = y) \equiv \widetilde{\{true\}?}(x,y)$. Using Lemma 11 to eliminate negations, we obtain the result for all quantifier-free formulas.

The case $\Phi = \forall x.\Psi \equiv \neg\exists x.\neg\Psi$ reduces to the case of existential quantification, applying again Lemma 11 to eliminate negations.

We are left with the case $\Phi = \exists x.\Psi$. If x is not free in Ψ, then $\Phi \equiv \Psi$ (since Ψ has at least one free variable) and we are done by induction. Otherwise, assume that $\mathsf{Free}(\Psi) = \{x_1,\dots,x_n\}$ with $n > 1$ and $x = x_n$. By induction, Ψ is equivalent to a positive boolean combination of formulas of the form $\widetilde{\pi}(x_i,x_j)$ with $\pi \in \mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq, \mathsf{loop}, \mathsf{c1}, \mathsf{c2}, \mathsf{c3}, \mathsf{c4}]$. We replace $\widetilde{\pi}(x_i,x_j)$ with $\widetilde{\pi^{-1}}(x_j,x_i)$ whenever $j < i$, and bring the resulting formula into disjunctive normal form. Each conjunct is then of the form $\Upsilon = \Upsilon_1 \wedge \Upsilon_2 \wedge \Upsilon_3$, where Υ_1 uses only variables from $\{x_1,\dots,x_{n-1}\}$, $\Upsilon_2 = \bigwedge_i \widetilde{\pi}_i(y_i,x)$ with $y_i = x_j$ for some $1 \leq j < n$, and $\Upsilon_3 = \bigwedge_j \widetilde{\pi}_j(x,x)$. Note that $\Upsilon_3 \equiv \widetilde{\varphi}(x)$, where $\varphi = \bigwedge_j \mathsf{loop}(\pi_j)$. Then $\exists x.\Psi$ is equivalent to a finite disjunction of formulas

$$\exists x.\Upsilon \;\equiv\; \Upsilon_1 \wedge \exists x.\,(\Upsilon_2 \wedge \widetilde{\varphi}(x))$$

with Υ_1 and Υ_2 as above. If Υ_2 is empty, then we replace $\exists x.\widetilde{\varphi}(x)$ with the formula

$$(\leq \cdot \{\varphi\}? \cdot \geq)(x_1,x_1) \vee (\geq \cdot \{\varphi\}? \cdot \leq)(x_1,x_1)\,.$$

Otherwise, we apply Lemma 13 to $\exists x.\,(\Upsilon_2 \wedge \widetilde{\varphi}(x))$. In all cases, we obtain an equivalent formula which is a positive boolean combination of formulas $\widetilde{\pi}(x_i,x_j)$ with $1 \leq i,j < n$ and $\pi \in \mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq, \mathsf{loop}, \mathsf{c1}, \mathsf{c2}, \mathsf{c3}, \mathsf{c4}]$. ◀

▶ Remark 14. Without the assumption that all atomic binary relations are interval-preserving, $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq]$ is still equivalent to $\mathrm{FO}^3[\Gamma, \leq]$. Indeed, in the proof of Theorem 7, the assumption that all atomic binary relations are interval-preserving is only used in Lemmas 11 and 13. But if Φ uses only three variables x, y and z, this assumption is not needed in the proof of Lemma 13. Indeed, we then have $\Phi(y,z) \equiv (\pi \cdot \widetilde{\{\varphi\}?} \cdot \pi'^{-1})(y,z)$, where π is the intersection of all π_i such that $x_i = y$, and π' is the intersection of all π_i such that $x_i = z$. Moreover, Lemma 11 is no longer needed if we translate an $\mathrm{FO}^3[\Gamma, \leq]$ formula into a positive boolean combination of $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq]$ formulas, since $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq]$ allows to take the complement of a path formula. Note that the equivalence with $\mathrm{FO}^3[\Gamma, \leq]$ is already proven in [28] (for the calculus of relations).

5 Conclusion

We proved that every $\mathrm{FO}[\Gamma, \leq]$ formula over linear orders with interval-preserving binary relations can be translated into an equivalent positive boolean combination of path formulas in $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq, \mathsf{loop}, \mathsf{c1}, \mathsf{c2}, \mathsf{c3}, \mathsf{c4}]$. In particular, any $\mathrm{FO}[\Gamma, \leq]$ formula is equivalent to a boolean combination of formulas in $\mathrm{FO}^3[\Gamma, \leq]$, which shows that $\mathrm{FO}[\Gamma, \leq]$ has the three-variable property. This generalizes several known results.

It would be interesting to see if the equivalence between $\mathrm{FO}[\Gamma, \leq]$ and $\mathrm{PDL}_{\mathsf{sf}}[\Gamma, \leq]$ can be used as an intermediate step to prove that a temporal logic is expressively complete. It is not the case in general, since [13] provides an example of a class of structures which fits our assumptions but does not admit any expressively complete temporal logic. However, the equivalence could still be useful in more restricted settings.

References

1 Timos Antonopoulos, Paul Hunter, Shahab Raza, and James Worrell. Three Variables Suffice for Real-Time Logic. In *FoSSaCS 2015*, volume 9034 of *LNCS*, pages 361–374. Springer, 2015.

2 Benedikt Bollig, Marie Fortin, and Paul Gastin. It Is Easy to Be Wise After the Event: Communicating Finite-State Machines Capture First-Order Logic with "Happened Before". In *CONCUR 2018*, volume 118 of *LIPIcs*, pages 7:1–7:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

3 Ryszard Danecki. Nondeterministic Propositional Dynamic Logic with intersection is decidable. In *Computation Theory*, pages 34–53. Springer Berlin Heidelberg, 1985.

4 Anuj Dawar. How Many First-order Variables are Needed on Finite Ordered Structures? In *We Will Show Them! (1)*, pages 489–520. College Publications, 2005.

5 G. De Giacomo and M. Lenzerini. Boosting the Correspondence between Description Logics and Propositional Dynamic Logics. In *Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 1.*, pages 205–212. AAAI Press / The MIT Press, 1994.

6 M. J. Fischer and R. E. Ladner. Propositional Dynamic Logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979.

7 D. M. Gabbay. Expressive Functional Completeness in Tense Logic. In Uwe Mönnich, editor, *Aspects of Philosophical Logic: Some Logical Forays into Central Notions of Linguistics and Philosophy*, pages 91–117. Springer Netherlands, Dordrecht, 1981.

8 D. M. Gabbay, I. M. Hodkinson, and M. A. Reynolds. *Temporal expressive completeness in the presence of gaps*, volume Volume 2 of *Lecture Notes in Logic*, pages 89–121. Springer-Verlag, 1993.

9 S. Göller, M. Lohrey, and C. Lutz. PDL with intersection and converse: satisfiability and infinite-state model checking. *Journal of Symbolic Logic*, 74(1):279–314, 2009.

10 Martin Grohe. Finite variable logics in descriptive complexity theory. *Bulletin of Symbolic Logic*, 4(4):345–398, 1998.

11 J. Y. Halpern and Y. Moses. A Guide to Completeness and Complexity for Modal Logics of Knowledge and Belief. *Artif. Intell.*, 54(2):319–379, 1992.

12 L. Henkin. *Logical Systems Containing Only a Finite Number of Symbols*. Séminaire de Mathématiques Supérieures: Publications. Presses de l'Université de Montréal, 1967.

13 Yoram Hirshfeld and Alexander Rabinovich. Expressiveness of Metric modalities for continuous time. *Logical Methods in Computer Science*, 3(1), 2007.

14 Ian M. Hodkinson. Finite H-dimension does not imply expressive completeness. *J. Philosophical Logic*, 23(5):535–573, 1994.

15 Ian M. Hodkinson and András Simon. The k-variable property is stronger than H-dimension k. *J. Philosophical Logic*, 26(1):81–101, 1997.

16 Paul Hunter, Joël Ouaknine, and James Worrell. Expressive Completeness for Metric Temporal Logic. In *LICS*, pages 349–357. IEEE Computer Society, 2013.

17 Neil Immerman. Upper and Lower Bounds for First Order Expressibility. *J. Comput. Syst. Sci.*, 25(1):76–98, 1982.

18 Neil Immerman. DSPACE[n^k] = VAR[k+1]. In *Structure in Complexity Theory Conference*, pages 334–340. IEEE Computer Society, 1991.

19 Neil Immerman and Dexter Kozen. Definability with Bounded Number of Bound Variables. *Inf. Comput.*, 83(2):121–139, 1989.

20 H. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, Los Angeles, 1968.

21 Michal Koucký, Clemens Lautemann, Sebastian Poloczek, and Denis Thérien. Circuit Lower Bounds via Ehrenfeucht-Fraisse Games. In *IEEE Conference on Computational Complexity*, pages 190–201. IEEE Computer Society, 2006.

22 M. Lange. Model checking propositional dynamic logic with all extras. *Journal of Applied Logic*, 4(1):39–49, 2006.

23 M. Lange and C. Lutz. 2-ExpTime lower bounds for Propositional Dynamic Logics with intersection. *Journal of Symbolic Logic*, 70(5):1072–1086, 2005.

24 Carsten Lutz and Dirk Walther. PDL with negation of atomic programs. *Journal of Applied Non-Classical Logics*, 15(2):189–213, 2005.

25 Bruno Poizat. Deux Ou Trois Choses Que Je Sais de Ln. *Journal of Symbolic Logic*, 47(3):641–658, 1982.

26 Benjamin Rossman. On the constant-depth complexity of k-clique. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 721–730. ACM, 2008.

27 R. S. Streett. Propositional Dynamic Logic of Looping and Converse. In *Proceedings of STOC'81*, pages 375–383. ACM, 1981.

28 Alfred Tarski and Steven R. Givant. *A formalization of set theory without variables*. American Mathematical Society Providence, R.I, 1987.

29 Yde Venema. Expressiveness and Completeness of an Interval Tense Logic. *Notre Dame Journal of Formal Logic*, 31(4):529–547, 1990.

A Linear Upper Bound on the Weisfeiler-Leman Dimension of Graphs of Bounded Genus

Martin Grohe
RWTH Aachen University, Aachen, Germany
grohe@cs.rwth-aachen.de

Sandra Kiefer
RWTH Aachen University, Aachen, Germany
kiefer@cs.rwth-aachen.de

Abstract

The Weisfeiler-Leman (WL) dimension of a graph is a measure for the inherent descriptive complexity of the graph. While originally derived from a combinatorial graph isomorphism test called the Weisfeiler-Leman algorithm, the WL dimension can also be characterised in terms of the number of variables that is required to describe the graph up to isomorphism in first-order logic with counting quantifiers.

It is known that the WL dimension is upper-bounded for all graphs that exclude some fixed graph as a minor [17]. However, the bounds that can be derived from this general result are astronomic. Only recently, it was proved that the WL dimension of planar graphs is at most 3 [26].

In this paper, we prove that the WL dimension of graphs embeddable in a surface of Euler genus g is at most $4g + 3$. For the WL dimension of graphs embeddable in an orientable surface of Euler genus g, our approach yields an upper bound of $2g + 3$.

2012 ACM Subject Classification Theory of computation → Finite Model Theory; Mathematics of computing → Graph theory

Keywords and phrases Weisfeiler-Leman algorithm, finite-variable logic, isomorphism testing, planar graphs, bounded genus

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.117

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version `https://arxiv.org/abs/1904.07216`

1 Introduction

The Weisfeiler-Leman (WL) algorithm is a simple combinatorial graph isomorphism test. The 1-dimensional version of the algorithm, also known as colour refinement and naive vertex classification, is known since at least the mid 1960s, and it is used as a subroutine in almost all practical graph isomorphism tools (see, for instance, [9, 25, 34, 35]), but also in machine learning (see, for instance, [1, 22, 29, 37, 40]). The 2-dimensional version can be traced back to an article by Weisfeiler and Leman that appeared 50 years ago [41]. It is closely related to the algebraic theory of coherent configurations. The generalisation to higher dimensions is due to Babai (see [6, 8]), and again it plays an important role as a subroutine in graph isomorphism algorithms, albeit more on the theoretical side. Notably, Babai uses the $(\log n)$-dimensional version in his quasipolynomial isomorphism test [6].

The connection between the WL algorithm and logic was made by Immerman and Lander [24] and Cai, Fürer, and Immerman [8]. They showed that two graphs are distinguished by the k-dimensional WL algorithm if and only if they can be distinguished in the logic C^{k+1}, the $(k + 1)$-variable fragment of first-order logic using counting quantifiers of the form $\exists^{\geq m} x$. The connection between the WL algorithm and logical definability is at the core of some of the most interesting developments in descriptive complexity theory (see, for

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 117; pp. 117:1–117:15


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

example, [17, 23, 39]). Only recently, it has been noted that the WL algorithm (and thus the finite-variable counting logic) has further surprising characterisations. In a breakthrough paper, Atserias and Maneva [4] showed that the dimension k of the WL algorithm required to distinguish two graphs corresponds to the level of the Sherali-Adams relaxation of the natural integer linear program for graph isomorphism testing (also see [21, 33]). This spawned a lot of work relating the WL algorithm to semidefinite programming [5, 38] and algebraic (Gröbner basis) approaches [7, 13] to graph isomorphism testing. These results can also be phrased in terms of propositional proof complexity. The latest facet of the theory is a characterisation in terms of homomorphism counts of graphs of tree width k [10]. Various aspects of the WL algorithm and its relation to logic have been studied in detail in recent years (see, for instance, [2, 3, 12, 27, 28, 31]).

Cai, Fürer, and Immerman [8] proved that for every k there are non-isomorphic (3-regular) graphs G_k, H_k of size $O(k)$ that are not distinguished by the k-dimensional WL algorithm. Thus, as an isomorphism test, the k-dimensional WL algorithm is incomplete. But, in view of the wide variety of seemingly unrelated combinatorial, logical, and algebraic characterisations of the algorithm, *we are convinced that the structural information the algorithm does detect is of fundamental importance.* The basic parameter of the algorithm is the dimension, corresponding to the number of variables in logical and the degree of polynomials in algebraic characterisations. It yields a structural invariant called the *WL dimension* of a graph G [17], defined to be the least k such that the k-dimensional WL algorithm distinguishes G from every graph H not isomorphic to G (we say that k-WL *identifies* G), or equivalently, the least k such that G can be characterised up to isomorphism (or *identified*) in the logic C^{k+1}. It is also convenient to define the *WL dimension* of a class $\mathcal{C}$ of graphs to be the maximum of the WL dimensions of all graphs in $\mathcal{C}$ if this maximum exists, and ∞ otherwise. We see the WL dimension as a measure for the inherent combinatorial or descriptive complexity of a graph or a class of graphs. We are mostly interested in the relation between the WL dimension and other graph invariants.

Work in descriptive complexity shows that the WL dimension is bounded for many natural graph classes, among them trees [24], graphs of bounded tree width [19], planar graphs [14], graphs of bounded genus [15, 16], all graph classes that exclude some fixed graph as a minor [17], interval graphs [30, 32], and graphs of bounded rank width [20]. However, most of these results do not give explicit bounds on the WL dimension, and the bounds that can be derived from the proofs are usually bad. Only recently, the second author of this paper, jointly with Ponomarenko and Schweitzer, established an almost tight bound for planar graphs [26]: the WL dimension of planar graphs is at most 3, and there are planar graphs of WL dimension 2.

In this paper, we establish bounds for graphs that can be embedded into an arbitrary surface, for example, a torus or a projective plane. By the classification theorem for surfaces (see [36, Theorem 3.1.3]), up to homeomorphism (that is, topological equivalence), all surfaces fall into only two countably infinite families, the family $(\boldsymbol{S}_k)_{k\geq 0}$ of orientable surfaces and the family $(\boldsymbol{N}_\ell)_{\ell\geq 1}$ of non-orientable surfaces. For example, the sphere $\boldsymbol{S}_0$, the torus $\boldsymbol{S}_1$, and the double torus $\boldsymbol{S}_2$ are the first three orientable surfaces, and the projective plane $\boldsymbol{N}_1$ and the Klein bottle $\boldsymbol{N}_2$ are the first two non-orientable surfaces. The *Euler genus* $\mathrm{eg}(\boldsymbol{S})$ of a surface $\boldsymbol{S}$ is $2k$ if $\boldsymbol{S}$ is homeomorphic to the orientable surface $\boldsymbol{S}_k$, and ℓ if $\boldsymbol{S}$ is homeomorphic to the non-orientable surface $\boldsymbol{N}_\ell$. We define the *Euler genus* of a graph G to be the least g such that G is embeddable (that is, can be drawn without edge crossings) in a surface of Euler genus g. Ssee Figure 1 for an example.

▶ **Theorem 1.** *The WL dimension of the class of graphs of Euler genus g is at most $4g + 3$.*

For graphs embeddable in orientable surfaces, we can improve the bound further.

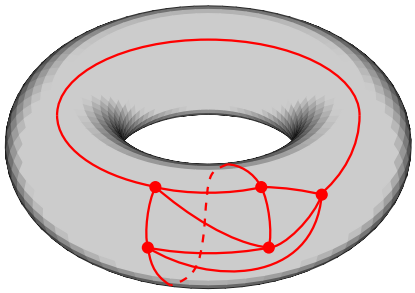


Figure 1 Embedding of K_5 into the torus.

▶ **Corollary 2.** *The WL dimension of the class of graphs embeddable in an orientable surface of Euler genus g is at most $2g + 3$.*

As mentioned above, it was first proved in [15] that the WL dimension of graphs of bounded genus is bounded. A more detailed proof of the same result can be found in the journal paper [16]. The proof of [16] only yields an asymptotically quadratic bound (in terms of the genus) on the dimension, but neither of the two papers gives an explicit bound. It seems that the proof of [15] gives a linear bound, albeit with a large constant factor of at least 80 (not all details are worked out there, so it is difficult to determine the exact bound). The proof in both of these papers is based on the fact that sufficiently large graphs of minimum degree at least 3, embedded in a surface, will have a facial cycle of length at most 6. The proof we give here is completely different. It is based on the straightforward idea of removing a non-contractible cycle to reduce the Euler genus and then using induction. The problem with this idea is that we cannot define non-contractible cycles, but rather only families of such cycles that may intersect in complicated patterns. Understanding these families leads to significant technical complications, but in the end enables us to obtain a much better bound than the simpler proofs of [15, 16]. Our proof is based on a simplified version of a construction from [17, Chapter 15], applied there to graphs "almost embeddable" in a surface.

Outline of the Paper

In Section 2, we present the conventions as well as some topological notions and facts that we use throughout the paper. In Section 3, we introduce the WL dimension and relate it to logic. In Section 4, we present the graph-theoretic machinery that we need in the proof of our main theorem. The proof is outlined in Section 5. The detailed proof is long and complicated and can be found in [18], which also contains further material with respect to all other sections.

2 Preliminaries

2.1 Graphs

We use a standard graph terminology and notation. The only slightly unusual object is our version of coloured graphs. In an *arc-coloured* graph, we colour both vertices and orientations of edges. Formally, an arc-coloured graph is a graph G together with a function $\chi\colon \{(u,u) \mid u \in V(G)\} \cup \{(u,v) \mid \{u,v\} \in E(G)\} \to \mathcal{C}$, where $\mathcal{C}$ is some set of colours. We interpret $\chi(u,u)$ as the colour of the vertex u. Whenever we refer to coloured graphs in this paper, we mean arc-coloured graphs.

2.2 Topology

We have already discussed surfaces and their Euler genus in the introduction. In our presentation and notation, we follow [17, Chapter 9]. As an important notational convention, we always use bold face letters to denote topological spaces. Many more details on surface topology can be found in [17, 36], and in [11, Appendix B].

For a topological space $\boldsymbol{X}$ and a subset $\boldsymbol{Y} \subseteq \boldsymbol{X}$, we define the *boundary* of $\boldsymbol{Y}$ in $\boldsymbol{X}$ to be the set $\boldsymbol{bd_X(Y)}$ of all points $x \in \boldsymbol{X}$ such that every neighbourhood of x has a nonempty intersection with both $\boldsymbol{Y}$ and $\boldsymbol{X} \setminus \boldsymbol{Y}$. We omit the subscript $\boldsymbol{X}$ if the space, usually a surface, is clear from the context.

A *closed disk* is a homeomorphic image of $\{x \in \mathbb{R}^2 \mid \|x\| \leq 1\}$ equipped with the usual topology, and an *open disk* is a subspace of $\mathbb{R}^2$ homeomorphic to $\mathbb{R}^2$ (viewed as a topological space). Let $\boldsymbol{g}$ be a simple closed curve in a surface $\boldsymbol{S}$. Then $\boldsymbol{g}$ is *contractible* if it is the boundary of a closed disk in $\boldsymbol{S}$, otherwise g is *non-contractible*. If $\boldsymbol{g}$ is non-contractible, we can obtain one or two surfaces of strictly smaller Euler genus by the following construction: we cut the surface along $\boldsymbol{g}$; what remains is a surface with one or two holes in it. Then we glue a disk onto each hole and obtain one or two simpler surfaces.

It will be important for us to distinguish between graphs in their standard combinatorial form – we refer to them as *abstract graphs* – and *embedded graphs*. The vertices of a graph *embedded* in a surface $\boldsymbol{S}$ are points in $\boldsymbol{S}$, and the edges are simple curves connecting the vertices in such a way that they do not cross. If G is a graph embedded in $\boldsymbol{S}$, we denote by $\boldsymbol{G}$ the subset of $\boldsymbol{S}$ consisting of all points that are either vertices of G or contained in an edge of G. The *faces* of G are the arcwise connected components of $\boldsymbol{S} \setminus \boldsymbol{G}$.

We say that an abstract graph G is *embeddable* into a surface $\boldsymbol{S}$ if it is isomorphic to (the underlying graph of) a graph embedded in $\boldsymbol{S}$. The *Euler genus* $\mathrm{eg}(G)$ of a graph G is the smallest g such that G is embeddable into a surface of Euler genus g. The graphs of Euler genus 0 are precisely the *planar graphs*, because a graph is embeddable into the 2-sphere $\boldsymbol{S}_0$ if and only if it is embeddable into the plane $\boldsymbol{R}^2$. The class of all graphs of Euler genus at most g is denoted by $\mathcal{E}_g$.

A graph G is *polyhedrally embedded* in a surface $\boldsymbol{S}$ if G is embedded in $\boldsymbol{S}$, 3-connected, and every non-contractible simple closed curve $\boldsymbol{g} \subseteq \boldsymbol{S}$ intersects $\boldsymbol{G}$ in at least three points. Just like 3-connected graphs embedded in a plane, polyhedrally embedded graphs have many nice properties that we will exploit here.

2.3 Logic

C is the extension of first-order logic FO by *counting quantifiers* $\exists^{\geq m} x$ with the obvious meaning. C is only a syntactical extension of FO, because $\exists^{\geq m} x \varphi(x)$ is equivalent to $\exists x_1 \ldots \exists x_m \Big(\bigwedge_{i \neq j} x_i \neq x_j \wedge \bigwedge_i \varphi(x_i) \Big)$. However, we are mainly interested in the fragments C^k of C consisting of all formulae with at most k variables. If $m > k$, then $\exists^{\geq m} x$ cannot be expressed in the k-variable fragment of FO, this is why we add the counting quantifiers. The logics C^k have played an important role in finite-model theory since the 1980s.

We often write $\varphi(x_1, \ldots, x_\ell)$ to indicate that the free variables of φ are among $x_1, \ldots, x_\ell$. (Not all of these variables are required to appear in φ.) Then for a graph G and vertices $u_1, \ldots, u_\ell \in V(G)$, we write $G \models \varphi(u_1, \ldots, u_\ell)$ to denote that G satisfies φ if for all i the variable x_i is interpreted by u_i. Moreover, we write $\varphi[G, u_1, \ldots, u_i, x_{i+1}, \ldots, x_\ell]$ to denote the set of all $(\ell - i)$-tuples $(u_{i+1}, \ldots, u_\ell)$ such that $G \models \varphi(u_1, \ldots, u_\ell)$.

The *width* of a formula $\varphi \in \mathsf{C}$ is the maximum number of free variables of a subformula of φ. Clearly, every formula in C^k has width at most k. An important observation that we often use is that every C-formula of width at most k is equivalent to a C^k-formula. We denote the set of all C-formulae of width at most k by $\mathsf{C}^k_{\mathsf{w}}$.

3 The WL Dimension

We start by reviewing the *k-dimensional WL algorithm* (for short: *k-WL*) for $k \geq 1$.

The *atomic type* $\mathrm{atp}(G, \bar{u})$ of a k-tuple $\bar{u} = (u_1, \dots, u_k)$ of vertices of a (possibly coloured) graph G is the set of all atomic formulae satisfied by these vertices. The exact encoding is not important for us, the relevant property is that tuples $\bar{u} = (u_1, \dots, u_k)$ and $\bar{v} = (v_1, \dots, v_k)$ of vertices of graphs G and H, respectively, have the same atomic type if and only if the mapping $u_i \mapsto v_i$ is an isomorphism from the induced subgraph $G[\{u_1, \dots, u_k\}]$ to the induced subgraph $H[\{v_1, \dots, v_k\}]$.

Now k-WL is the algorithm that, given a graph G, computes the following sequence of "colourings" C^k_i of $\big(V(G)\big)^k$ for $i \geq 0$ until it returns $C^k_\infty = C^k_i$ for the smallest i such that for all $\bar{u}, \bar{v}$ it holds that $C^k_i(\bar{u}) = C^k_i(\bar{v}) \iff C^k_{i+1}(\bar{u}) = C^k_{i+1}(\bar{v})$. The initial colouring C^k_0 assigns to each tuple its atomic type: $C^k_0(\bar{u}) \coloneqq \mathrm{atp}(G, \bar{u})$. In the $(i+1)$-st *refinement round*, the colouring C^k_{i+1} is defined by $C^k_{i+1}(\bar{u}) \coloneqq \big(C^k_i(\bar{u}), M_i(\bar{u})\big)$, where, for $\bar{u} = (u_1, \dots, u_k)$, $M_i(\bar{u})$ is the multiset

$$\Big\{\!\!\Big\{ \Big(\mathrm{atp}\big(G, (u_1, \dots, u_k, v)\big), C^k_i(u_1, \dots, u_{k-1}, v), \\ C^k_i(u_1, \dots, u_{k-2}, v, u_k), \dots, C^k_i(v, u_2, \dots, u_k)\Big) \;\Big|\; v \in V \Big\}\!\!\Big\}$$

We say that k-WL *distinguishes* two graphs G and H if there is some colour c in the range of C^k_∞ such that the number of tuples $\bar{u} \in \big(V(G)\big)^k$ with $C^k_\infty(\bar{u}) = c$ is different from the number of tuples $\bar{v} \in \big(V(H)\big)^k$ with $C^k_\infty(\bar{v}) = c$. We say that k-WL *identifies* G if it distinguishes G from all graphs H not isomorphic to G. The *WL dimension* of G is the smallest k such that k-WL identifies G.

In this paper, we reason about the WL dimension in terms of logic, using the following theorem.

▶ **Theorem 3** ([8, 24]). *Let $k \geq 1$. Let G and H be graphs, possibly coloured, and $\bar{u} = (u_1, \dots, u_k) \in \big(V(G)\big)^k$ and $\bar{v} = (v_1, \dots, v_k) \in \big(V(H)\big)^k$. Then the following are equivalent:*

1. $C^k_\infty(\bar{u}) = C^k_\infty(\bar{v})$;
2. $G \models \varphi(u_1, \dots, u_k) \iff H \models \varphi(v_1, \dots, v_k)$ *for all C^{k+1}-formulae $\varphi(x_1, \dots, x_k)$.*

We say that a graph G is *identified* by the logic C^k if there is a sentence $\mathsf{iso}_G \in \mathsf{C}^k$ such that for all graphs H we have $H \models \mathsf{iso}_G$ if and only if H is isomorphic to G.

▶ **Corollary 4.** *A graph has WL dimension k if and only if it is identified by C^{k+1}.*

The WL dimension of the class of planar graphs is at most 3 [26]. Using the previous corollary, we can re-phrase this as follows.

▶ **Theorem 5** ([26]). *For every planar graph G there is a C^4-sentence iso_G that identifies G.*

4 Shortest Path Systems, Patches and Necklaces

Here we introduce the graph-theoretic machinery necessary to prove our main theorem. Essentially, the definitions and results of this section are from [17, Chapter 15]. In fact, things are simpler here because [17, Chapter 15] deals with graphs *almost* embedded in a surface, whereas we only need to consider embedded graphs. Sometimes, we need to change the definitions in order to improve the resulting bounds on the WL dimension later. Notably, our *necklaces* play the role of the *belts* in [17], but the definition is slightly different. This also requires an adaptation of the proof that reducing necklaces exist.

▶ **Definition 6.** *Let G be a graph and $u, u' \in V(G)$. A* shortest path system (sps) from u to u' *is a family $\mathcal{Q}$ of shortest paths in G from u to u' such that every shortest path from u to u' in the subgraph $\bigcup_{Q\in\mathcal{Q}} Q$ is contained in $\mathcal{Q}$.*

We let $V(\mathcal{Q}) \coloneqq \bigcup_{Q\in\mathcal{Q}} V(Q)$ and $E(\mathcal{Q}) \coloneqq \bigcup_{Q\in\mathcal{Q}} E(Q)$ and $G(\mathcal{Q}) \coloneqq \big(V(\mathcal{Q}), E(\mathcal{Q})\big) = \bigcup_{Q\in\mathcal{Q}} Q$. We call $\mathcal{Q}$ trivial *if $|V(\mathcal{Q})| \leq 2$, that is, if $G(\mathcal{Q})$ consists of a single vertex or a single edge.*

The height $\mathrm{ht}^{\mathcal{Q}}(v)$ *of $v \in V(\mathcal{Q})$ is the distance from u to v. The vertices in $\bigcap_{Q\in\mathcal{Q}} V(Q)$ are the* articulation vertices *of $\mathcal{Q}$. An articulation vertex v is* proper *if $v \neq u$ and $v \neq u'$. We denote the set of all articulation vertices of $\mathcal{Q}$ by* $\mathrm{art}(\mathcal{Q})$.

For all $u, u' \in V(G)$ such that there is a path from u to u' in G, the canonical sps from u to u' in G *is the set $\mathcal{Q}^G(u, u')$ of all shortest paths from u to u' in G.*

While shortest paths systems are defined with respect to abstract graphs, the following notions are defined with respect to embedded graphs. For the rest of the section, we make the following assumption.

▶ **Assumption 7.** *G is a graph polyhedrally embedded in a surface $\boldsymbol{S}$ of Euler genus $g \geq 1$.*

▶ **Definition 8.** *A* patch *in G is an sps $\mathcal{Q}$ in G such that:*

(i) *$\mathcal{Q}$ has no proper articulation vertices.*

(ii) *There is a closed disk $\boldsymbol{D} \subseteq \boldsymbol{S}$ such that $\boldsymbol{G(\mathcal{Q})} \subseteq \boldsymbol{D}$.* ⌟

It can be shown that if $\mathcal{Q}$ is a non-trivial patch (i.e., a patch that does not consist of just a single vertex or a single edge), then there is a unique disk $\boldsymbol{D}(\mathcal{Q})$ such that $\boldsymbol{G(\mathcal{Q})} \subseteq \boldsymbol{D}(\mathcal{Q})$ and there is a cycle $C(\mathcal{Q}) \subseteq G(\mathcal{Q})$ such that $\boldsymbol{bd}\big(\boldsymbol{D}(\mathcal{Q})\big) = \boldsymbol{C}(\mathcal{Q})$. Furthermore, there are two paths $Q, Q' \in \mathcal{Q}$ such that $C(\mathcal{Q}) = Q \cup Q'$.

We call a subgraph $H \subseteq G$ *simplifying* if every connected component of $G \setminus H$ is contained in $\mathcal{E}_{g-1}$. Otherwise, H is *non-simplifying*.

▶ **Lemma 9** ([17], Corollary 15.3.5)**.** *For every non-simplifying subgraph $H \subseteq G$, there is exactly one connected component A^* of $G \setminus H$ with $A^* \notin \mathcal{E}_{g-1}$, and all other connected components are planar.*

A patch $\mathcal{Q}$ is *simplifying* if the graph $G(\mathcal{Q})$ is. It turns out that non-simplifying patches form the basic building blocks of our theory. Let $\mathcal{Q}$ be a non-trivial non-simplifying patch. Let A^* be the unique connected component of $G \setminus V(\mathcal{Q})$ that is not planar (the existence and uniqueness of A^* follows from Lemma 9). Then we define G/A^* to be the graph obtained from G by contracting the subgraph A^* to a single vertex a^*. By [17, Corollary 15.4.5], G/A^* is a 3-connected planar graph. Figure 2 displays a schematic view of a patch $\mathcal{Q}$ with some attached (planar) connected components as well as the non-planar component A^*, the disk $\boldsymbol{D}(\mathcal{Q})$, and the boundary cycle $\boldsymbol{C}(\mathcal{Q})$.

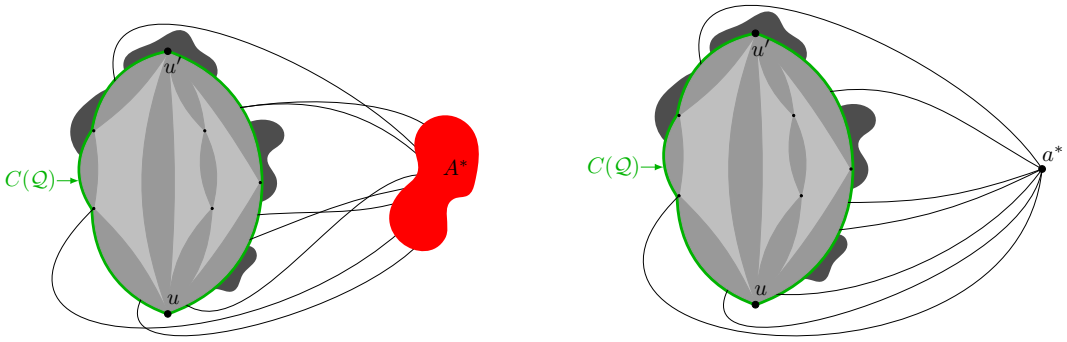


Figure 2 Left: A patch $\mathcal{Q}$ with non-planar component A^* and boundary cycle $C(\mathcal{Q})$. The curve $\boldsymbol{C}(\mathcal{Q})$ is the boundary of the disk $\boldsymbol{D}(\mathcal{Q})$, which consists of the light gray and medium gray areas. Right: the (planar) factor graph G/A^*.

We define the *internal graph* of a non-trivial non-simplifying patch $\mathcal{Q}$ to be the graph $I := I(\mathcal{Q})$ with vertex set $V(I) := V(G) \cap \boldsymbol{D}(\mathcal{Q})$ and edge set $E(I) := \{\boldsymbol{e} \in E(G) \mid \boldsymbol{e} \subseteq \boldsymbol{D}(\mathcal{Q})\}$. Note that $C(\mathcal{Q}) \subseteq I$. The definitions of the graphs $C(\mathcal{Q})$ and $I(\mathcal{Q})$ do not only depend on the abstract graph G and the sps $\mathcal{Q}$, but on the embedding of G in $\boldsymbol{S}$. However, it can be proved that actually the graphs are invariant under embeddings.

▶ **Lemma 10** ([17]). *Let $\mathcal{Q}$ be a non-trivial non-simplifying patch in G. Let G' be a graph embedded in a surface $\boldsymbol{S}'$ of Euler genus g such that G and G' are isomorphic (as abstract graphs), and let f be an isomorphism from G to G'. Then $\mathcal{Q}' := f(\mathcal{Q})$ is a non-simplifying patch in G', and it holds that $f\big(C(\mathcal{Q})\big) = C(\mathcal{Q}')$ and $f\big(I(\mathcal{Q})\big) = I(\mathcal{Q}')$.*

This follows from [17, Lemma 15.4.10]. Intuitively, the reason why this holds is that the 3-connected planar graph G/A^* has a "unique" embedding.

▶ **Corollary 11.** *Let $u, u' \in V(G)$ and $\mathcal{Q} := \mathcal{Q}^G(u, u')$ such that $\mathcal{Q}$ is a non-trivial non-simplifying patch. Let f be an automorphism of G such that $f(u) = u$ and $f(u') = u'$. Then $f\big(C(\mathcal{Q})\big) = C(\mathcal{Q})$ and $f\big(I(\mathcal{Q})\big) = I(\mathcal{Q})$.*

We remark that the analogue of Corollary 11 for simplifying patches does not hold (see [18, Figure 4]). The analysis of simplifying patches is much more involved, and we defer the reader to [18].

The final objects we define in this section are *necklaces*.

▶ **Definition 12.** *A* necklace *in G is a tuple $\mathcal{B} := (u^0, \mathcal{Q}^0, u^1, \mathcal{Q}^1, u^2, \mathcal{Q}^2)$, where $u^0, u^1, u^2 \in V(G)$ and $\mathcal{Q}^i = \mathcal{Q}^G(u^i, u^{i+1})$ (indices taken modulo 3) is the canonical sps from u^i to u^{i+1}, such that the following conditions are satisfied for every $i \in \{0, 1, 2\}$:*

- *u^0, u^1, u^2 are pairwise distinct.*
- *$V(\mathcal{Q}^i) \cap V(\mathcal{Q}^{i+1}) = \{u^{i+1}\}$ (indices modulo 3).*
- *There is a disk $\boldsymbol{D}^i \subseteq \boldsymbol{S}$ such that $\boldsymbol{G(\mathcal{Q}^i)} \subseteq \boldsymbol{D}^i$.* ⌟

For a necklace $\mathcal{B} := (u^0, \mathcal{Q}^0, u^1, \mathcal{Q}^1, u^2, \mathcal{Q}^2)$ we write $V(\mathcal{B})$ for the set $\bigcup_{i=0}^{2} \bigcup_{Q \in \mathcal{Q}^i} V(Q)$ and $E(\mathcal{B})$ for $\bigcup_{i=0}^{2} \bigcup_{Q \in \mathcal{Q}^i} E(Q)$, and we let $G(\mathcal{B}) := \big(V(\mathcal{B}), E(\mathcal{B})\big)$. Moreover, we define the set of *articulation vertices* of $\mathcal{B}$ to be $\mathrm{art}(\mathcal{B}) := \bigcup_{i=0}^{2} \mathrm{art}(\mathcal{Q}^i)$.

▶ **Definition 13.** *A necklace $\mathcal{B} := (u^0, \mathcal{Q}^1, u^1, \mathcal{Q}^2, u^2, \mathcal{Q}^3)$ is* reducing *if there are paths $Q^i \in \mathcal{Q}^i$ such that $B := Q^1 \cup Q^2 \cup Q^3$ is a non-contractible cycle.* ⌟

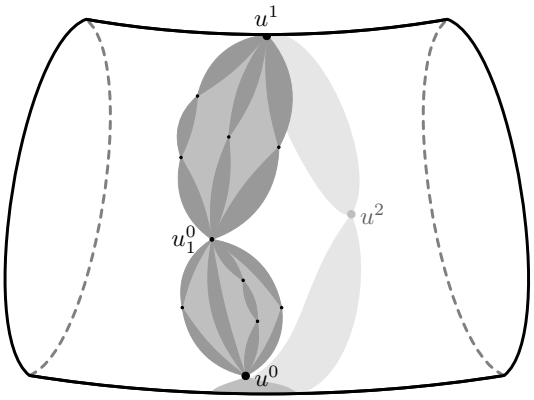


Figure 3 A necklace on a torus section.

We can think of a reducing necklace as a necklace around a handle of our surface (or a crosscap of the surface in the non-orientable case). The beads of the necklace are the disks of the patches that form the necklace. Figure 3 shows a necklace on a torus with articulation vertices u^0, u_1^0, u^1, u^2.

▶ **Lemma 14** (Necklace Lemma)**.** *G has a reducing necklace.*

Essentially, this is [17, Lemma 15.5.8], with the necklaces corresponding to the belts there. But since apart from a renaming, we have also slightly changed the content of the definition of a necklace/belt, the proof needs to be adapted, too. Again, we defer the reader to [18].

5 Upper Bound on the WL Dimension

In this section, we give an outline of the proof of our main theorem (Theorem 1). The full proof can be found in [18].

By the correspondence between k-WL and the logic C^{k+1} stated in Corollary 4, we need to prove that every graph of Euler genus at most g can be identified by a C^{4g+4}-sentence. The proof is by induction on g. The base step $g = 0$ is Theorem 5.

For the inductive step, we make the following assumption.

▶ **Assumption 15.** *$g \geq 1$, and there is a natural number $s \geq 4$ such that every graph in $\mathcal{E}_{g-1}$ can be identified by a C^s-sentence.*

We shall prove that every graph in $\mathcal{E}_g$ can be defined by a C^{s+4}-sentence. Then Theorem 1 follows by induction. After some fairly straightforward reduction steps, which include the reduction to arc-coloured 3-connected graphs from [26], we find that it suffices to prove the following lemma.

▶ **Lemma 16.** *Let G be a coloured graph polyhedrally embedded in a surface $\boldsymbol{S}$ of Euler genus g. Then there is a sentence $\mathsf{iso}_G \in \mathsf{C}^{s+3}$ that identifies G.*

For the rest of the section, we fix a positive integer n. The intended meaning of n is that it is the size of the target graph G. At this point, we have fixed three numerical parameters: the Euler genus g, the number s of variables required to identify graphs of smaller Euler genus, and the order n.

We start the proof by showing that basic objects such as shortest path systems, (pseudo-) patches and (pseudo-)necklaces are definable in the logic C^{s+3}. The following lemma illustrates the type of definability result we can expect, more of the same kind can be found in the full version [18]. Instead of definability in C^k, we actually always study definability in $\mathsf{C}^k_{\mathsf{w}}$ (see Section 2.3) and exploit the fact that every $\mathsf{C}^k_{\mathsf{w}}$-formula is equivalent to a C^k-formula.

▶ **Lemma 17.** *There are formulae* $\mathsf{csps\text{-}vert}(x,x',y) \in \mathsf{C}^3_{\mathsf{w}}$ *and* $\mathsf{csps\text{-}edge}(x,x',y_1,y_2) \in \mathsf{C}^4_{\mathsf{w}}$*, such that for all connected graphs* G *of order* $|G| \leq n$ *and all vertices* $u,u' \in V(G)$*,*

$$\mathsf{csps\text{-}vert}[G,u,u',y] = V\big(\mathcal{Q}^G(u,u')\big),$$
$$\mathsf{csps\text{-}edge}[G,u,u',y_1,y_2] = E\big(\mathcal{Q}^G(u,u')\big).$$

Recall that $\mathcal{Q}^G(u,u')$ is the canonical sps from u to u', that is, the set of all shortest paths from u to u'.

Proof. It is straightforward to define, for every $k \geq 0$, a $\mathsf{C}^3_{\mathsf{w}}$-formula $\mathrm{dist}_{=k}(x,y)$ stating that the distance between the vertices x and y is exactly k. Then we let $\mathsf{csps\text{-}vert}(x,x',y) := \bigvee_{k=0}^{n} \Big(\mathsf{dist}_{=k}(x,x') \wedge \bigvee_{i=0}^{k} \big(\mathsf{dist}_{=i}(x,y) \wedge \mathsf{dist}_{=k-i}(y,x')\big)\Big)$.
The formula $\mathsf{csps\text{-}edge}(x,x',y_1,y_2)$ can be defined similarly. ◀

With a little more effort, we can prove the following lemma.

▶ **Lemma 18.** *Let* $h < g$*. Then there is a formula* $\mathsf{csps\text{-}comp\text{-}genus}_h(x,x',y) \in \mathsf{C}^{s+2}_{\mathsf{w}}$ *such that for all connected graphs* G *of order* $|G| \leq n$ *and all* $u,u',v \in V(G)$ *the following holds. Let* $\mathcal{Q} := \mathcal{Q}^G(u,u')$*, and let* A *be the connected component of* v *in* $G \setminus G(\mathcal{Q})$ *(assuming* $v \notin V(\mathcal{Q})$*). Then*

$$G \models \mathsf{csps\text{-}comp\text{-}genus}_h(u,u',v) \iff v \notin V(\mathcal{Q}) \text{ and } \mathrm{eg}(A) \leq h.$$

Proof. It follows from Assumption 15 that for every $h < g$, there is a sentence genus_h such that for all graphs G of order $|G| \leq n$, it holds that $G \models \mathsf{genus}_h \iff \mathrm{eg}(G) \leq h$. Indeed, we can simply let genus_h be the disjunction of all sentences iso_H identifying the graphs H with $\mathrm{eg}(H) \leq h$ and $|H| \leq n$.

Using careful bookkeeping and some tricks to reduce the number of variables. we can combine genus_h with the formulae defined in Lemma 17 to obtain the desired formula. ◀

▶ **Corollary 19.** *There is a formula* $\mathsf{csps\text{-}simplifying}(x,x') \in \mathsf{C}^{s+2}_{\mathsf{w}}$ *such that for all connected graphs* $G \in \mathcal{E}_g$ *of order* $|G| \leq n$ *and all* $u,u' \in V(G)$*,*

$$G \models \mathsf{csps\text{-}simplifying}(u,u') \iff \mathcal{Q}^G(u,u') \text{ is simplifying.}$$

The formulae we have defined so far make no reference to an embedding of the input graph. However, if we want to talk about patches and necklaces, we need to take the embedding into account. For the rest of the section, we fix a specific embedded graph G.

▶ **Assumption 20.** *G is a coloured graph of order $|G| = n$ that is polyhedrally embedded in a surface $\boldsymbol{S}$ of Euler genus g.*

It is our goal to construct a $\mathsf{C}^{s+3}_{\mathsf{w}}$-sentence that identifies G. The following lemma is a key step towards this goal, and we find it worthwhile to go into some of the details of its proof.

▶ **Lemma 21.** *There are* $\mathsf{C}^7_{\mathsf{w}}$*-formulae* $\mathsf{int\text{-}vert}(x, x', y)$ *and* $\mathsf{int\text{-}edge}(x, x', y_1, y_2)$ *such that for all vertices* $u, u' \in V(G)$ *for which* $\mathcal{Q} \coloneqq \mathcal{Q}^G(u, u')$ *is a non-trivial non-simplifying patch, the following holds:*

$$\mathsf{int\text{-}vert}[G, u, u', y] = V\big(I(\mathcal{Q})\big),$$
$$\mathsf{int\text{-}edge}[G, u, u', y_1, y_2] = E\big(I(\mathcal{Q})\big).$$

Intuitively, the lemma says that even though the logical formulae only have access to the abstract graph and the disk of a patch and the internal graph depend on the embedding, we can still define the internal graph. This is non-trivial and somewhat surprising.

Proof. Let $u, u' \in V(G)$ such that $\mathcal{Q} \coloneqq \mathcal{Q}^G(u, u')$ is a non-trivial non-simplifying patch. Let $\boldsymbol{D} \coloneqq \boldsymbol{D}(\mathcal{Q})$, $C \coloneqq C(\mathcal{Q})$, and $I \coloneqq I(\mathcal{Q})$ (see Section 4).

By Lemma 9, the graph $G \setminus G(\mathcal{Q})$ has a unique non-planar connected component A^*. Since we can detect the planar connected components using the fact that we can identify all planar graphs in C^4, we can also detect A^* as the only non-planar component. This allows us to construct a $\mathsf{C}^7_{\mathsf{w}}$-formula $\mathsf{astar}(x, x', y)$ such that $\mathsf{astar}[G, u, u', y] = V(A^*)$.

Let v_1 be a vertex in $V(\mathcal{Q})$ that is adjacent to A^* and among all such vertices has minimal height in the sps $\mathcal{Q}$, and let h be this height. Since A^* is embedded outside of the disk $\boldsymbol{D}$, the vertex v_1 must be on the boundary cycle C of $\boldsymbol{D}$. There is at most one other vertex of height h on this cycle. Thus, even though v_1 is not unique, there are at most two choices. If there is a second vertex of height h adjacent to A^*, let us call it v_1'. Using the formula $\mathsf{astar}(x, x', y)$ and the fact that vertices of a certain height are definable in $\mathsf{C}^3_{\mathsf{w}}$, we can construct a $\mathsf{C}^7_{\mathsf{w}}$-formula $\varphi_1(x, x', y_1)$ such that v_1 and possibly v_1' are the only vertices in $\varphi_1[G, u, u', y_1]$.

Recall that G/A^* denotes the graph obtained from G by contracting the connected subgraph A^* to a single vertex, which we call a^*, and that the graph G/A^* is a 3-connected planar graph. By Whitney's Theorem, the facial subgraphs (i.e., the subgraphs induced by the boundaries of the faces of an embedding) of a 3-connected plane graph are precisely the chordless non-separating cycles. In particular, they are independent of the embedding. Furthermore, every edge is contained in exactly two of these facial cycles. Let us consider the edge v_1a^* in the graph G/A^*. Let F and F' be the two facial cycles that contain this edge. Both F and F' contain exactly one neighbour of a^* distinct from v_1. Let v_2, v_2' be these neighbours.

By [26, Lemma 22], if we have a 3-connected planar graph H and three vertices w_1, w_2, w_3 on a common facial cycle, then after individualising these three vertices, the 1-dimensional WL algorithm computes a discrete colouring, i.e., a colouring in which every vertex has its own unique colour. By Theorem 3, this implies that for every vertex w of H there is a formula $\psi_{H,w}(z_1, z_2, z_3, y) \in \mathsf{C}^5_{\mathsf{w}}$ such that $\psi_{H,w}[H, w_1, w_2, w_3, y] = \{w\}$. We apply this to the graph G/A^* and the three vertices a^*, v_1, v_2 and obtain, for every vertex $w \in V(G/A^*) = (V(G) \setminus V(A^*)) \cup \{a^*\}$, a formula $\psi_w(z^*, y_1, y_2, y) \in \mathsf{C}^5_{\mathsf{w}}$ such that $\psi_w[G/A^*, a^*, v_1, v_2, y] = \{w\}$. From this formula and the formula $\mathsf{astar}(x, x', y)$ we can construct a $\mathsf{C}^7_{\mathsf{w}}$-formula $\widetilde{\psi}_w(x, x', y_1, y_2, z)$ such that $\widetilde{\psi}_w[G, u, u', v_1, v_2, z] = \{w\}$. (Unfortunately, this construction is quite tedious; details can be found in [18].)

Since $A^* \cap \boldsymbol{D} = \emptyset$, we have $V(I) = V(G) \cap \boldsymbol{D} \subseteq V(G \setminus A^*)$. We let

$$\delta(x, x', y_1, y_2, z) \coloneqq \bigvee_{w \in V(I)} \widetilde{\psi}_w(x, x', y_1, y_2, z).$$

Then $\delta[G, u, u', v_1, v_2, z] = V(I)$. Thus $\delta(x, x', y_1, y_2, z)$ is almost the formula $\mathsf{int\text{-}vert}(x, x', y)$ we want, except that it has two additional parameters v_1, v_2, which we have to get rid of.

We apply [26, Corollary 27], which says that the 3-dimensional WL algorithm *determines orbits* in coloured 3-connected graphs. This means that it distinguishes two vertices if and only if they belong to different orbits of the automorphism group of the given graph. It follows that for every 3-connected planar graph H and for every orbit O of the automorphism group of H, there is a formula $\xi_{H,O}(y_2) \in \mathsf{C}^4_{\mathsf{w}}$ such that $\xi_{H,O}[H, y_2] = O$.

To eliminate the parameter v_2, we apply the corollary to the graph G/A^*, but only after individualising the vertices a^* and v_1. That is, we modify the colouring such that each of the two vertices gets its own colour and is thus fixed by all automorphisms. Let O_2 be the orbit of v_2 in this coloured graph. By the definition of v_2, either $O_2 = \{v_2, v_2'\}$ or $O_2 = \{v_2\}$. Since the graph G/A^* is 3-connected, by eliminating the colour relations for a^* and v_1 at the cost of new free variables z^* and y_1, we obtain a new formula $\psi_2(z^*, y_1, y_2) \in \mathsf{C}^6_{\mathsf{w}}$ such that $\psi_2[G/A^*, a^*, v_1, y_2] = O_2$. We can transform this formula ψ_2 into a $\mathsf{C}^7_{\mathsf{w}}$-formula $\widetilde{\psi}_2(x, x', y_1, y_2)$ such that $\widetilde{\psi}_2[G, u, u', v_1, y_2] = O_2$. We let

$$\delta'(x, x', y_1, z) \coloneqq \exists y_2 \big(\widetilde{\psi}_2(x, x', y_1, y_2) \wedge \delta(x, x', y_1, y_2, z)\big).$$

If $O_2 = \{v_2\}$, then clearly $\delta'[G, u, u', v_1, z] = \delta[G, u, u', v_1, v_2, z] = V(I)$. So suppose that $O_2 = \{v_2, v_2'\}$, and let f be an automorphism of G with $f(u) = u, f(u') = u', f(v_1) = v_1$, and $f(v_2) = v_2'$. By Corollary 11, we have $f\big(V(I)\big) = V(I)$ and thus

$$\begin{aligned}\delta[G, u, u', v_1, v_2', z] &= \delta[f(G), f(u), f(u'), f(v_1), f(v_2), z] \\ &= f\big(\delta[G, u, u', v_1, v_2, z]\big) \\ &= f\big(V(I)\big) = V(I).\end{aligned}$$

It follows that

$$\delta'[G, u, u', v_1, z] = \delta[G, u, u', v_1, v_2, z] \cup \delta[G, u, u', v_1, v_2', z] = V(I).$$

So we have eliminated the parameter v_2. To eliminate v_1, we use a similar argument, which gives us the formula $\mathsf{int\text{-}vert}(x, x', y)$. The formula $\mathsf{int\text{-}edge}(x, x', y_1, y_2)$ can be constructed similarly. ◀

At this point, the proof of Lemma 16 branches into two cases.

Case 1: G does not contain any simplifying patches

By Lemma 14, the graph G contains a reducing necklace. We fix such a necklace $\mathcal{B} = (u^0, \mathcal{Q}^1, u^1, \mathcal{Q}^2, u^2, \mathcal{Q}^3)$. For this, it is sufficient to fix the three vertices u^0, u^1, u^2, because the $\mathcal{Q}^i$ are canonical shortest path systems. We are going to define a subgraph $\mathrm{Cut}(\mathcal{B})$ of G obtained from G by "cutting through the beads". Since the necklace is reducing, the Euler genus of every connected component of $\mathrm{Cut}(\mathcal{B})$ is at most $g - 1$ and we can identify it via a C^s-sentence using Assumption 15. We colour $\mathrm{Cut}(\mathcal{B})$ in such a way that we can reconstruct G and identify it.

Since in this case, there are no simplifying patches, we can define the internal graphs of all patches that form the necklace using Lemma 21. The cut graph $\mathrm{Cut}(\mathcal{B})$ is obtained from G by removing all trivial patches contained in $\mathcal{B}$, all articulation vertices and the internal graphs of all non-trivial patches contained in $\mathcal{B}$. Since the necklace is reducing, there is a non-contractible simple closed curve through the necklace, and we can choose this curve such that it is disjoint from the cut graph. Thus, the cut graph is embeddable in the surface obtained by cutting $\boldsymbol{S}$ along this non-contractible closed curve and therefore has smaller Euler genus.

To prove that G is identified in the logic $\mathsf{C}^{s+3}_{\mathsf{w}}$, we need to show that for every graph $\widehat{G}$ that is not isomorphic to G, there is a $\mathsf{C}^{s+3}_{\mathsf{w}}$-sentence that distinguishes G and $\widehat{G}$. To prove

this, we show that if G and $\widehat{G}$ satisfy the same $\mathsf{C}_{\mathsf{w}}^{s+3}$-sentences, then they are isomorphic. We use the fact that the necklace $\mathcal{B}$ is definable in G by $\mathsf{C}_{\mathsf{w}}^{7}$-formulae using the three parameters u^0, u^1, u^2. The same formulae define some object $\widehat{B}$ in $\widehat{G}$. We call $\widehat{B}$ a "pseudo-necklace". It is not necessarily a necklace, because $\widehat{G}$ is not necessarily an embedded graph and objects like patches and necklaces, which depend on an embedding, do not exist in $\widehat{G}$. Nevertheless, from the pseudo-necklace we can define a pseudo-cut graph $\mathrm{Cut}(\widehat{\mathcal{B}})$. Since $\mathrm{Cut}(\mathcal{B})$ is identified in the logic $\mathsf{C}_{\mathsf{w}}^{s}$, we can show that $\mathrm{Cut}(\mathcal{B})$ and $\mathrm{Cut}(\widehat{\mathcal{B}})$ are isomorphic. Since we can reconstruct G and $\widehat{G}$ from the respective (pseudo-)cut graphs, we conclude that G and $\widehat{G}$ are isomorphic.

Case 2: G contains a simplifying patch

This case sounds simpler than the first one: instead of a complicated necklace, here we only need to remove a simplifying patch from our graph. The remaining pieces have smaller Euler genus and thus can be identified in the logic $\mathsf{C}_{\mathsf{w}}^{s}$. Hence, all we need to do is colour the pieces in such a way that we can reconstruct the original graph. The problem with this line of reasoning is that simplifying patches have a much more complicated structure than non-simplifying patches. For example, we cannot define the interior of a simplifying patch in the same way as we did for non-simplifying patches in Lemma 21 since there is not necessarily a non-planar connected component which marks the "outside region" of the patch. Therefore, the idea of the proof is to remove the canonical sps and some interior parts of the corresponding patch, which are actually interiors of non-simplifying subpatches and thus definable by Lemma 21.

More precisely, we fix two vertices u and u' such that $\mathcal{Q} \coloneqq \mathcal{Q}^G(u, u')$ is a minimal simplifying canonical patch in G, that is, a simplifying canonical patch all of whose proper canonical subpatches are non-simplifying. We extend $\mathcal{Q}$ by the internal graphs of all proper subpatches and obtain a graph J, which is actually embedded in the disk of $\mathcal{Q}$ and therefore planar.

Now we distinguish between two cases. If $J \setminus \{u, u'\}$ is connected, the patch $\mathcal{Q}$ behaves almost like a non-simplifying patch, and we can argue similarly as in Case 1. If $J \setminus \{u, u'\}$ is disconnected, the patch $\mathcal{Q}$ can be split into several so-called *fibres*. The subgraph J contained in the fibres is definable in our logic when we fix one more particular vertex. In fact, the subgraph contained in every single fibre is definable. We exploit this in order to encode in the boundary vertices of the fibres the way in which the remainder of the graph is attached to them, similarly as in Case 1, but due to the possibly complex structure of J a lot more involved.

An improved bound in case the surface is orientable

The combination of Case 1 and Case 2 yields Theorem 1. Exploiting our inductive approach further, we can deduce a better bound in case we know that the given graph is embeddable in an *orientable* surface of Euler genus at most g, as stated in Corollary 2.

Proof of Corollary 2. The Euler genus of an orientable surface is always even. Suppose G is a graph embeddable in an orientable surface of Euler genus g. Since the subgraphs obtained by cutting through the beads are also embeddable in orientable surfaces of smaller Euler genus, their Euler genus is at least 2 smaller than the Euler genus of G. Therefore, proceeding inductively and redefining s to be the number of variables needed for graphs embeddable in orientable surfaces of Euler genus at most $g - 2$, we can improve our bound from Theorem 1 to $2g + 3$. ◀

6 Concluding Remarks

The WL dimension is a measure for the combinatorial and descriptive complexity of a graph. In view of its numerous, seemingly unrelated characterisations in terms of logic, algebra, mathematical programming, and homomorphisms, we can arguably regard the WL dimension as a natural and robust graph invariant.

We have proved an upper bound of $4g + 3$ for the WL dimension of graphs of Euler genus g and showed that if G is known to be embeddable in an orientable surface of Euler genus g, the bound improves to $2g + 3$. The immediate remaining question is how tight our bound is.

We believe that by refining our arguments in some places it might be possible to reduce the bound from Theorem 1 to $3g + 3$ or even $2g + 3$; any further improvement seems to require substantial additional ideas. It is conceivable that the WL dimension of planar graphs is 2. If this is the case, the additive term in our bound would automatically drop to 2.

In terms of lower bounds, using the so-called CFI construction [8], it is easy to prove a linear lower bound of $\epsilon \cdot g$ for the WL dimension of graphs of Euler genus g, albeit with a rather small constant $\epsilon > 0$. To close the gap between upper and lower bound, it may be worthwhile to spend some effort on improving the lower bound.

Beyond graphs of bounded genus, we can try to determine the WL dimension of other graph classes and tie the WL dimension to other graph invariants. A natural target would be the class of all graphs that exclude the complete graph K_ℓ as a minor. We know that the WL dimension of this class is bounded [17]. But even an exponential bound on the WL dimension in terms of ℓ would be major progress.

References

1 B. Ahmadi, K. Kersting, M. Mladenov, and S. Natarajan. Exploiting symmetries for scaling loopy belief propagation and relational training. *Machine Learning Journal*, 92(1):91–132, 2013.

2 V. Arvind, J. Köbler, G. Rattan, and O. Verbitsky. On Tinhofer's Linear Programming Approach to Isomorphism Testing. In G. F. Italiano, G. Pighizzini, and D. Sannella, editors, *Proceedings of the 40th International Symposium on Mathematical Foundations of Computer Science 2015 - 40th International Symposium, MFCS, Part II*, volume 9235 of *Lecture Notes in Computer Science*, pages 26–37. Springer Verlag, 2015.

3 V. Arvind, J. Köbler, G. Rattan, and O. Verbitsky. Graph Isomorphism, Color Refinement, and Compactness. *Computational Complexity*, 26(3):627–685, 2017.

4 A. Atserias and E. Maneva. Sherali–Adams relaxations and indistinguishability in counting logics. *SIAM J. Comput.*, 42(1):112–137, 2013.

5 A. Atserias and J. Ochremiak. Definable Ellipsoid Method, Sums-of-Squares Proofs, and the Isomorphism Problem. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 66–75, 2018.

6 L. Babai. Graph Isomorphism in Quasipolynomial Time. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC '16)*, pages 684–697, 2016.

7 C. Berkholz and M. Grohe. Limitations of Algebraic Approaches to Graph Isomorphism Testing. In M. M. Halldórsson, K. Iwama, N. Kobayashi, and B. Speckmann, editors, *Proceedings of the 42nd International Colloquium on Automata, Languages and Programming, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 155–166. Springer Verlag, 2015.

8 J. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12:389–410, 1992.

9 P. T. Darga, M. H. Liffiton, K. A. Sakallah, and I. L. Markov. Exploiting structure in symmetry detection for CNF. In *Proceedings of the 41st Design Automation Conference*, pages 530–534. ACM, 2004. doi:10.1145/996566.996712.

10 H. Dell, M. Grohe, and G. Rattan. Lovász Meets Weisfeiler and Leman. In I. Chatzigiannakis, C. Kaklamanis, D. Marx, and D. Sannella, editors, *Proceedings of the 45th International*

Colloquium on Automata, Languages and Programming (Track A), volume 107 of *LIPIcs*, pages 40:1–40:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

11 R. Diestel. *Graph Theory*. Springer Verlag, 4th edition, 2010.

12 M. Fürer. On the Combinatorial Power of the Weisfeiler-Lehman Algorithm. In D. Fotakis, A. Pagourtzis, and V. Th. Paschos, editors, *Proceedings of the 10th International Conference on Algorithms and Complexity*, volume 10236 of *Lecture Notes in Computer Science*, pages 260–271. Springer Verlag, 2017.

13 E. Grädel, M. Grohe, B. Pago, and W. Pakusa. A Finite-Model-Theoretic View on Propositional Proof Complexity, 2018. `arXiv:1802.09377`.

14 M. Grohe. Fixed-point logics on planar graphs. In *Proceedings of the 13th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 6–15, 1998.

15 M. Grohe. Isomorphism testing for embeddable graphs through definability. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, pages 63–72, 2000.

16 M. Grohe. Fixed-Point Definability and Polynomial Time on Graphs with Excluded Minors. *Journal of the ACM*, 59(5), 2012.

17 M. Grohe. *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory*, volume 47 of *Lecture Notes in Logic*. Cambridge University Press, 2017.

18 M. Grohe and S. Kiefer. A Linear Upper Bound on the Weisfeiler-Leman Dimension of Graphs of Bounded Genus, 2019. `arXiv:1904.07216`.

19 M. Grohe and J. Mariño. Definability and descriptive complexity on databases of bounded tree-width. In C. Beeri and P. Buneman, editors, *Proceedings of the 7th International Conference on Database Theory*, volume 1540 of *Lecture Notes in Computer Science*, pages 70–82. Springer-Verlag, 1999.

20 M. Grohe and D. Neuen. Canonisation and Definability for Graphs of Bounded Rank Width, 2019. `arXiv:1901.10330`.

21 M. Grohe and M. Otto. Pebble Games and Linear Equations. *Journal of Symbolic Logic*, 80(3):797–844, 2015.

22 W. L. Hamilton, Z. Ying, and J. Leskovec. Inductive Representation Learning on Large Graphs. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, pages 1025–1035, 2017. URL: `http://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs`.

23 N. Immerman. *Descriptive Complexity*. Springer Verlag, 1999.

24 N. Immerman and E. Lander. Describing graphs: A first-order approach to graph canonization. In A. Selman, editor, *Complexity theory retrospective*, pages 59–81. Springer-Verlag, 1990.

25 T. A. Junttila and P. Kaski. Engineering an Efficient Canonical Labeling Tool for Large and Sparse Graphs. In *Proceedings of the 9th SIAM Workshop on Algorithm Engineering and Experiments*. SIAM, 2007. `doi:10.1137/1.9781611972870.13`.

26 S. Kiefer, I. Ponomarenko, and P. Schweitzer. The Weisfeiler-Leman dimension of planar graphs is at most 3. In *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science*, 2017.

27 S. Kiefer and P. Schweitzer. Upper Bounds on the Quantifier Depth for Graph Differentiation in First Order Logic. In M. Grohe, E. Koskinen, and N. Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 287–296, 2016.

28 S. Kiefer, P. Schweitzer, and E. Selman. Graphs Identified by Logics with Counting. In G. F. Italiano, G. Pighizzini, and D. Sannella, editors, *Proceedings of the 40th International Symposium on Mathematical Foundations of Computer Science 2015 - 40th International Symposium, MFCS, Part I*, volume 9235 of *Lecture Notes in Computer Science*, pages 319–330. Springer Verlag, 2015.

29 T. N. Kipf and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, 2017. URL: `https://openreview.net/forum?id=SJU4ayYgl`.

30 J. Köbler, S. Kuhnert, B. Laubner, and O. Verbitsky. Interval graphs: Canonical representations in logspace. *SIAM Journal on Computing*, 40(5):1292–1315, 2011.

31 A. Krebs and O. Verbitsky. Universal Covers, Color Refinement, and Two-Variable Counting Logic: Lower Bounds for the Depth. In *Proceedings of the 30th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 689–700, 2015.

32 B. Laubner. Capturing Polynomial Time on Interval Graphs. In *Proceedings of the 25th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 199–208, 2010.

33 P. Malkin. Sherali-Adams relaxations of graph isomorphism polytopes. *Discrete Optimization*, 12:73–97, 2014.

34 B. D. McKay. Practical graph isomorphism. *Congressus Numerantium*, 30:45–87, 1981.

35 B. D. McKay and A. Piperno. Practical graph isomorphism, II. *J. Symb. Comput.*, 60:94–112, 2014. doi:10.1016/j.jsc.2013.09.003.

36 B. Mohar and C. Thomassen. *Graphs on Surfaces*. Johns Hopkins University Press, 2001.

37 C. Morris, M. Ritzert, M. Fey, W. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 2019.

38 R. O'Donnell, J. Wright, C. Wu, and Y. Zhou. Hardness of Robust Graph Isomorphism, Lasserre Gaps, and Asymmetry of Random Graphs. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1659–1677, 2014.

39 M. Otto. *Bounded variable logics and counting – A study in finite models*, volume 9 of *Lecture Notes in Logic*. Springer Verlag, 1997.

40 N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-Lehman Graph Kernels. *Journal of Machine Learning Research*, 12:2539–2561, 2011.

41 B. Weisfeiler and A. Leman. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series 2*, 1968. English translation by G. Ryabov available at https://www.iti.zcu.cz/wl2018/pdf/wl_paper_translation.pdf.

Termination of Linear Loops over the Integers

Mehran Hosseini
Department of Computer Science, University of Oxford, UK
mehran.hosseini@cs.ox.ac.uk

Joël Ouaknine
Max Planck Institute for Software Systems, Germany
Department of Computer Science, University of Oxford, UK
joel@mpi-sws.org

James Worrell
Department of Computer Science, University of Oxford, UK
jbw@cs.ox.ac.uk

Abstract

We consider the problem of deciding termination of single-path while loops with integer variables, affine updates, and affine guard conditions. The question is whether such a loop terminates on all integer initial values. This problem is known to be decidable for the subclass of loops whose update matrices are diagonalisable, but the general case has remained open since being conjectured decidable by Tiwari in 2004. In this paper we show decidability of determining termination for arbitrary update matrices, confirming Tiwari's conjecture. For the class of loops considered in this paper, the question of deciding termination on a specific initial value is a longstanding open problem in number theory. The key to our decision procedure is in showing how to circumvent the difficulties inherent in deciding termination on a fixed initial value.

2012 ACM Subject Classification Computing methodologies → Algebraic algorithms; Theory of computation → Logic and verification

Keywords and phrases Program Verification, Loop Termination, Linear Integer Programs, Affine While Loops

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.118

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Funding *Mehran Hosseini*: Mehran Hosseini was supported by ERC grant AVS-ISS (648701).
Joël Ouaknine: Joël Ouaknine was supported by ERC grant AVS-ISS (648701) and by DFG grant 389792660 as part of TRR 248 (see `https://perspicuous-computing.science`).
James Worrell: James Worrell was supported by EPSRC Fellowship EP/N008197/1.

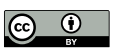
1 Introduction

Termination is a central problem in program verification. In this paper we study termination of *single-path linear loops*, i.e., programs of the form

$$\textit{while } (g_1(\boldsymbol{x}) > 0 \wedge \ldots \wedge g_m(\boldsymbol{x}) > 0) \textit{ do } \boldsymbol{x} := f(\boldsymbol{x})\,,$$

where $g_1, \ldots, g_m : \mathbb{R}^d \to \mathbb{R}$ and $f : \mathbb{R}^d \to \mathbb{R}^d$ are affine maps with integer coefficients. Here the loop body has a single control path that performs a simultaneous affine update of the program variables. Analysis of loops of this form, including acceleration and termination, is an important part of analysing more complex programs (see, e.g., [7, 14, 16]).

For a set $S \subseteq \mathbb{R}^d$, we say that the above loop *terminates on* S if it terminates on all initial vectors $\boldsymbol{x} \in S$. Despite the simplicity of single-path linear loops, the question of deciding termination has proven challenging (and termination already becomes undecidable if the loop



46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 118; pp. 118:1–118:13


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

body consists of a nondeterministic choice between two different linear updates). Tiwari [25] showed that termination of single-path linear loops is decidable over $\mathbb{R}^d$. Subsequently, Braverman [9], using a more refined analysis of the loop components, showed that termination is decidable over $\mathbb{Q}^d$ and noted that termination on $\mathbb{Z}^d$ can be reduced to termination on $\mathbb{Q}^d$ in the homogeneous case, i.e., when the update map f and guards $g_1, \ldots, g_m$ are linear. More recently, Ouaknine, Sousa-Pinto, and Worrell [18] have proven that termination over $\mathbb{Z}^d$ is decidable in the non-homogeneous case under the assumption that the update function f has the form $f(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{a}$ for A a diagonalisable integer matrix. Decidability of termination for non-homogeneous linear loops over $\mathbb{Z}^d$ was conjectured by Tiwari [25, Conjecture 1], but has remained open until now.

In this paper we give a procedure for deciding termination of the general class of single-path linear loops over the integers, i.e., we generalise the result of [18] by lifting the assumption of diagonalisability. Note that for this class of programs, the question of termination on a *given initial value* in $\mathbb{Z}^d$ (as opposed to termination over all of $\mathbb{Z}^d$) is equivalent to the *Positivity Problem* for linear recurrence sequences, i.e., the problem of whether all terms in a given integer linear recurrence sequence are positive. Decidability of the Positivity Problem is a longstanding open problem (going back at least as far as the 1970s [22, 24]), and results in [19] suggest that a solution to the problem will require significant breakthroughs in number theory. However, in considering termination over $\mathbb{Z}^d$ one can benefit from the freedom to choose the initial values of the loop variables. In the present paper we exploit this freedom in order to circumvent the need to solve "hard instances" of the Positivity Problem when deciding termination of linear loops. In particular, we avoid the use of sophisticated Diophantine-approximation techniques, such as the S-units theorem, that were employed in [19]. By eschewing such tools we lose all hope of obtaining an effective characterisation of the set of non-terminating points, as was done in the diagonalisable case in [19], but our methods nevertheless manage to solve the decision problem in the general case.

Among the tools we use are a circle of closely related results in the geometry of numbers, including Khinchine's flatness theorem, Kronecker's theorem on simultaneous Diophantine approximation, and the result of Khachiyan and Porkolab that it is decidable whether a convex semi-algebraic set contains an integer point. In tandem with these, from algebraic number theory, we use a result of Masser that allows to compute all algebraic relations among the eigenvalues of the update matrix of a given loop. Using this last result, we define a semi-algebraic subset of "non-termination candidates" such that the loop is non-terminating if and only if this set contains an integer point.

In this paper we focus on the foundational problem of providing complete methods to solve termination. Much effort has been devoted to scalable and pragmatic methods to prove termination for classes of programs that subsume linear loops. In particular, techniques to prove termination via synthesis of linear ranking functions [4, 5, 8, 10, 11, 20, 21] and their extension, multiphase linear ranking functions [6, 3], have been developed. Many of these techniques have been implemented in software verification tools, such as Microsoft's Terminator [12]. Although these methods are capable of handling non-deterministic linear loops, they can only guarantee termination whenever ranking functions of a certain form exist.

2 Background

2.1 Convexity

The *affine hull* of $S \subseteq \mathbb{R}^d$ is the smallest affine set that contains S, where an affine set is the translation of a vector subspace of $\mathbb{R}^d$. The affine hull of S can be characterised as follows:

$$\mathrm{aff}(S) := \left\{ \sum_{i=1}^{k} \alpha_i \boldsymbol{x}_i \mid k > 0, \boldsymbol{x}_i \in S, \alpha_i \in \mathbb{R}, \sum_{i=1}^{k} \alpha_i = 1 \right\}.$$

The *convex hull* of $S \subseteq \mathbb{R}^d$ is the smallest convex set that contains S. The convex hull of S can be characterised as follows:

$$\mathrm{conv}(S) := \left\{ \sum_{i=1}^{k} \alpha_i \boldsymbol{x}_i \mid k > 0, \boldsymbol{x}_i \in S, \alpha_i \in \mathbb{R}_{\geq 0}, \sum_{i=1}^{k} \alpha_i = 1 \right\}.$$

Clearly $\mathrm{conv}(S) \subseteq \mathrm{aff}(S)$. The *relative interior* of a convex set $S \subseteq \mathbb{R}^d$ is its interior with respect to the restriction of the Euclidean topology to $\mathrm{aff}(S)$. We have the following easy proposition, characterising the relative interior.

▶ **Proposition 1.** *Let $S = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\} \subseteq \mathbb{R}^d$. If $\boldsymbol{u}$ lies in the relative interior of $\mathrm{conv}(S)$ then there exist $\alpha_1, \ldots, \alpha_n > 0$ such that $u = \sum_{i=1}^{n} \alpha_i \boldsymbol{a}_i$ and $\sum_{i=1}^{n} \alpha_i = 1$.*

Proof. Since $\boldsymbol{u}$ lies in the relative interior of $\mathrm{conv}(S)$, for $\varepsilon > 0$ sufficiently small we have that

$$(1 + n\varepsilon)u - \sum_{i=1}^{n} \varepsilon \boldsymbol{a}_i \in \mathrm{conv}(S).$$

For such an ε there exist $\beta_1, \ldots, \beta_n \geq 0$ such that $(1 + n\varepsilon)u - \sum_{i=1}^{n} \varepsilon \boldsymbol{a}_i = \sum_{i=1}^{n} \beta_i \boldsymbol{a}_i$ and $\sum_{i=1}^{n} \beta_i = 1$. But then $\boldsymbol{u} = \sum_{i=1}^{n} \frac{\beta_i + \varepsilon}{1 + n\varepsilon} \boldsymbol{a}_i$. Defining $\alpha_i := \frac{\beta_i + \varepsilon}{1 + n\varepsilon}$ for $i \in \{1, \ldots, n\}$, the proposition is proved. ◀

A *lattice of rank r* in $\mathbb{R}^d$ is a set

$$\Lambda := \{z_1 \boldsymbol{v}_1 + \cdots + z_r \boldsymbol{v}_r \colon z_1, \ldots, z_r \in \mathbb{Z}\},$$

where $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r$ are linearly independent vectors in $\mathbb{R}^d$. Given a convex set $C \subseteq \mathbb{R}^d$, define the *width* of C along a vector $\boldsymbol{u} \in \mathbb{R}^d$ to be

$$\sup\{\boldsymbol{u}^\top (\boldsymbol{x} - \boldsymbol{y}) : \boldsymbol{x}, \boldsymbol{y} \in C\}.$$

Furthermore the *lattice width* of C is the infimum over all non-zero vectors $\boldsymbol{u} \in \Lambda$ of the width of C along $\boldsymbol{u}$.

The following result (see [2, Theorem 7.2.1]) captures the intuition that a convex set that contains no lattice point in its interior must be "thin" in some direction.

▶ **Theorem 2** (Flatness Theorem). *Given a full-rank lattice Λ in $\mathbb{R}^d$ there exists W such that any convex set $C \subseteq \mathbb{R}^d$ of lattice width at least W contains a lattice point.*

Recall that $C \subseteq \mathbb{R}^d$ is said to be *semi-algebraic* if it is definable by a boolean combination of polynomial constraints $p(x_1, \ldots, x_d) > 0$, where $p \in \mathbb{Z}[x_1, \ldots, x_d]$.

▶ **Theorem 3** (Khachiyan and Porkolab [15]). *It is decidable whether a given convex semi-algebraic set $C \subseteq \mathbb{R}^d$ contains an integer point, that is, whether $C \cap \mathbb{Z}^d \neq \emptyset$.*

2.2 Groups of Multiplicative Relations

In this subsection we will introduce some concepts concerning groups of multiplicative relations among algebraic numbers.

Let $\mathbb{T} = \{z \in \mathbb{C} : |z| = 1\}$. We define the s-dimensional torus to be $\mathbb{T}^s$, considered as a group under component-wise multiplication. Given a tuple of algebraic numbers $\boldsymbol{\gamma} = (\gamma_1, \cdots, \gamma_s) \in \mathbb{T}^s$, the orbit $\{\boldsymbol{\gamma}^n : n \in \mathbb{N}\}$ is a subset of $\mathbb{T}^s$. In the following we characterise the topological closure of the orbit as an algebraic subset of $\mathbb{T}^s$.

The *group of multiplicative relations* of $\boldsymbol{\gamma} \in \mathbb{T}^s$ is defined as the following additive subgroup of $\mathbb{Z}^s$:

$$L(\boldsymbol{\gamma}) = \{\boldsymbol{v} \in \mathbb{Z}^s : \boldsymbol{\gamma}^{\boldsymbol{v}} = 1\},$$

where $\boldsymbol{\gamma}^{\boldsymbol{v}}$ is defined to be $\gamma_1^{v_1} \cdots \gamma_s^{v_s}$ for $\boldsymbol{v} \in \mathbb{Z}^s$, that is, exponentiation acts coordinate-wise. Since $L(\boldsymbol{\gamma})$ is a subgroup of $\mathbb{Z}^s$, it is a free Abelian group and hence has a finite basis. The following powerful theorem of Masser [17] gives bounds on the magnitude of the components of such a basis.

▶ **Theorem 4** (Masser). *The free Abelian group $L(\boldsymbol{\gamma})$ has a basis $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_l \in \mathbb{Z}^s$ for which*

$$\max_{1 \le i \le l, 1 \le j \le s} |v_{i,j}| \le (D \log H)^{O(s^2)},$$

where H and D bound respectively the heights and degrees of all the γ_i.

Membership of a tuple $\boldsymbol{v} \in \mathbb{Z}^s$ in $L(\boldsymbol{\gamma})$ can be computed in polynomial space, using a decision procedure for the existential theory of the reals. In combination with Theorem 4, it follows that we can compute a basis for $L(\boldsymbol{\gamma})$ in polynomial space by brute-force search.

Corresponding to $L(\boldsymbol{\gamma})$, we consider the following multiplicative subgroup of $\mathbb{T}^s$:

$$T(\boldsymbol{\gamma}) = \{\boldsymbol{\mu} \in \mathbb{T}^s : \forall \boldsymbol{v} \in L(\boldsymbol{\gamma}), \boldsymbol{\mu}^{\boldsymbol{v}} = 1\}.$$

If $\mathcal{B}$ is a basis of $L(\boldsymbol{\gamma})$, we can equivalently characterise $T(\boldsymbol{\gamma})$ as $\{\boldsymbol{\mu} \in \mathbb{T}^s : \forall \boldsymbol{v} \in \mathcal{B}, \boldsymbol{\mu}^{\boldsymbol{v}} = 1\}$. Crucially, this finitary characterisation allows us to represent $T(\boldsymbol{\gamma})$ as an algebraic set in $\mathbb{T}^s$.

We will use the following classical lemma of Kronecker on simultaneous Diophantine approximation to show that the orbit $\{\boldsymbol{\gamma}^n : n \in \mathbb{N}\}$ is a dense subset of $T(\boldsymbol{\gamma})$.

▶ **Lemma 5.** *Let $\boldsymbol{\theta}, \boldsymbol{\psi} \in \mathbb{R}^s$. Suppose that for all $\boldsymbol{v} \in \mathbb{Z}^s$, if $\boldsymbol{v}^T\boldsymbol{\theta} \in \mathbb{Z}$ then also $\boldsymbol{v}^T\boldsymbol{\psi} \in \mathbb{Z}$, i.e., all integer relations among the coordinates of $\boldsymbol{\theta}$ also hold among those of $\boldsymbol{\psi}$ (modulo $\mathbb{Z}$). Then, for each $\varepsilon > 0$, there exist $\boldsymbol{p} \in \mathbb{Z}^s$ and a non-negative integer n such that*

$$\|n\boldsymbol{\theta} - \boldsymbol{p} - \boldsymbol{\psi}\|_\infty \le \varepsilon.$$

We now arrive at the main result of the section:

▶ **Theorem 6.** *Let $\boldsymbol{\gamma} \in \mathbb{T}^s$. Then the orbit $\{\boldsymbol{\gamma}^k : k \in \mathbb{N}\}$ is a dense subset of $T(\boldsymbol{\gamma})$.*

Proof. Let $\boldsymbol{\theta} \in \mathbb{R}^s$ be such that $\boldsymbol{\gamma} = e^{2\pi i\boldsymbol{\theta}}$ (with exponentiation operating coordinate-wise). Notice that $\boldsymbol{\gamma}^{\boldsymbol{v}} = 1$ if and only if $\boldsymbol{v}^T\boldsymbol{\theta} \in \mathbb{Z}$. If $\boldsymbol{\mu} \in T(\boldsymbol{\gamma})$, we can likewise define $\boldsymbol{\psi} \in \mathbb{R}^s$ to be such that $\boldsymbol{\mu} = e^{2\pi i\boldsymbol{\psi}}$. Then the premises of Kronecker's lemma apply to $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$. Thus, given $\varepsilon > 0$, there exist a non-negative integer k and $\boldsymbol{p} \in \mathbb{Z}^s$ such that $\|k\boldsymbol{\theta} - \boldsymbol{p} - \boldsymbol{\psi}\|_\infty \le \varepsilon$. Whence

$$\|\boldsymbol{\gamma}^k - \boldsymbol{\mu}\|_\infty = \|e^{2\pi i(k\boldsymbol{\theta} - \boldsymbol{p})} - e^{2\pi i\boldsymbol{\psi}}\|_\infty \le \|2\pi(k\boldsymbol{\theta} - \boldsymbol{p} - \boldsymbol{\psi})\|_\infty \le 2\pi\varepsilon. \qquad \blacktriangleleft$$

3 Termination Analysis via Spectral Theory

The general form of a simple linear loop in dimension d is as follows:

$$\textit{while } (g_1(\boldsymbol{x}) > 0 \wedge \ldots \wedge g_m(\boldsymbol{x}) > 0) \textit{ do } \boldsymbol{x} := f(\boldsymbol{x}) \,,$$

where $g_1, \ldots, g_m : \mathbb{R}^d \to \mathbb{R}$ and $f : \mathbb{R}^d \to \mathbb{R}^d$ are affine functions. We assume that f and $g_1, \ldots, g_m$ have integer coefficients, that is, $f(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{a}$ for $A \in \mathbb{Z}^{d\times d}$ and $\boldsymbol{a} \in \mathbb{Z}^d$, and $g_i(\boldsymbol{x}) = \boldsymbol{b}_i^\top \boldsymbol{x} + c_i$ for $\boldsymbol{b}_i \in \mathbb{Z}^d$, $c_i \in \mathbb{Z}$ and $i = 1, \ldots, m$.

Note that

$$\begin{pmatrix} f(\boldsymbol{x}) \\ 1 \end{pmatrix} = \begin{pmatrix} A & \boldsymbol{a} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} \text{ and } g_i(\boldsymbol{x}) = \begin{pmatrix} \boldsymbol{b}_i^\top & c_i \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} . \tag{1}$$

for all $\boldsymbol{x} \in \mathbb{R}^d$. We say that f is *non-degenerate* if no quotient of two distinct eigenvalues of the update matrix $\begin{pmatrix} A & \boldsymbol{a} \\ 0 & 1 \end{pmatrix}$ is a root of unity.

▶ **Proposition 7.** *The termination problem for simple linear loops on integers is reducible to the special case of the problem for non-degenerate update functions.*

Proof. Consider a simple linear loop, as described above, whose update matrix has distinct eigenvalues $\lambda_1, \ldots, \lambda_s$. Let L be the least common multiple of the orders of the roots of unity appearing among the quotients λ_i/λ_j for $i \neq j$. It is known that $L = 2^{O(d\sqrt{\log d})}$ [13, Subsection 1.1.9]. The update matrix corresponding to the affine map $f^L = \underbrace{f \circ \cdots \circ f}_{L}$ has eigenvalues $\lambda_1^L, \ldots, \lambda_s^L$ and hence is non-degenerate. Moreover the original loop terminates if and only if the following loop terminates:

$$\textit{while } \bigwedge_{i=0}^{L-1} \left(g_1(f^i(\boldsymbol{x})) > 0 \wedge \ldots \wedge g_m(f^i(\boldsymbol{x})) > 0\right) \textit{ do } \boldsymbol{x} := f^L(\boldsymbol{x}) \,,$$

This concludes the proof. ◀

In the rest of this section and in the next section we focus on the case of a loop

$$\mathsf{P}: \textit{ while } (g(\boldsymbol{x}) > 0) \textit{ do } x \leftarrow f(\boldsymbol{x}) \textit{ end} \tag{2}$$

with a single guard function $g(\boldsymbol{x}) = \boldsymbol{b}^\top \boldsymbol{x} + c$ and with non-degenerate update function $f(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{a}$, with both maps having integer coefficients. We show that a spectral analysis of the matrix underlying the loop update function suffices to classify almost all initial values of the loop as either terminating or eventually non-terminating. Towards the end of the section we isolate a class of so-called *critical* initial values that are not amenable to this analysis. We show how to deal with such points in Section 4.

With respect to the loop P we say that $\boldsymbol{x} \in \mathbb{R}^d$ is *terminating* if there exists n such that $g(f^n(\boldsymbol{x})) \leq 0$. We say that $\boldsymbol{x}$ is *eventually non-terminating* if the sequence $\langle g(f^n(\boldsymbol{x})) : n \in \mathbb{N}\rangle$ is *ultimately positive*, i.e., there exists N such that for all $n \geq N$ $g(f^n(\boldsymbol{x})) > 0$. Clearly there exists $\boldsymbol{z} \in \mathbb{Z}^d$ that is non-terminating if and only if there exists $\boldsymbol{z} \in \mathbb{Z}^d$ that is eventually non-terminating. Thus we can regard the problem of deciding termination on $\mathbb{Z}^d$ as that of searching for an eventually non-terminating point.

Let $\lambda_1, \ldots, \lambda_s$ be the non-zero eigenvalues of $\begin{pmatrix} A & \boldsymbol{a} \\ 0 & 1 \end{pmatrix}$ and let $k_{\max}$ be the maximum multiplicity over all these eigenvalues.

Define a linear preorder on $I := \{0, \dots, k_{\max} - 1\} \times \{1, \dots, s\}$ by $(i_1, j_1) \preccurlyeq (i_2, j_2)$ if either (i) $|\lambda_{j_1}| < |\lambda_{j_2}|$ or (ii) $|\lambda_{j_1}| = |\lambda_{j_2}|$ and $i_1 \leq i_2$. Write $(i_1, j_1) \prec (i_2, j_2)$ if $(i_1, j_1) \preccurlyeq (i_2, j_2)$ and $(i_2, j_2) \not\preccurlyeq (i_1, j_1)$. Then we have

$$(i_1, j_1) \prec (i_2, j_2) \text{ iff } \lim_{n\to\infty} \frac{\binom{n}{i_1}|\lambda_{j_1}|^n}{\binom{n}{i_2}|\lambda_{j_2}|^n} = 0\,,$$

that is, the preorder $\preccurlyeq$ characterises the asymptotic order of growth in absolute value of the terms $\binom{n}{i}\lambda_j^n$ for $(i, j) \in I$. This preorder moreover induces an equivalence relation $\approx$ on I where $(i_1, j_1) \approx (i_2, j_2)$ iff $(i_1, i_1) \preccurlyeq (i_2, j_2)$ and $(i_2, i_2) \preccurlyeq (i_1, j_1)$.

The following closed-form expression for $g(f^n(\boldsymbol{x}))$ will be the focus of the subsequent development.

▶ **Proposition 8.** *There is a set of affine functions $h_{i,j} : \mathbb{R}^d \to \mathbb{C}$ such that for all $\boldsymbol{x} \in \mathbb{R}^d$ and all $n \geq d$ we have*

$$\boldsymbol{g}(f^n(\boldsymbol{x})) = \sum_{(i,j)\in I} \binom{n}{i} \lambda_j^n \, h_{i,j}(\boldsymbol{x})\,.$$

Proof. By the Jordan-Chevalley decomposition we can write $\begin{pmatrix} A & \boldsymbol{a} \\ 0 & 1 \end{pmatrix} = P^{-1}DP + N$, where D is diagonal, N is nilpotent, P is invertible, $P^{-1}DP$ and N commute, and all matrices have algebraic coefficients. Moreover we can write $D = \lambda_1 D_1 + \cdots + \lambda_s D_s$ for appropriate idempotent diagonal matrices $D_1, \dots, D_s$. Then for all $n \in \mathbb{N}$ with $n \geq d$ we have

$$\begin{aligned}
g(f^n(\boldsymbol{x})) &= (\boldsymbol{b}^\top \; c) \begin{pmatrix} A & \boldsymbol{a} \\ 0 & 1 \end{pmatrix}^n \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} \\
&= (\boldsymbol{b}^\top \; c)(P^{-1}DP + N)^n \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} \\
&= (\boldsymbol{b}^\top \; c) \sum_{i=0}^{n} \binom{n}{i} P^{-1} D^{n-i} P N^i \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} \\
&= (\boldsymbol{b}^\top \; c) \sum_{i=0}^{d} \binom{n}{i} P^{-1} (\lambda_1^{n-i} D_1 + \cdots + \lambda_s^{n-i} D_s) P N^i \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} \quad (\text{since } N^{d+1} = 0) \\
&= \sum_{j=1}^{s} \lambda_j^n \sum_{i=0}^{d} \binom{n}{i} \underbrace{\lambda_j^{-i} (\boldsymbol{b}^\top \; c) P^{-1} D_j P N^i \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix}}_{h_{i,j}(\boldsymbol{x})} \qquad (3) \\
&= \sum_{j=1}^{s} \sum_{i=0}^{d} \binom{n}{i} \lambda_j^n h_{i,j}(\boldsymbol{x})\,,
\end{aligned}$$

where for $(i, j) \in I$ the affine function $h_{i,j}$ is defined in Line (3). Clearly each function $h_{i,j}$ is a complex-valued affine function on $\mathbb{R}^d$ with algebraic coefficients. ◀

Define $\gamma_i = \frac{\lambda_i}{|\lambda_i|}$ for $i = 1, \dots, s$, that is, we obtain the γ_i by normalising the eigenvalues to have length 1. Recall from Section 2.2 the definition of the group $L(\boldsymbol{\gamma})$ of multiplicative relations that hold among $\gamma_1, \dots, \gamma_s$, *viz.*,

$$L(\boldsymbol{\gamma}) = \{(n_1, \dots, n_s) \in \mathbb{Z}^s : \gamma_1^{n_1} \cdots \gamma_s^{n_s} = 1\}\,.$$

Recall also that we have $T(\boldsymbol{\gamma}) \subseteq \mathbb{T}^s$, given by

$$T(\boldsymbol{\gamma}) = \{(\mu_1, \ldots, \mu_s) \in \mathbb{T}^s : \mu_1^{n_1} \cdots \mu_s^{n_s} = 1 \text{ for all } (n_1, \ldots, n_s) \in L(\boldsymbol{\gamma})\}\,.$$

Given an $\approx$-equivalence class $E \subseteq I$, note that for all $(i_1, j_1), (i_2, j_2) \in E$ we have $i_1 = i_2$ and $|\lambda_{j_1}| = |\lambda_{j_2}|$. Thus E is determines a common multiplicity, which we denote i_E, and a set of eigenvalues that all have the same absolute value, which we denote ρ_E.

Given an $\approx$-equivalence class E, define $\Phi_E : \mathbb{R}^d \times T(\boldsymbol{\gamma}) \to \mathbb{R}$ by[1]

$$\Phi_E(\boldsymbol{x}, \boldsymbol{\mu}) = \sum_{(i,j)\in E} h_{i,j}(\boldsymbol{x})\mu_j\,. \tag{4}$$

From the above definition of Φ_E we have

$$\sum_{(i,j)\in E} \binom{n}{i} \lambda_j^n h_{i,j}(\boldsymbol{x}) = \binom{n}{i_E} \rho_E^n \Phi_E(\boldsymbol{x}, \boldsymbol{\gamma}^n)\,. \tag{5}$$

for all $\boldsymbol{x} \in \mathbb{R}^d$ and all $n \in \mathbb{N}$.

We say that an equivalence class E of I is *dominant* for $\boldsymbol{x} \in \mathbb{R}^d$ if E is the equivalence class of the maximal indices (i, j) for which $h_{i,j}(\boldsymbol{x})$ is non-zero. Equivalently, E is dominant for $\boldsymbol{x}$ if E is the maximal equivalence class such that $\Phi_E(\boldsymbol{x}, \cdot)$ is not identically zero on $T(\boldsymbol{\gamma})$. (The equivalence of these two characterisations follows from the linear independence of the functions $\binom{n}{i}\lambda_j^n$ for $(i, j) \in E$.)

The following proposition shows how information about termination of the loop P on an initial value $\boldsymbol{x} \in \mathbb{R}^d$ can be derived from properties of $\Phi_E(\boldsymbol{x}, \cdot)$.

▶ **Proposition 9.** *Consider the loop P in (2). Let $\boldsymbol{x} \in \mathbb{R}^d$ and let E be an $\approx$-equivalence class that is dominant for $\boldsymbol{x}$. Then*

1. *If $\inf_{\boldsymbol{\mu}\in T(\boldsymbol{\gamma})} \Phi_E(\boldsymbol{x}, \boldsymbol{\mu}) > 0$ then $\boldsymbol{x}$ is eventually non-terminating for P.*
2. *If $\inf_{\boldsymbol{\mu}\in T(\boldsymbol{\gamma})} \Phi_E(\boldsymbol{x}, \boldsymbol{\mu}) < 0$ then $\boldsymbol{x}$ is terminating for P.*

Proof. By Proposition 8 and Equation (5) we have that for all $n \geq d$,

$$\begin{aligned} g(f^n(\boldsymbol{x})) &= \sum_{(i,j)\in I} \binom{n}{i} \lambda_j^n h_{i,j}(\boldsymbol{x}) \\ &= \binom{n}{i_E} \rho_E^n \Phi_E(\boldsymbol{x}, \boldsymbol{\gamma}^n) + \sum_{(i,j)\in I\setminus E} \binom{n}{i} \lambda_j^n h_{i,j}(\boldsymbol{x})\,. \end{aligned} \tag{6}$$

Moreover by the dominance of E we have that

$$\lim_{n\to\infty} \frac{\binom{n}{i}|\lambda_j|^n}{\binom{n}{i_E}\rho_E^n} = 0 \tag{7}$$

for all $(i, j) \in I \setminus E$ such that $h_{i,j}(\boldsymbol{x}) \neq 0$.

We first prove Item 1. By assumption, in this case there exists $\varepsilon > 0$ such that $\Phi_E(\boldsymbol{x}, \boldsymbol{\mu}) \geq \varepsilon$ for all $\boldsymbol{\mu} \in T(\boldsymbol{\gamma})$. Together with Equation (7), this shows that the asymptotically dominant term in Equation (6) has positive sign. It follows that $g(f^n(\boldsymbol{x}))$ is positive for n sufficiently large and hence $\boldsymbol{x}$ is eventually non-terminating.

[1] That the function Φ_E is real-valued follows from the fact that if eigenvalues λ_{j_1} and λ_{j_2} are complex conjugates then γ_{j_1} and γ_{j_2} are also complex conjugates, as are $h_{i,j_1}(\boldsymbol{z})$ and $h_{i,j_2}(\boldsymbol{z})$ (see the proof of Proposition 8).

We turn now to Item 2. By assumption there exists $\varepsilon > 0$ and an open subset U of $T(\boldsymbol{\gamma})$ such that $\Phi_E(\boldsymbol{x}, \boldsymbol{\mu}) < -\varepsilon$ for all $\boldsymbol{\mu} \in U$. Moreover by density of $\{\boldsymbol{\gamma}^n : n \in \mathbb{N}\}$ in $T(\boldsymbol{\gamma})$ there exist infinitely many n such that $\boldsymbol{\gamma}^n \in U$. Exactly as in Case 1 we can now use the dominance of E to conclude that $g(f^n(\boldsymbol{x})) < 0$ for sufficiently large n such that $\boldsymbol{\gamma}^n \in U$ and hence $\boldsymbol{x}$ is terminating. ◀

Given $\boldsymbol{z} \in \mathbb{Z}^d$, since $T(\boldsymbol{\gamma})$ is an algebraic subset of $\mathbb{T}^s$, the number $\inf_{\boldsymbol{\mu} \in T(\boldsymbol{\gamma})} \Phi_E(\boldsymbol{z}, \boldsymbol{\mu})$ is algebraic and its sign can be decided. Note however that Proposition 9 does not completely resolve the question of termination with respect to guard g from a given initial value $\boldsymbol{z}$. Indeed, let us define $\boldsymbol{z} \in \mathbb{R}^d$ to be *critical* if $\inf_{\boldsymbol{\mu} \in E} \Phi_E(\boldsymbol{z}, \boldsymbol{\mu}) = 0$, where E is the dominant equivalence class for $\boldsymbol{z}$. Then neither clause in the above proposition suffices to resolve termination of the loop P in (2) on such a $\boldsymbol{z}$. Indeed the question of whether such a point is eventually non-terminating is equivalent to the *Ultimate Positivity Problem* for linear recurrence sequences: a longstanding and notoriously difficult open problem in number theory, only known to be decidable up to order 4 [1, 19]. Fortunately in the setting of deciding loop termination we can sidestep such difficult questions. The following section is devoted to handling critical points. The idea is to show that if there is a critical initial value then there is another initial value that is eventually non-terminating and moreover whose eventual non-termination can be established by Proposition 9.

4 Analysis of Critical Points

In this section we continue to analyse termination of the loop P, as given in (2) in the previous section, and refer to the notation established therein.

4.1 Transition Invariance of Critical Points

Intuitively critical points are those for which it is difficult to determine eventual non-termination. One should therefore expect that if $\boldsymbol{x} \in \mathbb{R}^d$ is critical then $f(\boldsymbol{x})$ should also be critical. This, and more, follows from the following proposition.

▶ **Proposition 10.** *Let $\boldsymbol{x} \in \mathbb{R}^d$ and let $E \subseteq I$ be an equivalence class that is dominant for $\boldsymbol{x}$. Then E is also dominant for $f(\boldsymbol{x})$ and for all $\boldsymbol{\mu} \in T(\boldsymbol{\gamma})$ we have $\Phi_E(f(\boldsymbol{x}), \boldsymbol{\mu}) = \rho_E \, \Phi_E(\boldsymbol{x}, \boldsymbol{\gamma}\boldsymbol{\mu})$, where the product $\boldsymbol{\gamma}\boldsymbol{\mu}$ is defined pointwise.*

Proof. By definition we have $\Phi_E(\boldsymbol{x}, \boldsymbol{\mu}) = \sum_{(i,j) \in E} h_{i,j}(\boldsymbol{x}) \mu_j$, where the $h_{i,j}$ satisfy

$$(\boldsymbol{b}^\top \; c) \begin{pmatrix} A & \boldsymbol{a} \\ 0 & 1 \end{pmatrix}^n \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} = \sum_{(i,j) \in I} h_{i,j}(\boldsymbol{x}) \binom{n}{i} \lambda_j^n \tag{8}$$

for all $n \geq d$. Likewise we have $\Phi_E(f(\boldsymbol{x}), \boldsymbol{\mu}) = \sum_{(i,j) \in S} \widetilde{h}_{i,j}(\boldsymbol{x}) \mu_j$, where the $\widetilde{h}_{i,j}$ satisfy

$$(\boldsymbol{b}^\top \; c) \begin{pmatrix} A & \boldsymbol{a} \\ 0 & 1 \end{pmatrix}^{n+1} \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} = \sum_{(i,j) \in I} \widetilde{h}_{i,j}(\boldsymbol{x}) \binom{n}{i} \lambda_j^n \,. \tag{9}$$

Combining Equations (8) and (9) we have the for all $n \geq d$,

$$\begin{aligned} \sum_{(i,j) \in I} \widetilde{h}_{i,j}(\boldsymbol{x}) \binom{n}{i} \lambda_j^n &= \sum_{(i,j) \in I} h_{i,j}(\boldsymbol{x}) \binom{n+1}{i} \lambda_j^{n+1} \\ &= \sum_{(i,j) \in I} h_{i,j}(\boldsymbol{x}) \left[\binom{n}{i} + \binom{n}{i-1} \right] \lambda_j \lambda_j^n \,. \end{aligned}$$

Now the collection of functions $n \mapsto \binom{n}{i}\lambda_j^n$ for $(i,j) \in I$ is linearly independent in the vector space $\mathbb{C}^{\mathbb{N}}$ (see, e.g., [23, Lemma 9.6]). Equating the coefficients of the functions $\binom{n}{i}\lambda_j^n$ for $(i,j) \in E$ in the above equation we have $\widetilde{h}_{i,j} = \lambda_j h_{i,j} = \rho_E \gamma_j h_{i,j}$ for all $(i,j) \in E$; likewise we have that E is dominant for $f(\boldsymbol{x})$. The proposition follows. ◀

The next lemma shows that the existence of a critical point entails the existence of an eventually non-terminating point.

▶ **Lemma 11.** *If $\boldsymbol{z} \in \mathbb{R}^d$ is critical then there exists a positive integer M such that for all $n \geq M$, all points in the relative interior of* $\mathrm{conv}(\{f^d(\boldsymbol{z}), f^{d+1}(\boldsymbol{z}), \ldots, f^n(\boldsymbol{z})\})$ *are eventually non-terminating.*

Proof. Given an arbitrary $\boldsymbol{\mu} \in T(\boldsymbol{\gamma})$ we claim that there exists $n \geq d$ for which we have $\Phi_E(f^n(\boldsymbol{z}), \boldsymbol{\mu}) > 0$. If this were not the case then for all $n \geq d$ we would have $\Phi_E(f^n(\boldsymbol{z}), \boldsymbol{\mu}) = \Phi_E(\boldsymbol{z}, \boldsymbol{\xi}^n \boldsymbol{\mu}) = 0$. But by Theorem 6, the set $\{\boldsymbol{\xi}^n \boldsymbol{\mu} : n \geq d\}$ is dense in $T(\boldsymbol{\gamma})$ and hence we would have that $\Phi_E(\boldsymbol{z}, \cdot)$ is identically 0 on $T(\boldsymbol{\gamma})$, contradicting the dominance of E.

For each $n \in \mathbb{N}$, the set $C_n = \{\boldsymbol{\mu} \in T(\boldsymbol{\gamma}) : \Phi_E(f^n(\boldsymbol{z}), \boldsymbol{\mu}) > 0\}$ is an open subset of $T(\boldsymbol{\gamma})$. Moreover, by the analysis above, the collection $\{C_n : n \geq d\}$ is an open cover of $T(\boldsymbol{\gamma})$. Thus by compactness of $T(\boldsymbol{\gamma})$ there exists $M \in \mathbb{N}$ such that $C_d, C_{d+1}, \cdots, C_M$ is a finite cover of $T(\boldsymbol{\gamma})$.

By Proposition 1, for all $n \geq M$ and all points $\boldsymbol{x}$ lying in the relative interior of $\mathrm{conv}(\{f^d(\boldsymbol{z}), f^{d+1}(\boldsymbol{z}), \ldots, f^n(\boldsymbol{z})\})$, there exist $\alpha_d, \ldots, \alpha_n > 0$ such that $\sum_{i=d}^n \alpha_i = 1$ and $\boldsymbol{x} = \sum_{i=d}^n \alpha_i f^i(\boldsymbol{z})$. Since Φ_E is an affine map in its first variable, it follows that $\Phi_E(\boldsymbol{x}, \cdot) = \sum_{i=d}^n \alpha_i \Phi_E(f^i(\boldsymbol{z}), \cdot)$ is strictly positive on $T(\boldsymbol{\gamma})$. Hence $\boldsymbol{x}$ is eventually non-terminating by Proposition 9. ◀

4.2 Integer Non-Terminating Points from Critical Points

Lemma 11 shows how to derive the existence of non-terminating points from the existence of a critical point. In this subsection we refine this analysis to derive the existence of *integer* non-terminating points. In particular, fixing an initial value $\boldsymbol{z}_* \in \mathbb{Z}^d$, we show that for n sufficiently large, the set

$$\mathrm{conv}(\{f^d(\boldsymbol{z}_*), f^{d+1}(\boldsymbol{z}_*), \ldots, f^n(\boldsymbol{z}_*)\})$$

contains an integer point in its relative interior.

Define $V := \mathrm{Aff}(\{f^n(\boldsymbol{z}_*) : n \geq d\})$ and let the vector subspace $V_0 \subseteq \mathbb{R}^d$ be the unique translate of V containing the origin. Write d_0 for the dimension of V_0 (equivalently the dimension of V).

▶ **Proposition 12.** *For all non-zero integer vectors $\boldsymbol{v} \in V_0$ the set $\{|\boldsymbol{v}^\top f^n(\boldsymbol{z}_*)| : n \geq d\}$ is unbounded.*

Proof. Consider the sequence $x_n := \boldsymbol{v}^\top f^n(\boldsymbol{z}_*) = \begin{pmatrix} A & \boldsymbol{a} \\ 0 & 1 \end{pmatrix}^n \begin{pmatrix} \boldsymbol{z}_* \\ 1 \end{pmatrix}$. If this sequence were constant then $\boldsymbol{v}$ would be orthogonal to V_0, contradicting the fact that $\boldsymbol{v}$ is non-zero. Since the sequence is non-constant, integer-valued, and satisfies a non-degenerate linear recurrence of order at most $d+1$ (see, e.g., [13, Subsection 1.1.12]), by the Skolem-Mahler-Lech Theorem

we have that $\{|\boldsymbol{v}^\top f^n(\boldsymbol{z}_*)| : n \geq d\}$ is unbounded (see the discussion of growth of linear recurrence in [13, Section 2.2]).[2] ◀

▶ **Proposition 13.** *There exists M such that for all $n \geq M$ the set*

$$\mathrm{conv}(\{f^d(\boldsymbol{z}_*), f^{d+1}(\boldsymbol{z}_*), \ldots, f^n(\boldsymbol{z}_*)\})$$

contains an integer point in its relative interior.

Proof. Since V_0 is spanned by integer vectors, $\Lambda := V_0 \cap \mathbb{Z}^d$ is a lattice of rank d_0 in $\mathbb{R}^d$. Define $C := \mathrm{conv}(\{f^n(\boldsymbol{z}_*) : n \geq d\}) \subseteq V$ and $C_0 := C - f^d(\boldsymbol{z}_*) \subseteq V_0$.

Let $\theta : \mathbb{R}^d \to \mathbb{R}^{d_0}$ be a linear map that takes V_0 bijectively onto $\mathbb{R}^{d_0}$ and whose kernel is the orthogonal complement of V_0. Then $\theta(\Lambda)$ is a lattice in $\mathbb{R}^{d_0}$ of full rank. We claim that the lattice width of $\theta(C_0)$ with respect to $\theta(\Lambda)$ is infinite. Indeed for any non-zero vector $\boldsymbol{v} \in \theta(\Lambda)$ we have

$$\boldsymbol{v}^\top(\theta(f^n(\boldsymbol{z}_*)) - \theta(f^d(\boldsymbol{z}_*))) = (\theta^*\boldsymbol{v})^\top(f^n(\boldsymbol{z}_*) - f^d(\boldsymbol{z}_*))\,, \tag{10}$$

where $\theta^* : \mathbb{R}^{d_0} \to \mathbb{R}^d$ is the adjoint map of θ. But $\theta^*\boldsymbol{v}$ is a non-zero rational vector in V_0 and hence Proposition 12 entails that the absolute value of (10) is unbounded as n runs over $\mathbb{N}$. This proves the claim.

By Theorem 2 we have that $\theta(C_0)$ contains a point of $\theta(\Lambda)$ in its relative interior and hence C_0 contains a point of Λ (necessarily an integer point) in its relative interior. We conclude that C also contains an integer point in its relative interior. ◀

We summarise Sections 3 and 4 with a theorem characterising when a loop with a single guard is terminating.

▶ **Theorem 14.** *The loop P, given in* (2)*, is non-terminating on $\mathbb{Z}^d$ if and only if there exists $\boldsymbol{z} \in \mathbb{Z}^d$ and an $\approx$-equivalence class E such that (i) E is dominating for $\boldsymbol{z}$ and (ii) $\inf_{\boldsymbol{\mu} \in T(\boldsymbol{\gamma})} \Phi_E(\boldsymbol{z}, \boldsymbol{\mu}) \geq 0$.*

Proof. If no such $\boldsymbol{z}$ exists then the loop is terminating by Proposition 9(2). Conversely, if such a $\boldsymbol{z}$ exists then by Lemma 11 and Proposition 13 there exists $\boldsymbol{z}' \in \mathbb{Z}^d$ such that $\inf_{\boldsymbol{\mu} \in T(\boldsymbol{\gamma})} \Phi_E(\boldsymbol{z}', \boldsymbol{\mu}) > 0$ (and with E still dominating for $\boldsymbol{z}'$.) Such a point is eventually non-terminating by Proposition 9(1). ◀

We postpone the question of the effectiveness of the above characterisation until we handle loops with multiple guards, in Section 5.

---

[2] The above argument actually establishes that $\langle x_n : n \in \mathbb{N}\rangle$ diverges to infinity in absolute value. We briefly sketch a more elementary proof of mere unboundedness. If the sequence $\langle x_n : n \in \mathbb{N}\rangle$ were bounded then by van der Waerden's Theorem, for all m it would contain a constant subsequence of the form $x_\ell, x_{\ell+p}, \ldots, x_{\ell+mp}$ for some $\ell, p \geq 1$. In particular, if $m = d$ then since every infinite subsequence $y_n := x_{\ell+pn}$ satisfies a linear recurrence of order at most $d+1$, $\langle x_n : n \in \mathbb{N}\rangle$ would have an infinite constant subsequence $\langle x_{\ell+pn} : n \in \mathbb{N}\rangle$. If $p = 1$ then $\langle x_n : n \in \mathbb{N}\rangle$ is constant and if $p > 1$ then by [23, Lemma 9.11] $\langle x_n : n \in \mathbb{N}\rangle$ is degenerate.

5 Multiple Guards

Now we are ready to present our decision procedure for a general linear loop program

$$\mathsf{Q} : \textit{while } (g_1(\boldsymbol{x}) > 0 \wedge \ldots \wedge g_m(\boldsymbol{x}) > 0) \textit{ do } \boldsymbol{x} := f(\boldsymbol{x}), \tag{11}$$

with multiple guards. Associated to the loop Q we consider m single-guard loops with a common update function:

$$\mathsf{Q}_i : \textit{while } (g_i(\boldsymbol{x}) > 0) \textit{ do } \boldsymbol{x} := f(\boldsymbol{x}),$$

for $i = 1, \ldots, m$. Clearly Q is non-terminating if and only if there exists $\boldsymbol{z} \in \mathbb{Z}^d$ such that each loop Q_i is non-terminating on $\boldsymbol{z}$. As we now explain, we can decide the existence of such a point following the proof of Theorem 14.

Let $\lambda_1, \ldots, \lambda_s$ be the distinct non-zero eigenvalues of the matrix corresponding to the update function f in the loop Q. As before, write $\gamma_j = \frac{\lambda_j}{|\lambda_j|}$ for $j = 1, \ldots, s$. For $i = 1, \ldots, m$, denote by $\Phi_E^{(i)} : \mathbb{R}^d \times T(\boldsymbol{\gamma}) \to \mathbb{R}$ the function associated to loop Q_i and $\approx$-equivalence class E as defined by (4). Given $\approx$-equivalence classes $E_1, \ldots, E_m$, we define $W_{E_1,\ldots,E_m} \subseteq \mathbb{R}^d$ to be the set of $\boldsymbol{\gamma} \in \mathbb{R}^d$ such that the following hold for $i = 1, \ldots, m$:

- E_i is dominant for $\boldsymbol{x}$ in loop Q_i, that is, $\Phi_{E_i}^{(i)}(\boldsymbol{x}, \cdot) \not\equiv 0$ and $\Phi_E^{(i)}(\boldsymbol{x}, \cdot) \equiv 0$ for all $E_i \prec E$.
- $\inf_{\boldsymbol{\mu} \in T(\boldsymbol{\gamma})} \Phi_{E_i}^{(i)}(\boldsymbol{x}, \boldsymbol{\mu}) \geq 0$.

▶ **Proposition 15.** *Loop Q is non-terminating if and only if there exist $\approx$-equivalence classes $E_1, \ldots, E_m$ such that $W_{E_1,\ldots,E_m}$ contains an integer point.*

Proof. Suppose that Q fails to terminate on $\boldsymbol{z} \in \mathbb{Z}^d$. Then each loop Q_i also fails to terminate on $\boldsymbol{z} \in \mathbb{Z}^d$. Thus if E_i is the dominant equivalence class for $\boldsymbol{z}$ in program Q_i, for $i = 1, \ldots, m$, applying Proposition 9(2) we get that $\boldsymbol{z} \in W_{E_1,\ldots,E_m}$.

Conversely, suppose $\boldsymbol{z} \in W_{E_1,\ldots,E_m}$ for some $\approx$-equivalence classes $E_1, \ldots, E_m$. Then, by Lemma 11 and Proposition 13, there is an integer point $\boldsymbol{z}' \in \mathrm{conv}(\{f^n(\boldsymbol{z}) : n \geq d\})$ such that $\inf_{\boldsymbol{\mu} \in T(\boldsymbol{\gamma})} \Phi_{E_i}^{(i)}(\boldsymbol{z}', \boldsymbol{\mu}) > 0$ for $i = 1, \ldots, m$. By Proposition 9(1), each loop Q_i fails to terminate on $\boldsymbol{z}'$ and hence also Q is non-terminating on $\boldsymbol{z}'$. ◀

Proposition 15 leads to the following procedure for deciding termination of a given linear loop Q, as shown in (11).

1. Compute the eigenvalues of the matrix corresponding to the loop update function, as given in (1).
2. Compute the dominance preorder $\preccurlyeq$ among eigenvalues.
3. Compute a basis of the group of multiplicative relations $L(\boldsymbol{\gamma})$.
4. Return "non-terminating" if some set $W_{E_1,\ldots,E_m}$ contains an integer point and otherwise return "terminating".

In terms of effectiveness, Steps 1 and 2 can be accomplished via standard symbolic computations with algebraic numbers. (We refer to [18] for a detailed treatment in a very similar setting.) By Theorem 4, computing a basis of $L(\boldsymbol{\gamma})$ reduces to checking a finite collection of multiplicative relations among algebraic numbers. Given a basis of $L(\boldsymbol{\gamma})$ we can directly obtain representations of each set $W_{E_1,\ldots,E_m}$ as semi-algebraic subsets of $\mathbb{R}^d$. Finally, since $W_{E_1,\ldots,E_m}$ is convex, we can decide the existence of an integer point in each set $W_{E_1,\ldots,E_m}$ using Theorem 3.

We have thus established the main result of the paper:

▶ **Theorem 16.** *There is a procedure to decide termination of single-path linear loops (of the form specified in* (11)*) over the integers.*

References

1 Shaull Almagor, Brynmor Chapman, Mehran Hosseini, Joël Ouaknine, and James Worrell. Effective Divergence Analysis for Linear Recurrence Sequences. In *29th International Conference on Concurrency Theory, CONCUR 2018, September 4-7, 2018, Beijing, China*, pages 42:1–42:15, 2018.

2 Alexander I. Barvinok. Lattice Points and Lattice Polytopes. In *Handbook of Discrete and Computational Geometry, Second Edition.*, pages 153–176. Chapman and Hall/CRC, 2004.

3 Amir M. Ben-Amram, Jesús Doménech, and Samir Genaim. Multiphase-Linear Ranking Functions and their Relation to Recurrent Sets. *CoRR*, abs/1811.07340, 2018. `arXiv:1811.07340`.

4 Amir M. Ben-Amram and Samir Genaim. On the linear ranking problem for integer linear-constraint loops. In *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013*, pages 51–62, 2013.

5 Amir M. Ben-Amram and Samir Genaim. Ranking Functions for Linear-Constraint Loops. *J. ACM*, 61(4):26:1–26:55, 2014.

6 Amir M. Ben-Amram and Samir Genaim. On Multiphase-Linear Ranking Functions. In *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II*, pages 601–620, 2017.

7 Bernard Boigelot. On iterating linear transformations over recognizable sets of integers. *Theor. Comput. Sci.*, 309(1-3):413–468, 2003.

8 Aaron R. Bradley, Zohar Manna, and Henny B. Sipma. Termination Analysis of Integer Linear Loops. In *CONCUR 2005 - Concurrency Theory, 16th International Conference, CONCUR 2005, San Francisco, CA, USA, August 23-26, 2005, Proceedings*, pages 488–502, 2005.

9 Mark Braverman. Termination of Integer Linear Programs. In *Computer Aided Verification, 18th International Conference, CAV 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*, pages 372–385, 2006.

10 Hong Yi Chen, Shaked Flur, and Supratik Mukhopadhyay. Termination proofs for linear simple loops. *STTT*, 17(1):47–57, 2015.

11 Michael Colón and Henny Sipma. Synthesis of Linear Ranking Functions. In *Tools and Algorithms for the Construction and Analysis of Systems, 7th International Conference, TACAS 2001 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2001 Genova, Italy, April 2-6, 2001, Proceedings*, pages 67–81, 2001.

12 Byron Cook, Andreas Podelski, and Andrey Rybalchenko. Termination proofs for systems code. In *Proceedings of the ACM SIGPLAN 2006 Conference on Programming Language Design and Implementation, Ottawa, Ontario, Canada, June 11-14, 2006*, pages 415–426, 2006.

13 Graham Everest, Alfred J. van der Poorten, Igor E. Shparlinski, and Thomas Ward. *Recurrence Sequences*, volume 104 of *Mathematical surveys and monographs*. American Mathematical Society, 2003.

14 Bertrand Jeannet, Peter Schrammel, and Sriram Sankaranarayanan. Abstract acceleration of general linear loops. In *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 529–540. ACM, 2014.

15 Leonid Khachiyan and Lorant Porkolab. Computing Integral Points in Convex Semi-algebraic Sets. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 162–171, 1997.

16 Zachary Kincaid, Jason Breck, John Cyphert, and Thomas W. Reps. Closed forms for numerical loops. *PACMPL*, 3(POPL):55:1–55:29, 2019.

17 David W Masser. Linear relations on algebraic groups. *New Advances in Transcendence Theory*, pages 248–262, 1988.

18 Joël Ouaknine, João Sousa Pinto, and James Worrell. On Termination of Integer Linear Loops. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 957–969, 2015.

19 Joël Ouaknine and James Worrell. Positivity Problems for Low-Order Linear Recurrence Sequences. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 366–379, 2014.

20 Andreas Podelski and Andrey Rybalchenko. A Complete Method for the Synthesis of Linear Ranking Functions. In *Verification, Model Checking, and Abstract Interpretation, 5th International Conference, VMCAI 2004, Venice, Italy, January 11-13, 2004, Proceedings*, pages 239–251, 2004.

21 Andreas Podelski and Andrey Rybalchenko. Transition Invariants. In *19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings*, pages 32–41, 2004.

22 G. Rozenberg and A. Salomaa. *Cornerstones of Undecidability*. Prentice Hall, 1994.

23 Arto Salomaa and Matti Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Texts and Monographs in Computer Science. Springer, 1978.

24 M. Soittola. On D0L Synthesis Problem. In A. Lindenmayer and G. Rozenberg, editors, *Automata, Languages, Development*. North-Holland, 1976.

25 Ashish Tiwari. Termination of Linear Programs. In *Computer Aided Verification, 16th International Conference, CAV 2004, Boston, MA, USA, July 13-17, 2004, Proceedings*, pages 70–82, 2004.

Büchi Objectives in Countable MDPs

Stefan Kiefer
University of Oxford, UK

Richard Mayr
University of Edinburgh, UK

Mahsa Shirmohammadi
CNRS, Paris, France
IRIF, Paris, France

Patrick Totzke
University of Liverpool, UK

Abstract

We study countably infinite Markov decision processes with Büchi objectives, which ask to visit a given subset F of states infinitely often. A question left open by T.P. Hill in 1979 [10] is whether there always exist ε-optimal Markov strategies, i.e., strategies that base decisions only on the current state and the number of steps taken so far. We provide a negative answer to this question by constructing a non-trivial counterexample. On the other hand, we show that Markov strategies with only 1 bit of extra memory are sufficient.

2012 ACM Subject Classification Theory of computation → Random walks and Markov chains; Mathematics of computing → Probability and statistics

Keywords and phrases Markov decision processes

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.119

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version The full version of this paper is available at `https://arxiv.org/abs/1904.11573`.

Funding *Stefan Kiefer*: Supported by a Royal Society University Research Fellowship.
Richard Mayr: Supported by EPSRC grant EP/M027651/1.
Mahsa Shirmohammadi: Supported by PEPS JCJC grant AAPS.

Acknowledgements The authors thank anonymous reviewers for their helpful comments.

1 Introduction

Background. Markov decision processes (MDPs) are a standard model for dynamic systems that exhibit both stochastic and controlled behavior [16]. MDPs play a prominent role in numerous domains, including artificial intelligence and machine learning [19, 18], control theory [4, 1], operations research and finance [5, 17], and formal verification [9, 2]. In an MDP, the system starts in the initial state and makes a sequence of transitions between states. Depending on the type of the current state, either the controller gets to choose an enabled transition (or a distribution over transitions), or the next transition is chosen randomly according to a defined distribution. By fixing a strategy for the controller, one obtains a probability space of runs of the MDP. The goal of the controller is to optimize the expected value of some objective function on the runs.

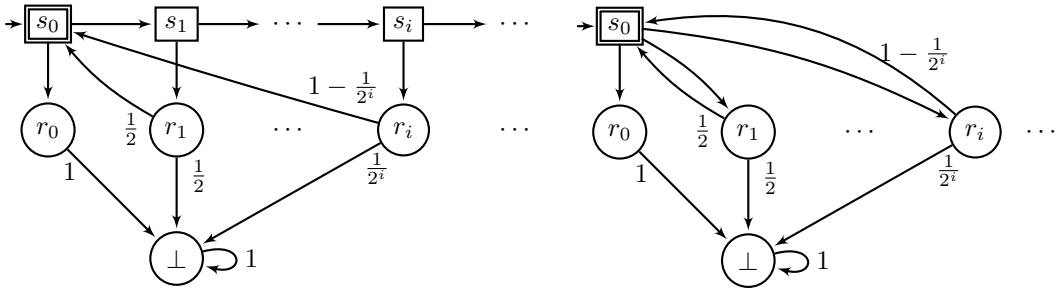
The type of strategy needed for an optimal (resp. ε-optimal) strategy for some objective is also called the *strategy complexity* of the objective. There are different types of strategies, depending on whether one can take the whole history of the run into account (history-dependent; (H)), or whether one is limited to a finite amount of memory (finite memory;

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 119; pp. 119:1–119:14

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

(F)) or whether decisions are based only on the current state (memoryless; (M)). Moreover, the strategy type depends on whether the controller can randomize (R) or is limited to deterministic choices (D). The simplest type MD refers to memoryless deterministic strategies. *Markov strategies* are strategies that base their decisions only on the current state and the number of steps in the history or the run. Thus they do use infinite memory, but only in a very restricted form by maintaining an unbounded step-counter. For finite MDPs, there exist optimal MD-strategies for many (but not all) objectives [6, 7, 8, 16], but the picture is more complex for countably infinite MDPs [13, 15, 16].

We study here so-called *Goal* objectives defined via a subset of goal states F: In the basic Goal objective (also called the *Reachability* objective) one simply wants to reach the set F. In the *Büchi* objective one wants to visit the set F infinitely often. For finite MDPs there exist optimal MD-strategies for both these objectives [7, 16]. For countably infinite MDPs, optimal strategies (where they exist) and ε-optimal strategies for Reachability can be chosen MD [15, 16]. Similarly, optimal strategies for Büchi (where they exist) can be chosen MD [13]. However, ε-optimal strategies for Büchi require infinite memory (cannot be chosen FR); cf. [13, 14].



(a) Finitely branching, but infinitely many controlled states.

(b) Infinitely branching, but just one controlled state.

Figure 1 Two MDPs where ε-optimal strategies for Büchi require infinite memory. Let $F = \{s_0\}$ be the set of goal states. Here and throughout the paper we indicate goal states by double borders, and controlled states as rectangles.

▶ **Example 1.** Consider the MDPs in Figure 1. Every finite memory (FR) strategy will only attain probability 0 for Büchi in these examples [13]. However, there exists an ε-optimal Markov strategy for every $\varepsilon > 0$: At the i-th time that state s_0 is visited, pick the successor state r_{i+k} where k is some sufficiently large number depending on ε, e.g., $k = \lceil \log_2(1/\varepsilon) \rceil$. For example (b) this can easily be done with a step-counter since s_0 is visited for the i-th time in step $2(i-1)$ unless the system has reached the state $\perp$. For example (a), under this strategy, state s_0 is visited for the i-th time in step $\sum_{j=1}^{i-1}(k+j+1)$ unless the system has reached the state $\perp$. ◀

▶ **Example 2.** Consider the MDP from Figure 2, taken from [11, Example 4.2]. Every FR-strategy attains only probability 0 of Büchi. Moreover, the strategy that, in state s_0, subsequently picks $r_1, r_2, \ldots$ also attains probability 0, unlike in Example 1. But a different infinite-memory strategy achieves a positive probability. Indeed, let σ be the strategy that, in s_0, picks 2^1 times r_1 and then 2^2 times r_2 and $\ldots 2^i$ times r_i etc. This strategy σ achieves a positive probability of Büchi. (In more detail, σ achieves a positive probability of not falling in a losing sink $\perp$, and in almost all of the remaining runs it visits a goal state infinitely often.) Note that σ is a Markov strategy. ◀

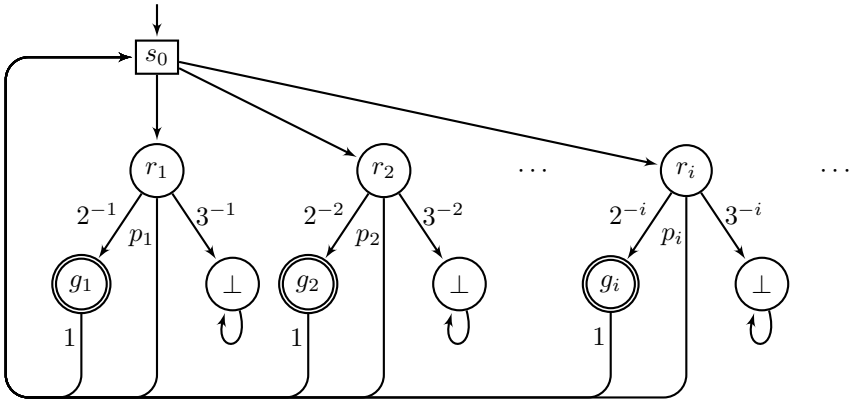


Figure 2 An MDP where ε-optimal strategies for Büchi require infinite memory. The transition probability p_i stands for $1 - 2^{-i} - 3^{-i}$. The state s_0 is the only controlled state.

The open problem. While the MDPs in Examples 1 and 2 require infinite memory, Markov strategies suffice for them. Such examples led to the question whether there always exists a family of ε-optimal Markov strategies for Büchi in all countably infinite MDPs.

A partial answer was given by Hill [10] (Proposition 5.1), who showed that ε-optimal Markov strategies for Büchi exist in the special case where the MDP contains only a *finite* number of controlled states. This result applies to the MDPs from Example 2 and Figure 1b), but not directly to the one in Figure 1a).

The question for general MDPs was stated as an open problem in [10] (p.158, l.4) and mentioned again in [11] (Q1 in Section 5).

Our contributions. We provide a negative answer to the open problem. We construct a non-trivial example of a countable acyclic and finitely branching MDP and prove that no ε-optimal Markov strategies for Büchi exist for it (for any $\varepsilon < 1$). In combination with the example from Figure 1, this shows that for general MDPs neither finite memory (FR) nor Markov strategies are sufficient.

Secondly, we provide an upper bound on the strategy complexity of Büchi. We show that for *acyclic* countable MDPs there always exist ε-optimal strategies that are deterministic and use only one bit of memory. Since every MDP can be transformed into an acyclic one by encoding a step-counter into the states, it follows that general countable MDPs have ε-optimal strategies for Büchi that are deterministic and use only a step-counter plus one extra bit of memory. Thus Markov strategies are almost, but not quite, sufficient. Table 1 summarizes these results.

Table 1 Existence of various types of ε-optimal strategies for the Büchi objective, for several classes of MDPs. New results are in boldface.

ε-optimal strategy for Büchi	MD	1-bit D	FR	Markov	Markov+1 bit D
Finite MDP	✓	✓	✓	✓	✓
MDP w. finitely many controlled states	×	×	×	✓	✓
Acyclic MDP	**×**	**✓**	**✓**	**×**	**✓**
General MDP	×	×	×	**×**	**✓**

2 Preliminaries

A *probability distribution* over a countable set S is a function $f : S \to [0,1]$ with $\sum_{s \in S} f(s) = 1$. We write $\mathcal{D}(S)$ for the set of all probability distributions over S.

For a set S we write S^* (resp. S^ω) for the set of all finite (resp. infinite) sequences of elements in S. We use slightly generalized regular expressions for sets of sequences, e.g., if $s_0 \in S$ we may write $s_0 S^\omega$ for the set of infinite sequences starting with s_0.

Markov decision processes. A *Markov decision process* (MDP) $\mathcal{M} = (S, S_\Box, S_\bigcirc, \longrightarrow, P)$ consists of a countable set S of *states*, which is partitioned into a set $S_\Box$ of *controlled states* and a set $S_\bigcirc$ of *random states*, a *transition relation* $\longrightarrow \subseteq S \times S$, and a *probability function* $P : S_\bigcirc \to \mathcal{D}(S)$. We write $s \longrightarrow s'$ if $(s, s') \in \longrightarrow$, and refer to s' as a *successor* of s. We assume that every state has at least one successor. The probability function P assigns to each random state $s \in S_\bigcirc$ a probability distribution $P(s)$ over its (non-empty) set of successor states. A *sink in* $\mathcal{M}$ is a subset $T \subseteq S$ closed under the $\longrightarrow$ relation, that is, $s \in T$ and $s \longrightarrow s'$ implies that $s' \in T$.

An MDP is *acyclic* if the underlying directed graph $(S, \longrightarrow)$ is acyclic, i.e., there is no directed cycle. It is *finitely branching* if every state has finitely many successors and *infinitely branching* otherwise. An MDP without controlled states ($S_\Box = \emptyset$) is called a *Markov chain*.

Strategies and Probability Measures. A *run* ρ is an infinite sequence $s_0 s_1 \cdots$ of states such that $s_i \longrightarrow s_{i+1}$ for all $i \in \mathbb{N}$; write $\rho(i) \stackrel{\text{def}}{=} s_i$ for the i-th state along ρ. A *partial run* is a finite prefix of a run. We say that (partial) run ρ *visits* s if $s = \rho(i)$ for some i, and that ρ starts in s if $s = \rho(0)$.

A *strategy* is a function $\sigma : S^* S_\Box \to \mathcal{D}(S)$ that assigns to partial runs $\rho s \in S^* S_\Box$ a distribution over the successors $\{s' \in S \mid s \longrightarrow s'\}$. The set of all strategies in $\mathcal{M}$ is denoted by $\Sigma_\mathcal{M}$ (we omit the subscript and write Σ if $\mathcal{M}$ is clear from the context). A (partial) run $s_0 s_1 \cdots$ is induced by strategy σ if for all i either $s_i \in S_\Box$ and $\sigma(s_0 s_1 \cdots s_i)(s_{i+1}) > 0$, or $s_i \in S_\bigcirc$ and $P(s_i)(s_{i+1}) > 0$.

An MDP $\mathcal{M} = (S, S_\Box, S_\bigcirc, \longrightarrow, P)$, an initial state $s_0 \in S$, and a strategy σ induce a probability space in which the outcomes are runs starting in s_0 and with measure $\mathcal{P}_{\mathcal{M},s_0,\sigma}$ defined as follows. It is first defined on *cylinders* $s_0 s_1 \dots s_n S^\omega$, where $s_1, \dots, s_n \in S$: if $s_0 s_1 \dots s_n$ is not a partial run induced by σ then $\mathcal{P}_{\mathcal{M},s_0,\sigma}(s_0 s_1 \dots s_n S^\omega) \stackrel{\text{def}}{=} 0$. Otherwise, $\mathcal{P}_{\mathcal{M},s_0,\sigma}(s_0 s_1 \dots s_n S^\omega) \stackrel{\text{def}}{=} \prod_{i=0}^{n-1} \bar{\sigma}(s_0 s_1 \dots s_i)(s_{i+1})$, where $\bar{\sigma}$ is the map that extends σ by $\bar{\sigma}(ws) = P(s)$ for all $ws \in S^* S_\bigcirc$. By Carathéodory's theorem [3], this extends uniquely to a probability measure $\mathcal{P}_{\mathcal{M},s_0,\sigma}$ on the Borel σ-algebra $\mathcal{F}$ of subsets of $s_0 S^\omega$. Elements of $\mathcal{F}$, i.e., measurable sets of runs, are called *events* or *objectives* here. For $X \in \mathcal{F}$ we will write $\overline{X} \stackrel{\text{def}}{=} s_0 S^\omega \setminus X \in \mathcal{F}$ for its complement and $\mathcal{E}_{\mathcal{M},s_0,\sigma}$ for the expectation w.r.t. $\mathcal{P}_{\mathcal{M},s_0,\sigma}$. We drop the indices wherever possible without introducing ambiguity.

Strategy Classes. Strategies are in general *randomized* (R) in the sense that they take values in $\mathcal{D}(S)$. A strategy σ is *deterministic* (D) if $\sigma(\rho)$ is a Dirac distribution for all runs $\rho \in S^* S_\Box$.

We formalize the amount of *memory* needed to implement strategies. Let M be a countable set of memory modes, and let $\tau : \mathsf{M} \times S \to \mathcal{D}(\mathsf{M} \times S)$ be a function that meets the following two conditions: for all modes $\mathsf{m} \in \mathsf{M}$,

- for all controlled states $s \in S_\Box$, the distribution $\tau(\mathsf{m}, s)$ is over $\mathsf{M} \times \{s' \mid s \longrightarrow s'\}$.
- for all random states $s \in S_\bigcirc$, we have $\sum_{\mathsf{m}' \in \mathsf{M}} \tau(\mathsf{m}, s)(\mathsf{m}', s') = P(s)(s')$.

The function τ together with an initial memory mode m_0 induce a strategy $\sigma_\tau : S^*S_\Box \to \mathcal{D}(S)$ as follows. Consider the Markov chain with the set $\mathsf{M} \times S$ of states and the probability function τ. A sequence $\rho = s_0 \cdots s_i$ corresponds to a set $H(\rho) = \{(\mathsf{m}_0, s_0) \cdots (\mathsf{m}_i, s_i) \mid \mathsf{m}_0, \ldots, \mathsf{m}_i \in \mathsf{M}\}$ of runs in this Markov chain. Each $\rho s \in s_0 S^* S_\Box$ induces a probability distribution $\mu_{\rho s} \in \mathcal{D}(\mathsf{M})$, the probability of being in state (m, s) conditioned on having taken some partial run from $H(\rho s)$. We define σ_τ such that $\sigma_\tau(\rho s)(s') = \sum_{\mathsf{m},\mathsf{m}' \in \mathsf{M}} \mu_{\rho s}(\mathsf{m}) \tau(\mathsf{m}, s)(\mathsf{m}', s')$ for all $\rho s \in S^* S_\Box$ and all $s' \in S$.

We say that a strategy σ can be *implemented* with memory M if there exist $\mathsf{m}_0 \in \mathsf{M}$ and τ such that $\sigma_\tau = \sigma$. We define certain classes of strategies:

- A strategy σ is *finite memory* (F) if there exists a finite memory M implementing σ.
- A strategy σ is *memoryless* (M) (also called *positional*) if it can be implemented with a memory of size 1. We may view M-strategies as functions $\sigma : S_\Box \to \mathcal{D}(S)$.
- A strategy σ is *1-bit* if it can be implemented with a memory of size 2. Such a strategy is then determined by a function $\tau : \{0,1\} \times S \to \mathcal{D}(\{0,1\} \times S)$. Intuitively τ uses one bit of memory to capture two different modes.
- A strategy σ is *Markov* if it can be implemented with the natural numbers $\mathbb{N}$ as the memory, and a function τ such that the distribution $\tau(\mathsf{m}, s)$ is over $\{\mathsf{m}+1\} \times S$ for all $\mathsf{m} \in \mathsf{M}$ and $s \in S$. Intuitively, such a strategy depends only on the the current state and the number of steps taken so far, i.e., it has access to a step-counter. We view Markov strategies as functions $\sigma : \mathbb{N} \times S_\Box \to \mathcal{D}(S)$. Note that such a strategy is generally not finite memory.
- A strategy σ is *1-bit Markov* if it can be implemented with $\mathbb{N} \times \{0,1\}$ as the memory, and a function τ such that the distribution $\tau(n, b, s)$ is over $\{n+1\} \times \{0,1\} \times S$ for all $(n, b) \in \mathsf{M}$ and $s \in S$. We view such strategies as functions $\sigma : \mathbb{N} \times \{0,1\} \times S_\Box \to \mathcal{D}(\{0,1\} \times S)$.

Payoffs, Values, Optimality. We are interested in strategies to maximize the expectation of a given measurable *payoff* function $f : S^\omega \to \mathbb{R}$, a random variable that assigns a real value to every run. The *value* of state s (w.r.t. f) is the supremum of expected values of f over all strategies:

$$\mathtt{val}_{\mathcal{M},f}(s) \stackrel{\text{def}}{=} \sup_{\sigma \in \Sigma} \mathcal{E}_{\mathcal{M},s,\sigma}(f),$$

For $\varepsilon \geq 0$ and $s \in S$, we say that a strategy σ is *ε-optimal* iff $\mathcal{E}_{\mathcal{M},s,\sigma}(f) \geq \mathtt{val}_{\mathcal{M},f}(s) - \varepsilon$ and *uniformly* ε-optimal iff this holds for every $s \in S$. A (uniformly) 0-optimal strategy is simply called (uniformly) *optimal*.

In this paper, we will need two types of payoff functions. The first is the *total reward*, a random variable given as $f(\rho) \stackrel{\text{def}}{=} \sum_{t=0}^{\infty} r(\rho(t))$, where $r : S \to \mathbb{R}$ is some given *reward* function. A useful fact [16, Theorem 7.1.9] is that if S is finite and the range of r is bounded then there exist optimal strategies (for total reward) which are memoryless and deterministic.

The second type of payoff functions we consider are those with range $\{0,1\}$. Each such payoff function f uniquely identifies an objective (set of runs) φ by viewing f as the characteristic function of φ, i.e., $f(\rho) = 1$ if $\rho \in \varphi$ and 0 otherwise. Then $\mathcal{E}_{\mathcal{M},s,\sigma}(f) = \mathcal{P}_{\mathcal{M},s,\sigma}(\varphi)$. We call this the *probability of achieving* φ (using strategy σ starting from the state s) and simply write $\mathtt{val}_{\mathcal{M},\varphi}(s) = \mathtt{val}_{\mathcal{M},f}(s) = \sup_{\sigma \in \Sigma} \mathcal{P}_{\mathcal{M},s,\sigma}(\varphi)$.

Our main focus are *reachability* (sometimes also called *goal*) and *Büchi* objectives, which are determined by a set of states $F \subseteq S$ and defined as follows. Let us slightly abuse notation and identify F with its characteristic function, i.e., $F(s) = 1$ if $s \in F$.

- The *reachability* objective is to visit F at least once during a run. The corresponding payoff is $f(\rho) \stackrel{\text{def}}{=} \max_{t \in \mathbb{N}} \rho(t)$, and we define $\mathtt{Goal}(F) \stackrel{\text{def}}{=} \{\rho \in S^\omega \mid \max_{t \in \mathbb{N}} F(\rho(t)) = 1\}$;
- The *Büchi* objective is to visit F infinitely often. The corresponding payoff function is $f(\rho) \stackrel{\text{def}}{=} \limsup_{t \to \infty} F(\rho(t))$, and we let $\mathtt{Büchi}(F) \stackrel{\text{def}}{=} \{\rho \in S^\omega \mid \limsup_{t \to \infty} F(\rho(t)) = 1\}$.

3 The Lower Bound

In this section we solve Hill's problem ([10] and [11, Q1]) by exhibiting an MDP where the initial state has value 1 w.r.t. the Büchi objective, but every Markov strategy achieves this objective with probability 0. As explained in the introduction, it follows that in acyclic MDPs, ε-optimal MR-strategies are not guaranteed to exist. In fact, in the following theorem we prove the latter fact first, and subsequently generalize it to solve Hill's problem.

▶ **Theorem 3.** *There exists an acyclic MDP $\mathcal{M}$, a state s_0 and a set of states F such that*

1. *for every Markov strategy σ, we have $\mathcal{P}_{\mathcal{M},s_0,\sigma}(\texttt{Büchi}(F)) = 0$, and*
2. $\texttt{val}_{\texttt{Büchi}(F)}(s_0) = 1$ *and for every $\varepsilon > 0$ there exists a deterministic 1-bit strategy σ_ε s.t. $\mathcal{P}_{\mathcal{M},s_0,\sigma_\varepsilon}(\texttt{Büchi}(F)) \geq 1 - \varepsilon$.*

In the remainder of this section we provide a proof sketch. The full proof is in [12].

Proof sketch for Theorem 3. Our construction is based on an infinite MDP $\mathcal{M}$ that consists of a chain of height-n trees, T^n, for $n \in \mathbb{N} = \{1, 2, \ldots\}$. Figure 3 depicts its initial segment T^1, T^2, T^3. Each such tree is "rooted" at a brown state on the top level, with a transition incoming from a blue state. We make use of some conventions that simplify the presentation and the analysis. In Figure 3, the different colors of the states highlight the structure of the MDP; the colors are also indicated by letters in the states: blue (L), brown (B), yellow (Y), red (R), green (G), white (W). The start state, s_0, is the blue state in the top-left corner. The controlled states are exactly the yellow states. The goal set F consists of the green states at the bottom. Two transitions emanate from each red state: a black (right) transition and a red (left) transition, both leading to the same (brown or green) state.

We consider the strengthened Büchi objective that asks to see F infinitely often and moreover that *no red transition* is taken. This corresponds exactly to the normal Büchi objective if we redirect every red transition to an infinite (losing) chain of non-green states (not depicted in Figure 3).

We first argue that no MR-strategy achieves a positive probability of that objective. Then we show that the MDP $\mathcal{M}$ can be modified so that no Markov strategy achieves a positive probability.

Intuition behind the construction of $\mathcal{M}$. The objective, say φ, of visiting infinitely many green states and no red transition creates tension between trying to visit green states and avoiding too many red states (the latter states incur a risk of taking a red transition). In the proof we need to show that no memoryless strategy strikes a good balance between these competing goals. On the one end of the spectrum, an MR-strategy might always choose the upward transition in the yellow states (which are the only controlled states). But such a strategy never visits any green state, thus clearly violates φ. On the other end of the spectrum lies the "greedy" MR-strategy, which always chooses the downward transition in the yellow states, in order to visit as many green states as possible. Indeed, under this strategy, let u_n denote the probability that, starting in the top-left brown state of T^n, no green state is visited in T^n. By induction (given in [12]) one can show that there is $u < 1$ such that $u_n \leq u$ holds for all n. Considering the probability of the transitions emanating from the blue states (at the top), the expected overall number of visited green states is at least $\sum_{n=1}^{\infty} \frac{1}{n}(1 - u_n) \geq (1 - u) \sum_{n=1}^{\infty} \frac{1}{n} = \infty$. It is not hard to strengthen this statement so that the greedy strategy almost surely visits infinitely many green states. So the greedy strategy satisfies one part of φ, but it does so at the expense of visiting many red states. Red states though are associated with a risk of taking a red transition, and it follows from the proof in [12] that the greedy strategy almost surely ends up taking at least one (and indeed infinitely many) red transition(s).

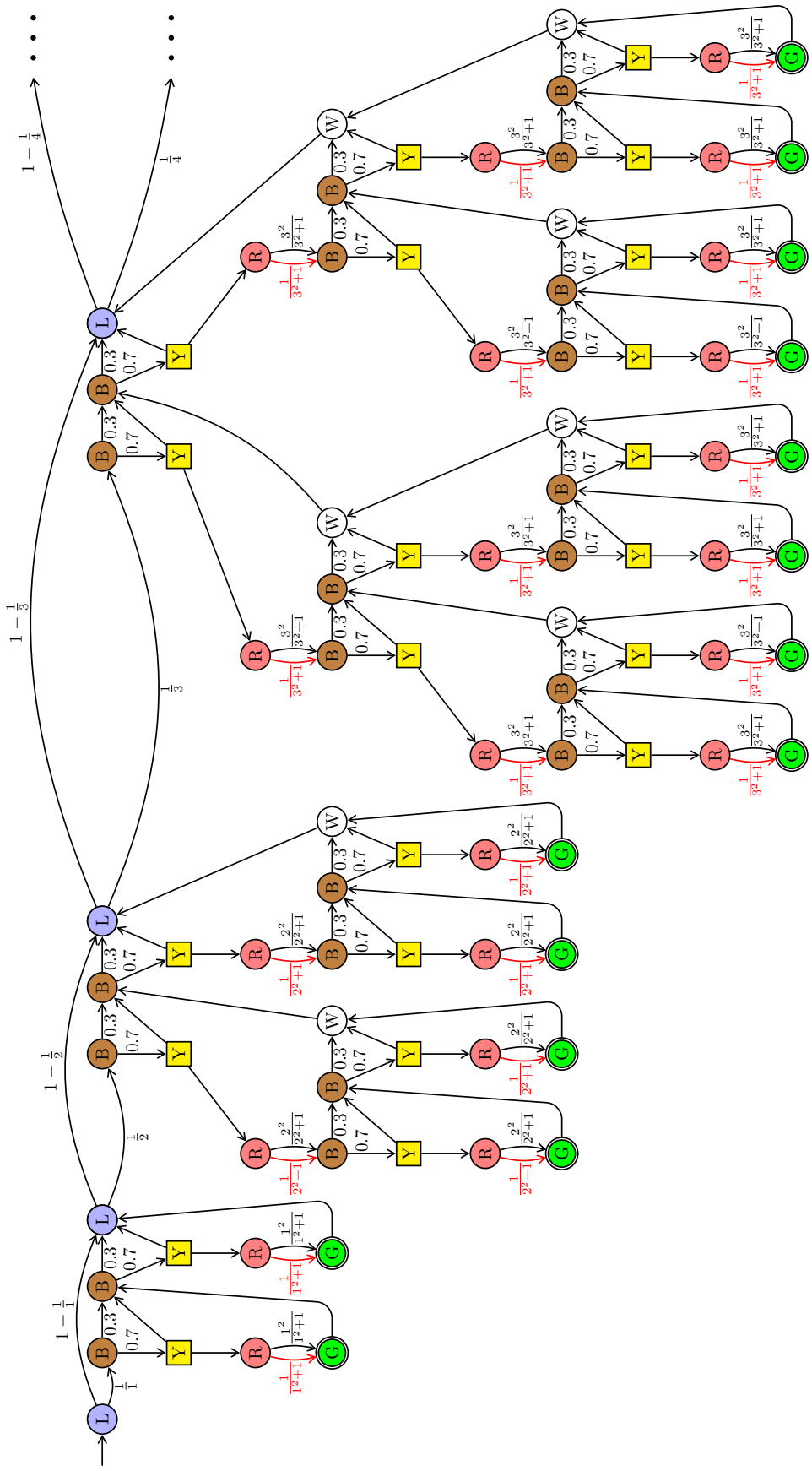


Figure 3 For this acyclic MDP $\mathcal{M}$ there are ε-optimal deterministic 1-bit strategies for the Büchi objective $\texttt{Büchi}(T)$ where T contains exactly all green states. No MR-strategy achieves even a positive probability.

Good 1-bit strategies. The two competing goals discussed in the previous paragraph can be balanced using a deterministic 1-bit strategy, which we describe in the following. This strategy, σ_1, sets its bit to 0 whenever a blue state (at the top) is entered. While the bit is 0, in each tree T^n it maximizes the probability of visiting a green state by choosing the downward transition in the yellow states, thus accepting a certain risk of taking a red transition. However, if and when a green state in T^n is visited, the bit is set to 1, and for the remaining sojourn in T^n the strategy σ_1 chooses the upward transitions in the yellow states, thus avoiding any risk of a red transition in the remainder of T^n. Although σ_1 appears to visit fewer green states than the aforementioned "greedy" MR-strategy, σ_1 still visits infinitely many green states almost surely. This is because for each tree T^n, the two strategies have the same probability of visiting at least one green state in T^n. The strategy σ_1 can be improved, for each $\varepsilon > 0$, to achieve φ with probability at least $1 - \varepsilon$, by fixing the bit to 1 in the first k trees $T^1, \ldots, T^k$, for a k that depends on ε. Thus the first k trees are virtually skipped, eliminating the risk of taking any red transition there. In this way one can make the risk of taking a red transition arbitrarily small, while still visiting infinitely many green states with probability 1.

No good MR-strategies. We need to show that not only the extreme MR-strategies described above are inadequate but that every MR-strategy achieves φ with probability 0. To this end, for each tree T^n, define two probabilities:

- t_n (for "total success"): the probability that, starting in the top-left brown state of T^n, at least one green state but no red transition is visited in T^n;
- d_n (for "death"): the probability that, starting in the top-left brown state of T^n, a red transition is visited in T^n.

A very technical proof shows that $d_n \geq 0.008 \cdot t_n$ holds for all n, and this key inequality captures the inability of *any* MR-strategy to strike an adequate balance between the mentioned competing goals. Indeed, one can show that for an MR-strategy to have a positive probability of not visiting any red transition, the series $\sum_{n=1}^{\infty} \frac{1}{n} \cdot d_n$ needs to converge; but to have a positive probability of visiting infinitely many green states, the series $\sum_{n=1}^{\infty} \frac{1}{n} \cdot t_n$ needs to diverge (in both cases, the factor $\frac{1}{n}$ is the probability of visiting the top-left brown node of T^n). By the inequality above, this is impossible.

No good Markov strategies. For the proof of Theorem 3, we also need to show that all Markov strategies achieve probability 0. To this end, we modify the MDP $\mathcal{M}$ so that for each state, all paths from the initial state s_0 to s have the same length. This can be achieved by replacing some transitions in $\mathcal{M}$ by longer chains consisting of non-green states. This modification does not change the fact that MR-strategies achieve probability 0. But since in the new MDP each state can only be visited at a certain time, which is known a priori, a step-counter does not help. Hence all Markov strategies, like MR-strategies, achieve φ with probability 0. ◀

Theorem 3 answers Hill's question negatively. By combining the MDP from Theorem 3 with one of the MDPs from Figure 1 (by adding a new initial random state that branches to the MDPs with probability $\frac{1}{2}$ each), one can even construct a single MDP whose value w.r.t. $\texttt{Büchi}(F)$ is 1, but every FR- and every Markov strategy achieves probability 0.

A slight modification of the example above yields a lower bound on the memory requirements for the almost-sure parity objective. Recall that the parity objective is defined on systems whose states are labeled by a finite set of colors $C \stackrel{\text{def}}{=} \{1, 2, \ldots, max\} \subseteq \mathbb{N}$, where a run is in $\texttt{Parity}(C)$ iff the highest color that is seen infinitely often in the run is even.

▶ **Corollary 4.** *There exist an acyclic MDP $\mathcal{M}'$ with colors $\{1,2,3\}$ and a state s_0 such that*
1. *for every Markov strategy σ, we have $\mathcal{P}_{\mathcal{M}',s_0,\sigma}(\texttt{Parity}(\{1,2,3\})) = 0$, and*
2. *there exists a deterministic 1-bit strategy σ' such that $\mathcal{P}_{\mathcal{M}',s_0,\sigma'}(\texttt{Parity}(\{1,2,3\})) = 1$.*

Proof. We obtain $\mathcal{M}'$ by modifying the MDP $\mathcal{M}$ from Theorem 3 as follows. Label all green states in F by color 2 and the rest by color 1. Then modify each red transition to go to its target via a fresh state labeled by color 3. Clearly $\mathcal{M}'$ is still acyclic and labeled by colors $\{1,2,3\}$.

From the proof of Theorem 3 (1), under every Markov strategy in $\mathcal{M}$ a.s. seeing infinitely many green states (in F) implies seeing infinitely many red transitions. So in $\mathcal{M}'$ every Markov strategy σ a.s. either sees color 2 only finitely often or color 3 infinitely often, thus $\mathcal{P}_{\mathcal{M}',s_0,\sigma}(\texttt{Parity}(\{1,2,3\})) = 0$.

From the proof of Theorem 3 (2), there is a deterministic 1-bit strategy σ in $\mathcal{M}$ that attains probability $\geq 1/2$ for $\texttt{Büchi}(F)$ without taking any red transition and otherwise a.s. takes a red transition. This property of σ holds not only when starting from s_0 but from every other state as well. We obtain σ' in $\mathcal{M}'$ by continuing to play σ even after red transitions have been taken. Under σ' the probability of going through infinitely many red transitions (and seeing color 3) is $\leq (1/2)^\infty = 0$, and the probability of seeing infinitely many states in F (with color 2) is 1. Thus $\mathcal{P}_{\mathcal{M}',s_0,\sigma'}(\texttt{Parity}(\{1,2,3\})) = 1$. ◀

4 The Upper Bound

We show that acyclic MDPs admit ε-optimal deterministic 1-bit strategies for Büchi.

We start by giving some intuition why 1 bit of memory is needed and how it is used. A step $s' \longrightarrow s''$ from some controlled state s' is *value-decreasing* iff $\texttt{val}(s'') < \texttt{val}(s')$. While an optimal strategy can never tolerate any value-decreasing step, an ε-optimal strategy might have to take value-decreasing steps infinitely often. The trick is to keep the collective value-loss sufficiently small ($\leq \varepsilon$), while satisfying the other requirements of the objective. So the strategy needs to play "ever better" (i.e., tolerate only smaller and smaller value decreases) along a run. In general this requires infinite memory, since one might re-visit the same state infinitely often and needs to choose a different transition from it every time; cf. Figure 1. However, in an acyclic MDP, with high probability, the distance to the initial state increases with the number of steps taken. Thus one can partition the state space into separate regions, depending on the distance from the initial state, and fix an acceptable rate of value-decrease for each region. Just limiting the collective value-loss is not sufficient for Büchi, one also needs to make progress and visit the set of goal states F at least once in each region. The problem is that some runs might linger in some region too long, and visit F *many times*, but see too many value-decreasing steps at the rate of this region. Therefore, as soon as one has visited F in some region, one should try to get to the next outer region (further away from the initial state) where the rate of value-loss is smaller. Thus one needs 1 bit of memory to record whether one has already seen F in this region. (Remember that the same state can be reached by different runs with different histories.) Just 1 bit suffices, because the probability of returning to a previous inner region (and misinterpreting the bit) can be made arbitrarily small, since the MDP is acyclic.

▶ **Theorem 5.** *For every acyclic countable MDP $\mathcal{M}$, finite set of initial states I, set of states F and $\varepsilon > 0$, there exists a deterministic 1-bit strategy for $\texttt{Büchi}(F)$ that is ε-optimal from every $s \in I$.*

Proof. Let $\mathcal{M} = (S, S_\Box, S_\bigcirc, \longrightarrow, P)$ be an acyclic MDP, $I \subseteq S$ a finite set of initial states and $F \subseteq S$ a set of goal states and $\varphi \stackrel{\text{def}}{=} \texttt{Büchi}(F)$ denote the Büchi objective w.r.t. F. We prove the claim for finitely branching $\mathcal{M}$ first and transfer the result to general MDPs at the end. For every $\varepsilon > 0$ and every $s \in I$ there exists an ε-optimal strategy σ_s such that

$$\mathcal{P}_{\mathcal{M},s,\sigma_s}(\varphi) \geq \texttt{val}_{\mathcal{M},\varphi}(s) - \varepsilon. \tag{1}$$

However, the strategies σ_s might differ from each other and might use randomization and a large (or even infinite) amount of memory. We will construct a single deterministic strategy σ' that uses only 1 bit of memory such that $\forall_{s\in I}\, \mathcal{P}_{\mathcal{M},s,\sigma'}(\varphi) \geq \texttt{val}_{\mathcal{M},\varphi}(s) - 2\varepsilon$. This proves the claim as ε can be chosen arbitrarily small.

In order to construct σ', we first observe the behavior of the finitely many σ_s for $s \in I$ on an infinite, increasing sequence of finite subsets of S. Based on this, we define a second stronger objective φ' with

$$\varphi' \subseteq \varphi, \tag{2}$$

and show that all σ_s attain at least $\texttt{val}_{\mathcal{M},\varphi}(s) - 2\varepsilon$ w.r.t. φ', i.e.,

$$\forall_{s\in I}\, \mathcal{P}_{\mathcal{M},s,\sigma_s}(\varphi') \geq \texttt{val}_{\mathcal{M},\varphi}(s) - 2\varepsilon. \tag{3}$$

We construct σ' as a deterministic 1-bit *optimal* strategy w.r.t. φ' from all $s \in I$ and obtain

$$\begin{aligned}
\mathcal{P}_{\mathcal{M},s,\sigma'}(\varphi) &\geq \mathcal{P}_{\mathcal{M},s,\sigma'}(\varphi') && \text{by (2)} \\
&\geq \mathcal{P}_{\mathcal{M},s,\sigma_s}(\varphi') && \text{by optimality of } \sigma' \text{ for } \varphi' \\
&\geq \texttt{val}_{\mathcal{M},\varphi}(s) - 2\varepsilon && \text{by (3).}
\end{aligned}$$

Informal outline: Behavior of σ_s, objective φ' and properties (2) and (3). For the formal proof see [12].

Let $\mathsf{bubble}_k(I)$ be the set of states that can be reached from some initial state in I within at most k steps. Since I is finite and $\mathcal{M}$ is finitely branching, $\mathsf{bubble}_k(I)$ is finite for every k.

We define a sequence of sufficiently large and increasing numbers k_i and l_i with $k_i < l_i < k_{i+1}$ for $i \in \mathbb{N}$ and finite sets $K_i \stackrel{\text{def}}{=} \mathsf{bubble}_{k_i}(I)$ and $L_i \stackrel{\text{def}}{=} \mathsf{bubble}_{l_i}(I)$. Every run from a $s \in I$ according to σ_s must eventually leave each of these finite sets, because $\mathcal{M}$ is acyclic. Moreover, we choose these numbers so that once a run has left L_i it is very unlikely to return to K_i. Let $F_i \stackrel{\text{def}}{=} F \cap K_i \setminus L_{i-1}$. Runs according to σ_s are very likely to follow a particular pattern. Let $R_1 \stackrel{\text{def}}{=} (K_1 \setminus F_1)^* F_1$, $R_2 \stackrel{\text{def}}{=} (K_2 \setminus F_2)^* F_2$ and $R_{i+1} \stackrel{\text{def}}{=} (K_{i+1} \setminus (F_{i+1} \cup K_{i-1}))^* F_{i+1}$ for $i \geq 2$. We show that

$$\forall_{s\in I}\, \mathcal{P}_{\mathcal{M},s,\sigma_s}(\varphi \cap \overline{R_1 R_2 \dots R_{i+1}(S \setminus K_i)^\omega}) \quad \leq \quad \varepsilon \tag{4}$$

We now define the Borel objectives $R_{\leq i} \stackrel{\text{def}}{=} R_1 R_2 \dots R_i S^\omega$ and $\varphi' \stackrel{\text{def}}{=} \bigcap_{i\in\mathbb{N}} R_{\leq i}$. Since $F_i \cap F_k = \emptyset$ for $i \neq k$ and φ' implies a visit to the set F_i for all $i \in \mathbb{N}$, we have $\varphi' \subseteq \varphi$ and obtain (2). Using (4), we show that $\forall_{s\in I}\, \mathcal{P}_{\mathcal{M},s,\sigma_s}(\varphi') \geq \texttt{val}_{\mathcal{M},\varphi}(s) - 2\varepsilon$ and thus obtain (3).

Definition of the 1-bit strategy σ'. We now define a deterministic 1-bit strategy σ' that is optimal for objective φ' from every $s \in I$. First we define certain "suffix" objectives of φ'. Recall that $R_i = (K_i \setminus (F_i \cup K_{i-2}))^* F_i$. Let $R_{i,j} \stackrel{\text{def}}{=} R_i R_{i+1} \dots R_j S^\omega$ and $R_{\geq i} \stackrel{\text{def}}{=} \bigcap_{j\geq i} R_{i,j}$. Consider the objectives $R_{\geq i+1}$ for runs that start in states $s' \in F_i$. For every state $s' \in F_i$ we consider its value w.r.t. the objective $R_{\geq i+1}$, i.e., $\texttt{val}_{\mathcal{M},R_{\geq i+1}}(s') \stackrel{\text{def}}{=} \sup_{\hat{\sigma}} \mathcal{P}_{\mathcal{M},s',\hat{\sigma}}(R_{\geq i+1})$.

For every $i \geq 1$ we consider the finite subspace $K_i \setminus K_{i-2}$. In particular, it contains the sets F_{i-1} and F_i. We define a bounded total reward objective B_i for runs starting in F_{i-1} as follows. Runs that exit the subspace (either by leaving K_i or by visiting K_{i-2}) before visiting F_i get reward 0. All other runs must visit F_i eventually (since $\mathcal{M}$ is acyclic and the subspace is finite). When some run reaches the set F_i *for the first time* in some state s' then this run gets the reward of $\mathtt{val}_{\mathcal{M},R_{\geq i+1}}(s')$. Using [16, Theorem 7.1.9], we show that there exists a uniform optimal MD-strategy σ_i for B_i on $K_i \setminus K_{i-2}$ in $\mathcal{M}$.

We now define σ' by combining different MD-strategies σ_i, depending on the current state and on the value of the 1-bit memory. The intuition is that the strategy σ' has two modes: normal-mode and next-mode. In a state $s' \in K_i \setminus K_{i-1}$, if the memory is $i \pmod 2$ then the strategy is in normal-mode and plays towards reaching F_i. Otherwise, the strategy is in next-mode and plays towards reaching F_{i+1}.

Initially σ' starts in a state $s \in I$ with the 1-bit memory set to 1. We define the behavior of σ' in a state $s' \in K_i \setminus K_{i-1}$ for every $i \geq 1$. If the 1-bit memory is $i \pmod 2$ and $s' \notin F_i$ then σ' plays like σ_i. (Intuitively, one plays towards F_i, since one has not yet visited it.) If the 1-bit memory is $i \pmod 2$ and $s' \in F_i$ then the 1-bit memory is set to $(i+1) \pmod 2$, and σ' plays like σ_{i+1}. (Intuitively, one records the fact that one has already seen F_i and then targets the next set F_{i+1}.) If the 1-bit memory is $(i+1) \pmod 2$ then σ' plays like σ_{i+1}. (Intuitively, one plays towards F_{i+1}, since one has already visited F_i.)

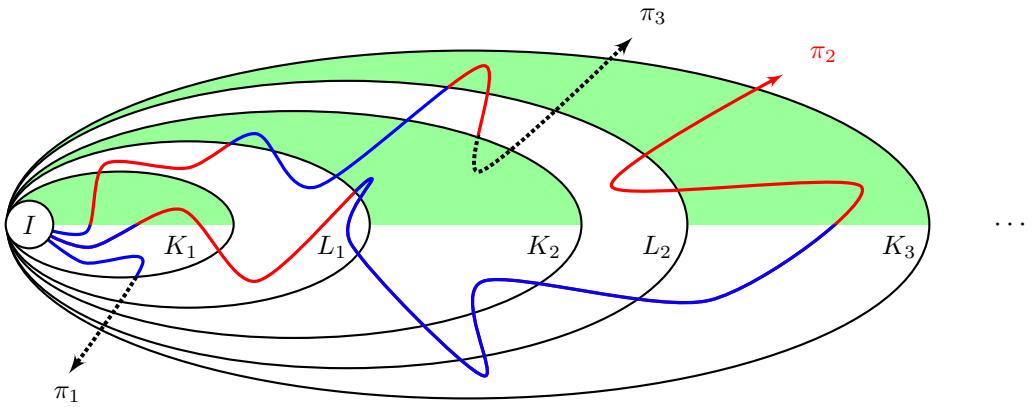


Figure 4 Memory updates along runs π_1, π_2, π_3, drawn in blue while the memory-bit is one and in red while the bit is zero. The green region in K_1 is F_1, and for all $i \geq 2$, the green region in $K_i \setminus L_{i-1}$ is F_i. Both π_1 and π_3 violate φ' and are drawn as dotted lines once they do.

Observe that if a run according to σ' exits some set K_i (and thus enters $K_{i+1} \setminus K_i$) with the bit still set to $i \pmod 2$ (normal-mode) then this run has not visited F_i and thus does not satisfy the objective φ'. (Or the same has happened earlier for some $j < i$, in which case also the objective φ' is violated.) An example is the run π_1 in Figure 4. However, if a run according to σ' exits some set K_i (and thus enters $K_{i+1} \setminus K_i$) with the bit set to $(i+1) \pmod 2$ (thus σ_{i+1} in next-mode) then in the new set $K_{i'} \setminus K_{i'-1}$ with $i' = i+1$ the bit is set to $i' \pmod 2$ and σ' continues to play like σ_{i+1} in normal-mode. Even if this run returns (temporarily) to K_i (but not to K_{i-1}) the strategy σ' continues to play like σ_{i+1} in next-mode. An example is the run π_2 in Figure 4. Finally, if a run returns to K_{i-1} after having visited F_i then it fails the objective φ', e.g., run π_3 in Figure 4.

The 1-bit strategy σ' is optimal for φ' from every $s \in I$. Let $s \in I$ be arbitrary. For a given run from s, let $\mathsf{firstin}(F_i)$ be the first state s' in F_i that is visited (if any). We define a bounded reward objective B'_i for runs starting at s as follows. Every run that does not satisfy the objective $R_{\leq i}$ gets assigned reward 0. Otherwise, consider a run from s that satisfies $R_{\leq i}$. When this run reaches the set F_i for the first time in some state s' then this run gets a reward of $\mathtt{val}_{\mathcal{M},R_{\geq i+1}}(s')$. Note that this reward is ≤ 1.

We show that for all $i \in \mathbb{N}$

$$\mathtt{val}_{\mathcal{M},\varphi'}(s) = \mathtt{val}_{\mathcal{M},B'_i}(s) \tag{5}$$

Towards the $\geq$ inequality, let $\hat{\sigma}$ be an $\hat{\varepsilon}$-optimal strategy for B'_i from s. We define the strategy $\hat{\sigma}'$ to play like $\hat{\sigma}$ until a state $s' \in F_i$ is reached and then to switch to some $\hat{\varepsilon}$-optimal strategy for objective $R_{\geq i+1}$ from s'. Every run from s that satisfies φ' can be split into parts, before and after the first visit to the set F_i, i.e., $\varphi' = \{w_1 s' w_2 \mid w_1 s' \in R_{\leq i}, s' \in F_i, s' w_2 \in R_{\geq i+1}\}$. Therefore we obtain that $\mathcal{P}_{\mathcal{M},s,\hat{\sigma}'}(\varphi') \geq \mathcal{E}_{\mathcal{M},s,\hat{\sigma}}(B'_i) - \hat{\varepsilon} \geq \mathtt{val}_{\mathcal{M},B'_i}(s) - 2\hat{\varepsilon}$. Since this holds for every $\hat{\varepsilon} > 0$, we obtain $\mathtt{val}_{\mathcal{M},\varphi'}(s) \geq \mathtt{val}_{\mathcal{M},B'_i}(s)$.

Towards the $\leq$ inequality, let $\hat{\sigma}$ be any strategy for φ' from s. We have $\mathcal{P}_{\mathcal{M},s,\hat{\sigma}}(\varphi') \leq \sum_{s' \in F_i} \mathcal{P}_{\mathcal{M},s,\hat{\sigma}}(R_{\leq i} \cap \mathsf{firstin}(F_i) = s') \cdot \mathtt{val}_{\mathcal{M},R_{\geq i+1}}(s') = \mathcal{E}_{\mathcal{M},s,\hat{\sigma}}(B'_i)$. Thus $\mathtt{val}_{\mathcal{M},\varphi'}(s) \leq \mathtt{val}_{\mathcal{M},B'_i}(s)$. Together we obtain (5).

For all $i \in \mathbb{N}$ and every state $s' \in F_i$ we show that

$$\mathtt{val}_{\mathcal{M},R_{\geq i+1}}(s') = \mathtt{val}_{\mathcal{M},B_{i+1}}(s') \tag{6}$$

Towards the $\geq$ inequality, let $\hat{\sigma}$ be an $\hat{\varepsilon}$-optimal strategy for B_{i+1} from $s' \in F_i$. We define the strategy $\hat{\sigma}'$ to play like $\hat{\sigma}$ until a state $s'' \in F_{i+1}$ is reached and then to switch to some $\hat{\varepsilon}$-optimal strategy for objective $R_{\geq i+2}$ from s''. We have that $\mathcal{P}_{\mathcal{M},s',\hat{\sigma}'}(R_{\geq i+1}) \geq \mathcal{E}_{\mathcal{M},s',\hat{\sigma}}(B_{i+1}) - \hat{\varepsilon} \geq \mathtt{val}_{\mathcal{M},B_{i+1}}(s) - 2\hat{\varepsilon}$. Since this holds for every $\hat{\varepsilon} > 0$, we obtain $\mathtt{val}_{\mathcal{M},R_{\geq i+1}}(s') \geq \mathtt{val}_{\mathcal{M},B_{i+1}}(s')$.

Towards the $\leq$ inequality, let $\hat{\sigma}$ be any strategy for $R_{\geq i+1}$ from $s' \in F_i$. We have

$$\begin{aligned}\mathcal{P}_{\mathcal{M},s',\hat{\sigma}}(R_{\geq i+1}) &\leq \sum_{s'' \in F_{i+1}} \mathcal{P}_{\mathcal{M},s',\hat{\sigma}}(R_{i+1} S^\omega \cap \mathsf{firstin}(F_{i+1}) = s'') \cdot \mathtt{val}_{\mathcal{M},R_{\geq i+2}}(s'') \\ &= \mathcal{E}_{\mathcal{M},s',\hat{\sigma}}(B_{i+1}).\end{aligned}$$

Thus $\mathtt{val}_{\mathcal{M},R_{\geq i+1}}(s') \leq \mathtt{val}_{\mathcal{M},B_{i+1}}(s')$. Together we obtain (6).

We show, by induction on i, that σ' is optimal for B'_i for all $i \in \mathbb{N}$ from start state s, i.e.,

$$\mathcal{E}_{\mathcal{M},s,\sigma'}(B'_i) = \mathtt{val}_{\mathcal{M},B'_i}(s) \tag{7}$$

In the base case of $i = 1$ we have that $B'_1 = B_1$. The strategy σ' plays σ_1 until reaching F_1, which is optimal for objective B_1 and thus optimal for B'_1. For the induction step we assume (IH) that σ' is optimal for B'_i.

$$\begin{aligned}
\mathtt{val}_{\mathcal{M},B'_{i+1}}(s) &= \mathtt{val}_{\mathcal{M},B'_i}(s) && \text{by (5)} \\
&= \mathcal{E}_{\mathcal{M},s,\sigma'}(B'_i) && \text{by (IH)} \\
&= \sum_{s' \in F_i} \mathcal{P}_{\mathcal{M},s,\sigma'}(R_{\leq i} \cap \mathsf{firstin}(F_i) = s') \cdot \mathtt{val}_{\mathcal{M},R_{\geq i+1}}(s') && \text{by def. of } B'_i \\
&= \sum_{s' \in F_i} \mathcal{P}_{\mathcal{M},s,\sigma'}(R_{\leq i} \cap \mathsf{firstin}(F_i) = s') \cdot \mathtt{val}_{\mathcal{M},B_{i+1}}(s') && \text{by (6)} \\
&= \sum_{s' \in F_i} \mathcal{P}_{\mathcal{M},s,\sigma'}(R_{\leq i} \cap \mathsf{firstin}(F_i) = s') \cdot \mathcal{E}_{\mathcal{M},s',\sigma_{i+1}}(B_{i+1}) && \text{opt. of } \sigma_{i+1} \text{ for } B_{i+1} \\
&= \mathcal{E}_{\mathcal{M},s,\sigma'}(B'_{i+1}) && \text{by def. of } \sigma' \text{ and } B'_{i+1}
\end{aligned}$$

So σ' attains the value $\mathtt{val}_{\mathcal{M},B'_{i+1}}(s)$ of the objective B'_{i+1} from s and is optimal. Thus (7).

Now we show that σ' performs well on the objectives $R_{\leq i}$ for all $i \in \mathbb{N}$.

$$\mathcal{P}_{\mathcal{M},s,\sigma'}(R_{\leq i}) \geq \mathtt{val}_{\mathcal{M},\varphi'}(s) \tag{8}$$

We have

$$\begin{aligned}
\mathcal{P}_{\mathcal{M},s,\sigma'}(R_{\leq i}) &\geq \mathcal{E}_{\mathcal{M},s,\sigma'}(B'_i) && \text{since } B'_i \text{ gives rewards 0 for runs} \notin R_{\leq i} \text{ and} \leq 1 \text{ otherwise}\\
&= \mathtt{val}_{\mathcal{M},B'_i}(s) && \text{by (7)}\\
&= \mathtt{val}_{\mathcal{M},\varphi'}(s) && \text{by (5)}
\end{aligned}$$

So we get (8). Now we are ready to prove the optimality of σ' for φ' from s.

$$\begin{aligned}
\mathcal{P}_{\mathcal{M},s,\sigma'}(\varphi') &= \mathcal{P}_{\mathcal{M},s,\sigma'}(\cap_{i\in\mathbb{N}} R_{\leq i}) && \text{by def. of } \varphi'\\
&= \lim_{i\to\infty} \mathcal{P}_{\mathcal{M},s,\sigma'}(R_{\leq i}) && \text{by continuity of measures from above}\\
&\geq \lim_{i\to\infty} \mathtt{val}_{\mathcal{M},\varphi'}(s) && \text{by (8)}\\
&= \mathtt{val}_{\mathcal{M},\varphi'}(s)
\end{aligned}$$

From finitely to infinitely branching MDPs. Encode infinite branching into finite branching like in Figure 1, apply the above result to obtain a 1-bit strategy for the finitely branching version, and then transform this strategy back into a 1-bit strategy for the original MDP. ◀

Now we show our upper bound on the strategy complexity of Büchi for general MDPs.

▶ **Theorem 6.** *For every countable MDP $\mathcal{M}$, finite set of initial states I, set of states F and $\varepsilon > 0$, there exists a deterministic 1-bit Markov strategy for* $\mathtt{Büchi}(F)$ *that is ε-optimal from every $s \in I$.*

Proof. Encode a step-counter into the states to obtain an acyclic MDP, apply Theorem 5 to obtain an ε-optimal deterministic 1-bit strategy for it, and then transform this strategy back into an ε-optimal deterministic 1-bit Markov strategy in the original MDP. ◀

References

1. Pieter Abbeel and Andrew Y. Ng. Learning first-order Markov models for control. In *Advances in Neural Information Processing Systems 17*, pages 1–8. MIT Press, 2004.
2. Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.
3. P. Billingsley. *Probability and Measure*. Wiley, 1995. Third Edition.
4. Vincent D. Blondel and John N. Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274, 2000.
5. Nicole Bäuerle and Ulrich Rieder. *Markov Decision Processes with Applications to Finance*. Springer-Verlag Berlin Heidelberg, 2011.
6. K. Chatterjee, L. de Alfaro, and T. Henzinger. Trading memory for randomness. In *Annual Conference on Quantitative Evaluation of Systems*, pages 206–217. IEEE Computer Society Press, 2004.
7. K. Chatterjee and T. Henzinger. A survey of stochastic ω-regular games. *Journal of Computer and System Sciences*, 78(2):394–413, 2012.
8. K. Chatterjee, M. Jurdziński, and T. Henzinger. Quantitative Stochastic Parity Games. In *Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 121–130. Society for Industrial and Applied Mathematics, 2004.
9. Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors. *Handbook of Model Checking*. Springer, 2018.

10 Theodore Preston Hill. On the Existence of Good Markov strategies. *Transactions of the American Mathematical Society*, 247:157–176, 1979.

11 Theodore Preston Hill. Goal Problems in Gambling Theory. *Revista de Matemática: Teoría y Aplicaciones*, 6(2):125–132, 1999.

12 Stefan Kiefer, Richard Mayr, Mahsa Shirmohammadi, and Patrick Totzke. Büchi Objectives in Countable MDPs. Technical report, arxiv.org, 2019. `arXiv:1904.11573`.

13 Stefan Kiefer, Richard Mayr, Mahsa Shirmohammadi, and Dominik Wojtczak. Parity Objectives in Countable MDPs. In *Annual IEEE Symposium on Logic in Computer Science*, 2017.

14 J. Krčál. Determinacy and Optimal Strategies in Stochastic Games. Master's thesis, Masaryk University, School of Informatics, 2009.

15 D. Ornstein. On the existence of stationary optimal strategies. *Proceedings of the American Mathematical Society*, 20:563–569, 1969.

16 Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1st edition, 1994.

17 Manfred Schäl. Markov decision processes in finance and dynamic options. In *Handbook of Markov Decision Processes*, pages 461–487. Springer, 2002.

18 Olivier Sigaud and Olivier Buffet. *Markov Decision Processes in Artificial Intelligence*. John Wiley & Sons, 2013.

19 R.S. Sutton and A.G Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. MIT Press, 2018.

Determinization of Büchi Automata: Unifying the Approaches of Safra and Muller-Schupp

Christof Löding
RWTH Aachen University, Ahornstr. 55, 52074 Aachen, Germany
loeding@cs.rwth-aachen.de

Anton Pirogov
RWTH Aachen University, Ahornstr. 55, 52074 Aachen, Germany
pirogov@cs.rwth-aachen.de

Abstract

Determinization of Büchi automata is a long-known difficult problem, and after the seminal result of Safra, who developed the first asymptotically optimal construction from Büchi into Rabin automata, much work went into improving, simplifying, or avoiding Safra's construction. A different, less known determinization construction was proposed by Muller and Schupp. The two types of constructions share some similarities but their precise relationship was still unclear. In this paper, we shed some light on this relationship by proposing a construction from nondeterministic Büchi to deterministic parity automata that subsumes both constructions: Our construction leaves some freedom in the choice of the successor states of the deterministic automaton, and by instantiating these choices in different ways, one obtains as particular cases the construction of Safra and the construction of Muller and Schupp. The basis is a correspondence between structures that are encoded in the macrostates of the determinization procedures – Safra trees on one hand, and levels of the split-tree, which underlies the Muller and Schupp construction, on the other hand. Our construction also allows for mixing the mentioned constructions, and opens up new directions for the development of heuristics.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects

Keywords and phrases Büchi automata, determinization, parity automata

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.120

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version A full version of the paper is available at `https://arxiv.org/abs/1902.02139`.

Funding *Anton Pirogov*: This author is supported by the German research council (DFG) Research Training Group 2236 UnRAVeL

1 Introduction

Büchi automata are finite automata for infinite words, and were initially introduced to show decidability of the logic S1S [2]. Infinite words can be used to model infinite execution traces of reactive, non-terminating systems, and serve as a translation target from logics like LTL (see, e.g., [15, 5]), which is a popular and well-understood specification formalism. For this reason, Büchi automata nowadays play a central role in formal methods like model-checking [1] and runtime-verification [6], because they can represent all ω-regular languages and are suitable for efficient algorithmic treatment. Unfortunately, the simplicity of the Büchi acceptance condition makes it crucially dependent on nondeterminism, i.e., not every ω-regular language (or LTL formula) can be accepted by a deterministic Büchi automaton (see, e.g., [16]). In some settings, this nondeterminism causes difficulties, such that algorithms require a representation of the property by a deterministic automaton, like in probabilistic model-checking (see, e.g., [1, Section 10.3]), or in synthesis (see [17] for an overview of the theory, and [9] for recent developments in practice).

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 120; pp. 120:1–120:13


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

A first determinization procedure that translates nondeterministic Büchi automata into deterministic automata was presented in [8]. The first asymptotically optimal and most well-known determinization construction for Büchi automata is the construction of Safra [13]. It translates a nondeterministic Büchi automaton with n states into a deterministic Rabin automaton with at most $2^{\mathcal{O}(n \log n)}$ states and $\mathcal{O}(n)$ sets in the acceptance condition. In applications like synthesis, the deterministic automaton is used to build a game that inherits as winning condition the acceptance condition of the automaton. In the theory of infinite duration games, the parity condition plays a central role (see, e.g., the survey [18]). For this reason, Piterman modified Safra's construction in order to directly obtain a parity automaton [11]. This construction was reformulated in [14], where also a tighter analysis of its state complexity is given with an upper bound of $\mathcal{O}(n!^2)$ for the number of states. A similar construction is presented in [12], adapted to the translation of ω-regular expressions directly into parity automata.

It is known that the Safra construction is essentially optimal [3], so there is no hope of significantly improving the worst-case upper bounds of the known constructions. However, the data structure of Safra trees (or history trees) that is used for the states of the deterministic automata, is challenging to deal with in implementations. Therefore, alternative approaches for determinization have been studied, leading to a family of constructions that are based on a construction by Muller and Schupp, which appeared in [10] as a by-product of a translation from alternating to non-deterministic tree automata. An explicit description of the construction specifically for determinization of Büchi automata is presented in [7]. A refinement of that construction is presented in [4], in which the states of the deterministic automaton are no longer represented as trees but as ordered and labelled tuples of sets.

The two approaches of Safra and Muller-Schupp show some similarities, as pointed out in the conclusion of [4], but from the existing formulations of the constructions, their precise relationship is not clear.

In this paper, we provide a construction for transforming nondeterministic Büchi automata into deterministic parity automata that cleanly explains the connections between the approaches of Safra and Muller-Schupp. It turns out that both types of constructions can be formulated on the same data structure, which can either be understood as ordered tuples of sets in which each set has an additional rank (a natural number), or as Safra trees in which each node has an additional rank (the same structure is essentially used in the constructions from [11] and [14]). The transitions are defined in terms of a sequence of simple operations, and it turns out that the two constructions only differ in one of these operations. In summary, our contributions are the following:

- We provide a new and relatively simple formulation of a Muller-Schupp style determinization construction that yields deterministic parity automata. Compared to previous constructions from [7] and [4], we encode less information in the states, and obtain a construction that has the same worst-case upper bound as the Safra style constructions.
- We extend our Muller-Schupp style construction by introducing a degree of freedom in the choice of the successor states. This freedom can be used to make the construction correspond to Safra's construction as presented in [11] and [14]. We therefore obtain a construction that unifies the approaches of Safra and Muller-Schupp in one general construction. Furthermore, the freedom in the choice of the successors of transitions also yields new ways of obtaining deterministic parity automata, and can be used in implementations as a heuristic to reduce the state space of the resulting automaton.

This work is organized as follows. After introducing the basic notations in Section 2, we present the new variant of the Muller-Schupp construction in Section 3, and then briefly review Safra's construction in Section 4. We explain the structural relationship between

those two constructions in Section 5, and finally introduce our generalized construction as a simple extension of the presented Muller-Schupp construction in Section 6. In Section 7 we discuss and conclude.

2 Preliminaries

First we briefly review basic definitions concerning ω-automata and ω-languages. If Σ is a finite alphabet, then Σ^ω is the set of all infinite *words* $w = w_0 w_1 \dots$ with $w_i \in \Sigma$. For $w \in \Sigma^\omega$ we denote by $w(i)$ the i-th symbol w_i. For convenience, we write $[n]$ for the set of natural numbers $\{1, \dots, n\}$. A *Büchi automaton* $\mathcal{A}$ is a tuple $(Q, \Sigma, \Delta, Q_0, F)$, where Q is a finite set of states, Σ a finite alphabet, $\Delta \subseteq Q \times \Sigma \times Q$ is the transition relation and $Q_0, F \subseteq Q$ are the sets of initial and accepting states, respectively. When Q is understood and $X \subseteq Q$, then $\overline{X} := Q \setminus X$. We write $\Delta(p, x) := \{q \mid (p, x, q) \in \Delta\}$ to denote the set of *successors* of p on symbol x and $\Delta(P, x)$ for $\bigcup_{p \in P} \Delta(p, x)$. A *run* of an automaton on a word $w \in \Sigma^\omega$ is an infinite sequence of states $q_0, q_1, \dots$ starting in some $q_0 \in Q_0$ such that $(q_i, w(i), q_{i+1}) \in \Delta$ for all $i \geq 0$. An automaton is *deterministic* if $|Q_0| = 1$ and $|\Delta(p, x)| \leq 1$ for all $p \in Q, x \in \Sigma$, and *non-deterministic* otherwise. In this work, we assume Büchi automata to be non-deterministic and refer to them as NBA. A *transition-based deterministic parity automaton* (TDPA) is a deterministic automaton $(Q, \Sigma, \Delta, Q_0, c)$ where instead of $F \subseteq Q$ there is a *priority function* $c : \Delta \to \mathbb{N}$ assigning a natural number to each transition.

A run of an NBA is *accepting* if it contains infinitely many accepting states. A run of a TDPA is accepting if the smallest priority that appears infinitely often on transitions along the run is even. An automaton $\mathcal{A}$ *accepts* $w \in \Sigma^\omega$ if there exists an accepting run on w, and the language $L(\mathcal{A}) \subseteq \Sigma^\omega$ *recognized* by $\mathcal{A}$ is the set of all accepted words. To avoid confusion, we sometimes refer to states of TDPA that we construct as *macrostates* to distinguish them from the states of the underlying Büchi automaton.

3 A Simplified Muller-Schupp Construction

The essential idea for determinization using the Muller-Schupp construction is the following: given some Büchi automaton $\mathcal{A}$ and input word w, the resulting deterministic automaton conceptually traverses a specific run-tree of $\mathcal{A}$ on w, called reduced split-tree in [7], and tracks enough information to decide whether an infinite path with a specific shape exists in this tree. Such a path is known to exist if and only if w is accepted by $\mathcal{A}$. The construction presented in [7] uses a structure called contraction trees in order to track the relevant information. This has been simplified in [4] to macrostates that consist of an ordered tuple of disjoint sets of Büchi states, and two preorders over the states appearing in the tuple.

In this section, we further simplify the structure of the macrostates for the deterministic automaton to ordered tuples of disjoint sets of Büchi states, and a single additional linear order on these sets (formally expressed as a ranking function that assigns to each set a natural number). This also results in a relatively simple transition function on the macrostates.

The *reduced split-tree* $t^{rs}(\mathcal{A}, w)$ for NBA $\mathcal{A}$ and word $w \in \Sigma^\omega$ is an ordered infinite tree in which the nodes are labelled by state-sets, and each node has at most two successors. Formally, it is constructed as follows. The first level of the tree consists of the root node labelled by the initial states Q_0. To construct level $i + 1$ from level i, for each node at level i labelled by set S of states, let the *left child* of S be labelled by $\Delta(S, w(i)) \cap F$ and the *right child* by $\Delta(S, w(i)) \cap \overline{F}$, i.e., accepting and non-accepting successor states are separated. Then keep only the leftmost (wrt. the natural ordering of neighbors) occurrence of each state

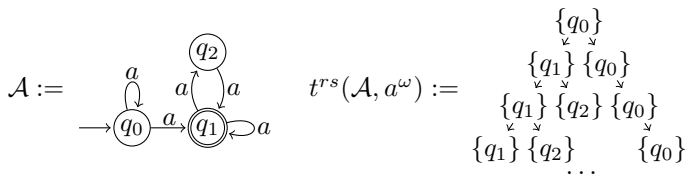


Figure 1 Example of a reduced split-tree $t^{rs}(\mathcal{A}, a^\omega)$ of an NBA $\mathcal{A}$. It has an infinite path representing the run q_0^ω and a left-path representing the run $q_0 q_1^\omega$, from which finite paths $q_0 q_1^* q_2$ branch off.

in the level and finally remove nodes labelled by $\emptyset$. Clearly, because of the normalization, the number of nodes on each level can be at most $|Q|$. An example of a reduced split-tree is shown in Figure 1. We call an infinite path in the tree that takes the left branch infinitely often a *left-path*. Reduced split-trees have the following useful property:

▶ **Lemma 1** ([7], Lemma 2). *$\mathcal{A}$ accepts $w \iff t^{rs}(\mathcal{A}, w)$ has a left-path.*

In the following, we identify nodes in the same level with their label sets. To obtain a deterministic automaton, we augment the nodes of the reduced split-tree with number tokens that we call *(age-)ranks*, which are used to infer a left-path.

The new macrostates in the deterministic automaton represent levels of reduced split-trees and consist of a tuple of disjoint non-empty sets $t := (S_1, S_2, \ldots, S_n)$ equipped with a bijection $\alpha : [n] \to [n]$ satisfying $\alpha(n) = 1$, which assigns to each set S_i the rank $\alpha(i)$. We call a pair (α, t) that satisfies these constraints *ranked slice* and we call it *pre-slice*, if t contains empty sets or α is not a bijection. Notice that all macrostates are ranked slices, whereas pre-slices occur only during intermediate steps. We introduce the following useful notations to work with ranked slices. Let $|t| := n$ and $Q_t := \bigcup_{i=1}^{|t|} S_i$. The function $\mathsf{idx} : Q_t \to [|t|]$ maps each state $q \in Q_t$ to the tuple index i such that $q \in S_i$, and by $\alpha(q)$ we denote $\alpha(\mathsf{idx}(q))$ for $q \in Q_t$.

When reading symbol $x \in \Sigma$ in macrostate (α, t), the successor macrostate (α', t') is obtained by a sequence of successive operations step, prune and normalize, where, roughly,

- step interprets t as nodes on a reduced split-tree level and calculates the next level sets,
- prune removes the empty sets produced by step, reassigning ranks in a specific way, and
- normalize just turns the ranking function obtained after prune into a bijection again.

Below, we formally define these operations (step and prune are illustrated in Figure 2).

First we describe step, which constructs the next level of the reduced split-tree and passes each existing rank on to the respective right child. Let

$$\Delta_t(q, x) := \Delta(q, x) \setminus \Delta\Big(\bigcup_{i=1}^{\mathsf{idx}(q)-1} S_i, x\Big),$$

restricting for each state $q \in Q_t$ the successors to only those which are not reached by some other state located in a set to the left of q. Then, for each node S_i let $\hat{S}_{2i-1} := \Delta_t(S_i, x) \cap F$ be the *left child* and $\hat{S}_{2i} := \Delta_t(S_i, x) \cap \overline{F}$ the *right child*, containing the accepting and non-accepting normalized successors, respectively. Let $\hat{\alpha}(2i) := \alpha(i)$ and $\hat{\alpha}(2i-1) := n+1$, i.e., the right children inherit the rank of the parent and the left children all get the same new maximal rank $n+1$, resulting in a pre-slice $(\hat{\alpha}, \hat{t})$.

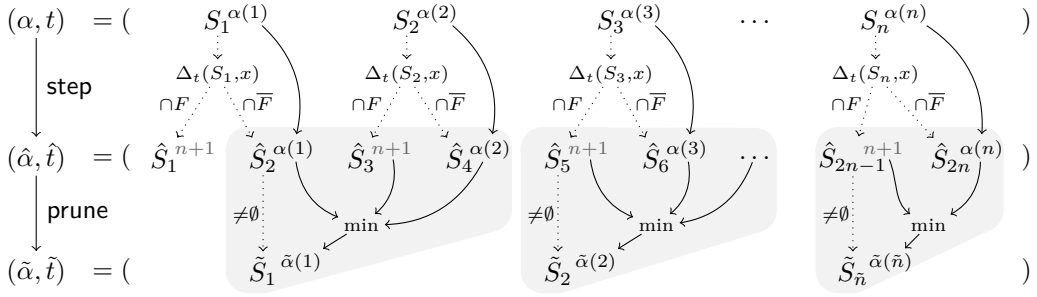


Figure 2 Abstract illustration of step and prune in a Muller-Schupp transition on some $x \in \Sigma$. The superscripts represent the assigned ranks. First, step calculates the normalized successors, separating accepting from non-accepting states and passing the parent rank on to the right child. In the illustration, we assume that the sets $\hat{S}_i \neq \emptyset$ for $i \in \{2, 5, 2n-1\}$, i.e., $x_1 = 2, x_2 = 5, x_{\tilde{n}} = x_3 = 2n-1$. Then prune keeps sets at these positions for the resulting tuple $\tilde{t}$, and $\tilde{\alpha}$ is obtained by taking the minimum of the ranks given by $\hat{\alpha}$ in the ranges spanning from one x_i up to the position before x_{i+1}. Finally $t' := \tilde{t}$ and $\tilde{\alpha}$ is normalized to α', while preserving strict ordering between positions wrt. $\tilde{\alpha}$. The dotted edges connect parent sets (in the top row) and resulting left/right children sets (bottom row) in the conceptual reduced split-tree, the solid edges show the movements of the rank values assigned to the sets.

Intuitively, in the prune operation, all ranks that mark empty sets after step are relocated onto the closest non-empty set to the left (or removed, if no such set exists). When multiple ranks occupy the same set, then the smallest one is preserved. Ranks that moved to the left in this way and are not removed, indicate a good (green) event, whereas ranks which were removed indicate a bad (red) event.

Formally, let $x_1 < x_2 < \ldots < x_{\tilde{n}}$ be the increasing sequence of all indices such that $\hat{S}_{x_j} \neq \emptyset$. Then prune returns $(\tilde{\alpha}, \tilde{t})$ with the tuple $\tilde{t} := (\hat{S}_{x_1}, \ldots, \hat{S}_{x_{\tilde{n}}})$ without empty sets, where $\tilde{\alpha}$ is defined as $\tilde{\alpha}(i) := \min\{\hat{\alpha}(j) \mid x_i \leq j < x_{i+1}\}$ with $x_{\tilde{n}+1} := |\hat{t}| + 1$.

The set of *green* ranks is given by $G := \mathsf{img}(\tilde{\alpha}) \cap \{\hat{\alpha}(j) \mid \hat{S}_j = \emptyset\}$, where $\mathsf{img}(\tilde{\alpha})$ denotes the image of $\tilde{\alpha}$. These are the ranks that mark empty sets after step and are not removed by prune. The set of *red* ranks given by $R := \mathsf{img}(\hat{\alpha}) \setminus \mathsf{img}(\tilde{\alpha})$ contains the ranks that were not preserved during prune. The set of *active ranks* is $A := G \uplus R$. Let $k := \min A$ (or $k := |Q| + 1$ if $A = \emptyset$) denote the *dominating* rank of the transition, i.e., the smallest active rank. We define the priority p of the transition as $2k$ if $k \in G$ and $2k - 1$ otherwise.

The function $\tilde{\alpha}$ might assign the same rank to several sets, and it might have gaps (unused rank values between used ones). So finally, normalize returns (α', t') with $t' := \tilde{t}$ and a final bijective ranking function $\alpha' : [|t'|] \to [|t'|]$ such that $\tilde{\alpha}(j) < \tilde{\alpha}(k) \Rightarrow \alpha'(j) < \alpha'(k)$ for all $j, k \in \{1, \ldots, |t'|\}$, i.e., a total order which is compatible with the preorder induced by $\tilde{\alpha}$. If there are several such ranking functions α', then any of these works.

A TDPA $\mathcal{B}$ is obtained by taking the initial state (α_0, t_0) with $t_0 := (Q_0), \alpha_0(1) := 1$ and a transition function that picks for each state a valid successor that satisfies the description above, and assigns the corresponding priority p to the edge. Observe that by construction, the sequence of states visited along some word $w \in \Sigma^\omega$ from the initial state represents exactly the levels of $t^{rs}(\mathcal{A}, w)$, marked with ranks.

▶ **Theorem 2.** *For a given NBA with n states, the TDPA obtained by the Muller-Schupp construction accepts the same language as the NBA, and its number of states is in $\mathcal{O}(n!^2)$.*

The correctness follows from the correctness of the generalized construction presented in Section 6. The claim on the state complexity directly follows from the upper bound given in [14, Proposition 2], and the bijection between the set of ranked slices and the set of ranked Safra trees presented in Section 5.

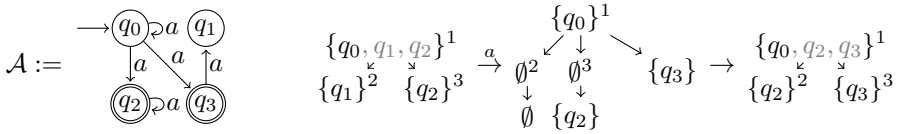


Figure 3 Example of a Safra-tree transition on letter a, based on NBA $\mathcal{A}$. The LIR position of nodes is depicted as superscript of the sets. The "redundant" states that are implicit in our definition are depicted in gray in the initial and resulting tree. In the intermediate step, the tree is depicted after calculating and pruning successor state sets. In the final tree the remaining actions are performed and LIR positions are updated. The transition has a red event for LIR position 2 and a green event for position 3. Because of the removal of the node at position 2 in the LIR, the node that originally was at position 3 moved up, whereas the fresh node labelled by $\{q_3\}$ comes last.

4 Sketch of the Safra Construction

In this section, we roughly illustrate the used structures and operations of the Safra construction along the lines of [11, 14], so that we can demonstrate its relationship with the Muller-Schupp construction in the next section. As before, $\mathcal{A}$ is an NBA with the usual components.

A *Safra tree* is a finite ordered tree with non-empty state-sets as labels. Usually, it is required that a parent is labelled by a strict superset of all states in its subtree and siblings are labelled by pairwise disjoint sets. We use the equivalent requirement that *all* labels in the tree are pairwise disjoint, i.e., refrain from listing states in the parent label which are already present in some descendant. One can easily reconstruct the "full" label set of a node wrt. the classical definition by taking the union of all the labels in its subtree. To obtain parity automata, each node of the Safra tree is associated with a number from $\{1, \ldots, n\}$, where n is the number of nodes in the Safra tree [11]. These numbers satisfy the property that parent nodes have smaller numbers than their children, and a node has a smaller number than its right sibling. The numbers correspond to the ranks that we use in Section 3, and we therefore refer to Safra trees in combination with these numbers as *ranked Safra trees*. Two ranked Safra trees are shown in Figure 3 (and an intermediate tree in the middle).

In [14], a slightly different representation is used based on a *later introduction record* (LIR), which just lists the tree nodes in their introduction order, i.e., nodes appear in this list after parents and older siblings (in this representation, nodes have canonical names depending only on their position in the tree). Safra trees with LIR directly correspond to ranked Safra trees by annotating each tree node with its position in the LIR.

A transition on symbol $x \in \Sigma$ is constructed as follows (see Figure 3 for an example). First, for each label set S, the set $S' := \Delta(S, x)$ of successor states is calculated. After this, each node gets a fresh right-most child, and the accepting states in S', that is $S' \cap F$, are moved into the label of this child. Then, disjointness is ensured by keeping of each state only the copy which is located at the deepest node along the leftmost branch where that state occurs (this stage is represented by the middle tree in Figure 3). If now some internal node has an empty label, but a non-empty subtree (a good event for the node), its subtree is collapsed into a single node by removing all descendants and moving the states in their labels into the parent label. Finally, all remaining sets that are labelled by $\emptyset$ are removed (being removed is a bad event for a node). In the following, we refer to good and bad events as *green* and *red*, respectively. The priority for the transition is derived from the green and red events, which are associated with the relative position of the corresponding nodes in the LIR. The LIR for the new tree is obtained by deleting removed nodes from the LIR and appending fresh nodes that remain in the resulting tree in arbitrary order.

5 From Safra-trees to ranked slices and back

In this section we state the key observation that was the starting point of this work: there is a bijection between the set of ranked slices and the set of ranked Safra trees. From a ranked Safra tree, one obtains the ranked slice by simply listing the nodes of the Safra tree by a depth-first post-order traversal (i.e., a parent processed after all its children). We formalize this relationship below, and then explain that the transitions defined in the Muller-Schupp construction and in the Safra construction are very similar, which then leads to the unified construction.

Let (α, t) be a ranked slice with $t = (S_1, \ldots, S_n)$. The tuple index of the *parent* of S_i is the closest index to the right of i that has a smaller rank and is formally defined as $\uparrow(i) := \min_{i<k\leq n}\{k \mid \alpha(k) < \alpha(i)\}$. As we require by definition of ranked slices that the rightmost position in the tuple always has rank 1, this is the only position in the tuple for which the parent is undefined. The ordered tree induced by $\uparrow$, with siblings in tuple index order, is called the *rank tree* of (α, t). The tuple index of the *left subtree boundary* of S_i is the closest index to the left with a smaller rank, and is denoted by $\leftarrow(i) := \max_{1\leq k<i}\{k \mid \alpha(k) < \alpha(i)\}$ or 0 if no such index exists. It points to either the direct left sibling of i, or the left sibling of the closest ancestor, if one exists. Effectively, $\leftarrow(i)$ is the closest neighbor to the left which is not a descendant of i. As children by definition are always to the left of their parents, every node at indices $\leftarrow(i)+1, \ldots, i$ is in the subtree of i.

For an example, consider the tuple $(\{q_3\}^4, \{q_1\}^2, \{q_2\}^3, \{q_0\}^1)$, where the superscripts denote the assigned rank (e.g., $\alpha(1) = 4$). The rightmost position 4 of the tuple is the root of the tree. For the positions 2 and 3, which have rank 2 and 3 respectively, the next position to the right with a smaller rank is in both cases position 4, i.e., $\uparrow(2) = \uparrow(3) = 4$. Finally, position 1 in the tuple has position 2 as parent, i.e., $\uparrow(1) = 2$. The discussed tuple is depicted with the parent edges at the bottom right of Figure 4. There is also one non-trivial left subtree boundary in this tuple, assigned by $\leftarrow(3) = 2$, i.e. index 2 is not in the subtree of index 3, and in this case is an actual left sibling of index 3.

We use the notation $\uparrow_\alpha := \alpha \circ \uparrow \circ \alpha^{-1}$ to denote the parent rank of another rank directly, without mentioning the indices in the tuple. In the previous example, we have $\uparrow_\alpha(4) = 2$, and $\uparrow_\alpha(2) = \uparrow_\alpha(3) = 1$. We identify the age-ranks $\alpha(i)$ as nodes of the tree, while each set S_i determines the label of the node $\alpha(i)$, called *hosted set*. We write $S_i^{\downarrow} := \bigcup_{k=\leftarrow(i)+1}^{i} S_k$ for the *subtree set* of node $\alpha(i)$.

▶ **Definition 3.** *Let* safra2slice *be the mapping which takes a ranked Safra tree and returns* (α, t)*, with* $t := (S_1, \ldots, S_n)$ *being the label sets of the nodes in depth-first post-order (i.e., a parent processed after all its children) traversal order and ranking* α *defined by the ranks of the corresponding Safra tree nodes.*

Let slice2safra *be the mapping which takes a ranked slice* (α, t) *and returns the ranked Safra tree given by the rank tree of* (α, t)*, i.e. the tree structure defined by* $\uparrow$ *and the ordering of siblings given by the order of the corresponding sets in* t*.*

It is easy to see from the definitions that safra2slice and slice2safra are injective and return a valid ranked slice and ranked Safra tree, respectively. This implies that there exists a bijection between the sets of ranked Safra trees and ranked slices. It is also not very hard to see that the following holds:

▶ **Lemma 4.** safra2slice *and* slice2safra *are inverses of each other and provide a bijection between ranked Safra trees and ranked slices.*

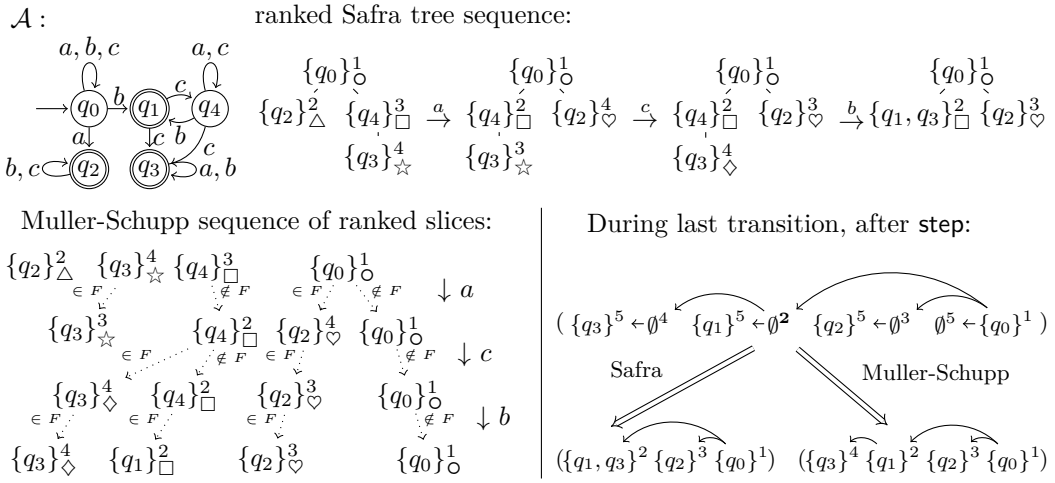


Figure 4 Transitions based on NBA $\mathcal{A}$ using both constructions. The superscripts denote the ranks of tree nodes / sets in the slice tuple. The subscripts are added for illustration purposes and conceptually track nodes throughout time, i.e., the same symbol marks the "same" node at different times. The algorithms agree on all but the last transition, where they differ due to different handling of green nodes/ranks, in this case rank 2 that marks an empty set after calculating and splitting the successors (illustrated on the bottom right). In the Muller-Schupp case, the rank is moved left during prune, resulting in a child being pulled into the parent in the rank tree, whereas in the Safra construction the whole subtree is collapsed. The solid edges between sets depict the rank tree induced by $\uparrow$, dotted edges depict the edges in the conceptual split-tree. In the bottom right the slices are shown together with their tree interpretation.

As we have established that both constructions, Muller-Schupp and Safra, operate on essentially the same structures, from now on we talk about ranked slices and trees interchangeably. Using this relationship, one can take the same tree/slice and apply both the successor calculation of the Safra construction and of the Muller-Schupp construction to it. What one first notices, is that the resulting tree/slice is very similar or equal in many cases. This is owed to the fact that most operations in one construction have an equivalent operation in the other, just formulated for the other representation.

For example, moving accepting successor states into a fresh child node in Safra's construction corresponds to splitting accepting successors from non-accepting ones during step in the Muller-Schupp construction, as in the successor tuple the new child (in the conceptual split-tree) gets a fresh, larger rank and by definition becomes the rightmost child in the rank tree of the resulting new slice. The normalization steps that make the successor sets pairwise disjoint also yield the same results. The ranks of nodes with green events in the Safra construction coincide with ranks of sets that signal green in the ranked slices, and ranks of Safra nodes with red events with ranks of sets that signal red. The removal of empty sets by prune and renumbering the ranks with normalize is the same as the removal of the corresponding nodes in the Safra tree and updating the LIR, i.e., the ranks of Safra nodes.

In fact, *the only difference* between the constructions is what happens with a tree node in case of a green event. Recall that in Safra's construction, the whole subtree of a green node is collapsed to a single node. In the Muller-Schupp construction, the green ranks are those that end up on an empty set after step, and that survive the prune operation, in which the ranks are moved to the next non-empty set to the left, and only the minimal ones are kept on each non-empty set. In the view of ranked trees, this corresponds to a green node absorbing its rightmost, uppermost child node into it, while keeping the rest of the subtree unchanged. See Figure 4 for an illustration.

After observing that both constructions differ in only a minor step and noticing that both yield correct (but possibly different) automata, it becomes apparent that the exact step performed for green events is not essential and there must be a more general mechanism to uncover. The construction we present in Section 6 results from this line of thought.

On the practical side, it is worth mentioning that the cost of switching between the representations using the presented bijection is negligible – the traversal of a ranked Safra tree to obtain a ranked slice is obviously possible in linear time. For the other direction there also exists a simple linear time algorithm that calculates the parent and left subtree boundary relation from the ranking α.

6 The unified construction

In this section, we present a construction that builds on the Muller-Schupp construction from Section 3, and unifies it with Safra's construction by adding another operation, called merge, between prune and normalize: $(\alpha, t) \xrightarrow{\text{step}} (\hat{\alpha}, \hat{t}) \xrightarrow{\text{prune}} (\tilde{\alpha}, \tilde{t}) \xrightarrow{\text{merge}} (\check{\alpha}, \check{t}) \xrightarrow{\text{normalize}} (\alpha', t')$.

This new operation is nondeterministic, and can be instantiated in different ways. In particular, it can be instantiated trivially and thus corresponds to the Muller-Schupp construction, and it can be used to emulate the Safra construction.

We first describe the idea of merge, and then give a formal definition. Assume that, after step and prune have been applied to some ranked slice (α, t), we have the pre-slice $(\tilde{\alpha}, \tilde{t})$, and the dominating (minimal active) rank k (determined by prune, see Section 3). Then merge can collapse groups of neighbouring sets in the tuple, and preserves the minimum rank from each collapsed range, similar to prune. In contrast to prune, which "merges" one non-empty set with multiple empty sets in a deterministic manner, merge may actually take the union of multiple adjacent non-empty sets, depending on the ranks currently assigned to them.

The non-overlapping intervals of sets that are collapsed together are not uniquely determined in general. They only have to satisfy the constraints that no sets with rank smaller than the dominating rank k are merged with anything else, and that the set with rank k is not merged with anything to the right of it. These constraints are important for the correctness, and ensure that in the ranked Safra tree perspective, the nodes with rank smaller than k do not change, and that the node with the dominating rank k is not merged with sets outside of its subtree.

Formally, merge returns a pre-slice $(\check{\alpha}, \check{t})$ obtained in the following way (see Figure 5 for an illustration). Let $I_1, I_2, \ldots, I_{n'}$ be a sequence of sets partitioning the set of indices $\{1, \ldots, \tilde{n}\}$ in $\tilde{t}$ into adjacent groups, i.e., $\min I_1 = 1$, $\max I_{n'} = \tilde{n}$ and for all $j > 1$ we have $\min I_j = \max I_{j-1} + 1$. This grouping should satisfy the following property for all $1 \leq j \leq n'$ and $l \in I_j$: if $\tilde{\alpha}(l) < k$, then $|I_j| = 1$, and if $\tilde{\alpha}(l) = k$, then $\max I_j = l$. Then the pre-slice $(\check{\alpha}, \check{t})$ is defined by the sets $\check{S}_i := \bigcup_{j \in I_i} \tilde{S}_j$ and the ranking function $\check{\alpha}(i) := \min\{\tilde{\alpha}(j) \mid j \in I_i\}$ for all $i \in \{1, \ldots, n'\}$, i.e., for each interval, the union of the sets and the smallest rank is taken.

As in the Muller-Schupp construction, normalize is applied to $(\check{\alpha}, \check{t})$ to obtain the successor macrostate (α', t'). This extended transition relation is used to obtain the transition-based deterministic parity automaton, as before.

An example showing how the choice of different merge strategies leads to different successor states is illustrated in Figure 6. Observe that we can recover the Muller-Schupp construction by using the identity function for merge, or in other words, putting each index into its own interval, which is the finest partitioning of indices that satisfies the requirements on merge. On the other hand, we can also take the coarsest compatible partitioning, i.e., minimize the number of intervals. We call this kind of update *maximal collapse*.

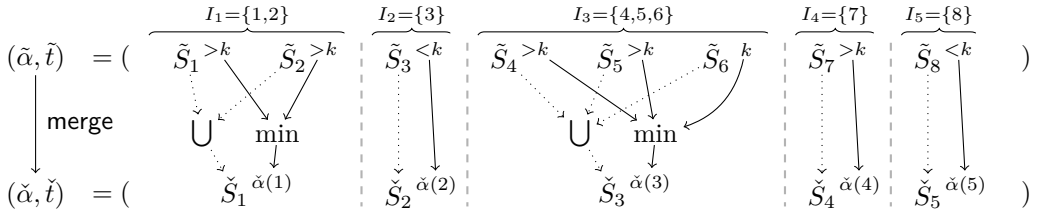


Figure 5 Illustration of the general merge operation that comes after prune and before normalize, with the minimal active rank k and ranks depicted as set superscripts. The illustrated intervals are the coarsest partitioning of indices in $\tilde{t}$ satisfying the constraints.

We can emulate a Safra-update by imposing some additional constraints on the intervals, ensuring that only the complete subtrees of nodes with green ranks are merged. More concretely, we require that intervals that are not singletons span exactly the nodes of the complete subtree that is rooted in a green rank in the view of the slice as ranked Safra tree. Note that for an index ℓ in the tuple, the subtree of the corresponding node in the ranked Safra tree corresponds to the interval that starts one step right of the left subtree boundary of ℓ, and ends in ℓ, that is, the interval $\leftarrow(l)+1, \ldots, \ell$. Thus, for imitating the Safra merge rule, the intervals $I_1, I_2, \ldots, I_{n'}$ from merge are the unique smallest intervals satisfying

$$\forall i \in [n'], l \in I_i : \quad \tilde{\alpha}(l) \in G \implies \leftarrow(l)+1 \in I_i \quad \text{(complete subtrees collapsed)}.$$

▶ **Proposition 5.** *The operation* merge *can be instantiated such that the transitions of the constructed TDPA correspond to the transitions of the Muller-Schupp construction or to the transitions of the Safra construction.*

Notice that for all merge rules except for the Muller-Schupp update, the relationship of ranked slices and consecutive levels of the reduced split-tree (see Section 3) breaks down. One can, however, reflect the merges also in the reduced split-tree by doing the merges of the corresponding sets on each level, which leads to an acyclic graph instead of a tree. This view is helpful in the correctness proof of the construction.

▶ **Theorem 6.** *Let $\mathcal{A}$ be an NBA. Then a deterministic parity automaton $\mathcal{B}$, obtained by the described determinization construction, has at most $\mathcal{O}(n!^2)$ states and recognizes the same language as $\mathcal{A}$.*

The upper bound holds because the same macrostates are used as in the presented Muller-Schupp construction in Section 3. The correctness can be shown by a refinement of the original correctness proof of the Safra construction [13].

7 Discussion and Conclusion

We have presented a new variant of the Muller-Schupp construction for determinization of Büchi automata into parity automata, reducing the information stored in the macrostates to ordered tuples of disjoint sets annotated with ranks. These ranked slices are in bijection with ranked Safra trees, which leads to a general construction that can emulate the Muller-Schupp construction and the Safra-construction. This answers, in some sense, the question from [4] on the relation between the two types of constructions.

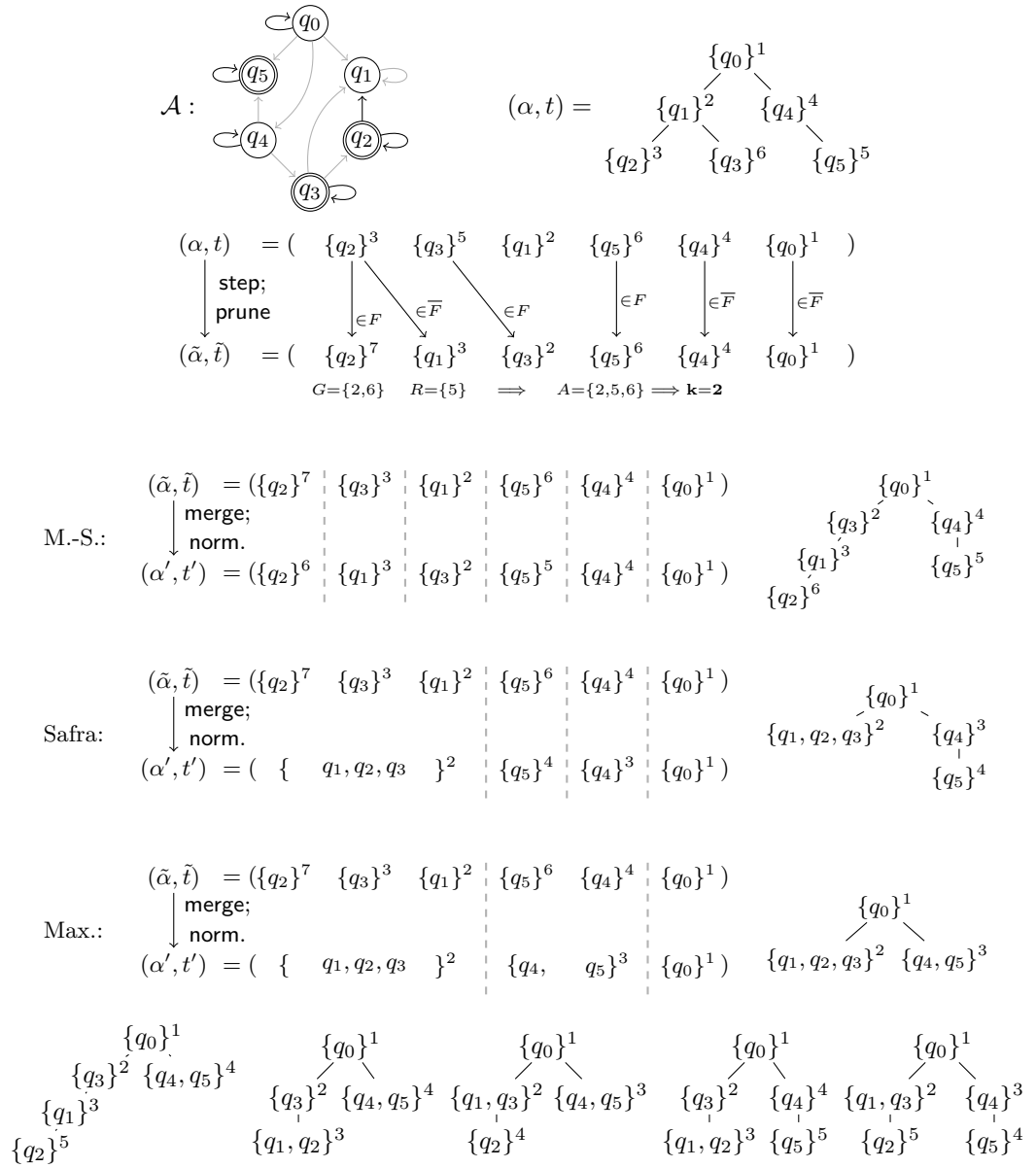


Figure 6 $\mathcal{A}$ illustrates the relevant part of an NBA during a transition on some symbol $x \in \Sigma$, that is, the arrows correspond to the x-transitions of $\mathcal{A}$. The gray edges are the ones pruned in the reduced transition relation Δ_t. The current macrostate (α, t) is represented as the rank tree to the right of $\mathcal{A}$, and as ranked slice below $\mathcal{A}$. The step and prune operations (see Fig. 2 for details) result in ranks 1,3 and 4 being passed down along the right child. Ranks 2 and 6 are moved to the left and hence are green. Rank 5 is overwritten by 2 and hence is red. Rank 7 is a fresh rank which is larger than the others. The dominating rank k is 2. The choice of different merge intervals (as shown in Fig. 5) results in different successors. The successors for the three discussed variants, Muller-Schupp, Safra, and maximal collapse, are shown as rank trees on the right. The 5 other permitted successors are depicted at the bottom.

In general, one can obtain many different valid deterministic automata by choosing different deterministic transition functions that are compatible with the described successor relation. One can also imagine this as constructing a non-deterministic automaton with *all* permitted successors, and then pruning the edges arbitrarily, while preserving for each state only one outgoing transition for each symbol, to "carve" out a valid deterministic automaton.

This non-determinism comes from two sources. One degree of freedom in our construction comes from the different ways of assigning ranks (to new nodes, and when closing gaps resulting from deleted ranks). This freedom is already mentioned in [14]. But here the flexibility is just in the choice of the specific permutation, which still describes structurally the same tree in any case. The novel and in our opinion powerful degree of freedom in our construction is the possibility for different valid merge operations, which allows for a vastly larger pool of possible successors, as the results may describe structurally different trees. Furthermore, the smaller the smallest active rank, the more different a permitted successor may look like.

We have explicitly mentioned the merge strategies that lead to the Muller-Schupp and Safra constructions, and also have mentioned a third strategy, the maximal collapse rule that merges as many sets as possible (as shown in e.g. Figure 6). We also want to point out that, while fixing one such merge-rule for the whole construction is the simplest implementation, the construction permits using *any* valid successor without the need to disambiguate the merge operation beforehand, i.e., picking the successor of a state from the set of permitted ones is a *local* choice. One may think of schemes where the successor is chosen dynamically, depending on the input or already computed information. For example, one can check whether a valid successor has already been constructed, and only add a new state according to a fixed policy if this is not the case. We have already implemented a prototype making use of such an optimization (among others) with encouraging results.

We also want to point out that the presented construction works equally well with transition-based Büchi automata as input, in which case the step operation separates states which are reached by at least one accepting transition from those that are not. One can easily verify that this does not impact the reasoning in the proofs.

It is also possible to adapt the construction to yield Rabin automata, such that the corresponding Safra construction as presented in [14] is subsumed. In this setting, however, the presentation of macrostates as ordered tuples of sets is less natural. Furthermore, in this setting the merges of sets needs to be restricted to subtrees of green nodes, because there is no total order of importance of nodes as provided by the ranks.

References

1 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

2 J Richard Büchi. On a decision method in restricted second order arithmetic. In *Studies in Logic and the Foundations of Mathematics*, volume 44, pages 1–11. Elsevier, 1966.

3 Thomas Colcombet and Konrad Zdanowski. A Tight Lower Bound for Determinization of Transition Labeled Büchi Automata. In *ICALP 2009*, volume 5556 of *Lecture Notes in Computer Science*, pages 151–162. Springer, 2009.

4 Seth Fogarty, Orna Kupferman, Moshe Y Vardi, and Thomas Wilke. Profile trees for Büchi word automata, with application to determinization. *Information and Computation*, 245:136–151, 2015.

5 Paul Gastin and Denis Oddoux. Fast LTL to Büchi Automata Translation. In *Computer Aided Verification, 13th International Conference, CAV 2001, Paris, France, July 18-22, 2001, Proceedings*, pages 53–65, 2001. doi:10.1007/3-540-44585-4_6.

6 Dimitra Giannakopoulou and Klaus Havelund. Automata-Based Verification of Temporal Properties on Running Programs. In *16th IEEE International Conference on Automated Software Engineering (ASE 2001), 26-29 November 2001, Coronado Island, San Diego, CA, USA*, pages 412–416, 2001. doi:10.1109/ASE.2001.989841.

7 Detlef Kähler and Thomas Wilke. Complementation, disambiguation, and determinization of Büchi automata unified. In *ICALP 2008*, pages 724–735. Springer, 2008.

8 Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and control*, 9(5):521–530, 1966.

9 Philipp J. Meyer, Salomon Sickert, and Michael Luttenberger. Strix: Explicit Reactive Synthesis Strikes Back! In *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part I*, pages 578–586, 2018. doi:10.1007/978-3-319-96145-3_31.

10 David E Muller and Paul E Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra. *Theoretical Computer Science*, 141(1-2):69–107, 1995.

11 Nir Piterman. From nondeterministic Buchi and Streett automata to deterministic parity automata. In *Logic in Computer Science, 2006 21st Annual IEEE Symposium on*, pages 255–264. IEEE, 2006.

12 Roman R Redziejowski. An improved construction of deterministic omega-automaton using derivatives. *Fundamenta Informaticae*, 119(3-4):393–406, 2012.

13 Shmuel Safra. On the complexity of omega-automata. In *29th Annual Symposium on Foundations of Computer Science, 1988.*, pages 319–327. IEEE, 1988.

14 Sven Schewe. Tighter bounds for the determinisation of Büchi automata. In *FOSSACS*, pages 167–181. Springer, 2009.

15 Fabio Somenzi and Roderick Bloem. Efficient Büchi Automata from LTL Formulae. In *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings*, pages 248–263, 2000. doi:10.1007/10722167_21.

16 Wolfgang Thomas. Languages, Automata, and Logic. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages, Vol. 3*, pages 389–455. Springer-Verlag New York, Inc., New York, NY, USA, 1997.

17 Wolfgang Thomas. Church's Problem and a Tour through Automata Theory. In *Pillars of Computer Science, Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday*, pages 635–655. Springer, 2008. doi:10.1007/978-3-540-78127-1.

18 Moshe Y. Vardi and Thomas Wilke. Automata: from logics to algorithms. In *Logic and automata - history and perspectives*, volume 2 of *Texts in Logic and Games*, pages 629–724. Amsterdam University Press, 2007.

Optimal Regular Expressions for Permutations

Antonio Molina Lovett
University of Waterloo, Canada
ajmolina@uwaterloo.ca

Jeffrey Shallit
University of Waterloo, Canada
https://cs.uwaterloo.ca/~shallit/
shallit@uwaterloo.ca

Abstract

The permutation language P_n consists of all words that are permutations of a fixed alphabet of size n. Using divide-and-conquer, we construct a regular expression R_n that specifies P_n. We then give explicit bounds for the length of R_n, which we find to be $4^n n^{-(\lg n)/4+\Theta(1)}$, and use these bounds to show that R_n has minimum size over all regular expressions specifying P_n.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases regular expressions, lower bounds, divide-and-conquer

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.121

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version https://arxiv.org/abs/1812.06347

1 Introduction

Given a regular language L defined in some way, it is a challenging problem to find good upper and lower bounds on the size of the smallest regular expression specifying L. (In this paper, by a regular expression, we always mean one using the operations of union, concatenation, and Kleene closure only.) Indeed, as a computational problem, it is known that determining the shortest regular expression corresponding to an NFA is PSPACE-hard [12]. Jiang and Ravikumar proved the analogous result for DFAs [11]. For more recent results on inapproximability, see [9].

For nontrivial families of languages, only a handful of results are already known. For example, Ellul et al. [7] showed that the shortest regular expression for the language $\{w \in \{0,1\}^n : |w|_1 \text{ is even}\}$ is of length $\Omega(n^2)$. Here $|w|_1$ denotes the number of occurrences of the symbol 1 in the word w. (A simple divide-and-conquer strategy provides a matching upper bound.) Chistikov et al. [4] showed that the regular language

$$\{ij : 1 \leq i < j \leq n\}$$

can be specified by a regular expression of size exactly $n(\lfloor \log_2 n \rfloor + 2) - 2^{\lfloor \log_2 n \rfloor + 1}$, and furthermore this bound is optimal. Mousavi [13] developed a general program for computing lower bounds on regular expression size for the binomial languages

$$B(n,k) = \{w \in \{0,1\}^n : |w|_1 = k\}.$$

Let n be a positive integer, and define $\Sigma_n = \{1, 2, \ldots, n\}$. In this paper we study the finite language P_n consisting of all permutations of Σ_n. Thus, for example,

$$P_3 = \{123, 132, 213, 231, 312, 321\}.$$

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 121; pp. 121:1–121:12

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

We are interested in regular expressions that specify P_n. In counting the length of regular expressions, we adopt the conventional measure of *alphabetic length* (see, for example, [6]): the length of a regular expression is the number of occurrences of symbols of the alphabet Σ_n. Thus, other symbols, such as parentheses and `+`, are ignored.

A brute-force solution, which consists of listing all the members of P_n and separating them by the union symbol `+`, evidently gives a regular expression for P_n of alphabetic length $n \cdot n!$. This can be improved to $n! \sum_{0 \le i < n} 1/i! \sim e \cdot n!$ by tail recursion, where $E(S)$ represents a regular expression for all permutations of the symbols of S:

$$E(S) = \sum_{i \in S} i(E(S - \{i\})); \quad E(i) = i.$$

For example, for P_4 this gives

```
1(2(34+43)+3(24+42)+4(23+32))+2(1(34+43)+3(14+41)+4(13+31))+
 3(1(24+42)+2(14+41)+4(12+21))+4(1(23+32)+2(13+31)+3(12+21)).
```

Can we do better?

Ellul et al. [7] proved the following weak lower bound: every regular expression for P_n has alphabetic length at least 2^{n-1}. A slightly stronger bound of $n2^{n-1}$ was also shown by Agrawal et al. [1]. In this note we derive an upper bound through divide-and-conquer. We then show that the regular expression this strategy produces is, in fact, actually optimal. This improves the results from [7, 1].

The language P_n is of particular interest because its complement has short regular expressions, as shown in [7]. For other results concerning context-free grammars for P_n, see [7, 2, 3, 8]. Cho et al. [5] considered the related problem of determining the size of the minimal DFA recognizing all permutations of a given finite language.

2 Divide-and-conquer

Consider the following divide-and-conquer strategy, as given by Agrawal et al. [1]. Let S be an alphabet of cardinality n. We consider all subsets $T \subseteq S$ of cardinality $\lfloor n/2 \rfloor$. For each subset we recursively determine a regular expression for the permutations of T, a regular expression for the permutations of $S - T$, and concatenate them together. This gives

$$E(S) = \sum_{\substack{T \subseteq S \\ |T| = \lfloor n/2 \rfloor}} (E(T))(E(S - T)); \quad E(i) = i. \tag{1}$$

Finally, we define $R_n = E(\Sigma_n)$.

Thus, for example, we get

```
R4 = (12+21)(34+43)+(13+31)(24+42)+(23+32)(14+41)+
     (14+41)(23+32)+(24+42)(13+31)+(34+43)(12+21)
```

for P_4.

The alphabetic length of the resulting regular expression R_n for all permutations of Σ_n is then $f(n)$, where

$$f(n) = \begin{cases} 1, & \text{if } n = 1; \\ \binom{n}{\lfloor n/2 \rfloor} \big(f(\lfloor n/2 \rfloor) + f(\lceil n/2 \rceil)\big), & \text{if } n > 1. \end{cases}$$

The first few values of $f(n)$ are given in the table below.

n	$f(n)$
1	1
2	4
3	15
4	48
5	190
6	600
7	2205
8	6720
9	29988
10	95760

It is sequence A320460 in the *On-Line Encyclopedia of Integer Sequences* [15].

It seems hard to determine a simple closed-form expression for $f(n)$. Agrawal et al. [1] show that $f(n) \leq 8 \cdot 4^n$. We can roughly estimate $f(n)$ as follows, at least when $n = 2^m$ is a power of 2:

$$\begin{aligned} f(2^m) &= 2\binom{2^m}{2^{m-1}} f(2^{m-1}) \\ &= 2^m \binom{2^m}{2^{m-1}} \binom{2^{m-1}}{2^{m-2}} \cdots \binom{2}{1} \\ &= 2^m \frac{(2^m)!}{(2^{m-1})!\,(2^{m-2})! \cdots 2!\,1!}. \end{aligned}$$

Substituting the Stirling approximation $n! \sim \sqrt{2\pi n}(n/e)^n$ and simplifying, we get that $f(2^m)$ is roughly equal to

$$4^{2^m} e^{-1} \pi^{(1-m)/2} 2^{-(m^2-5m+6)/4}.$$

To make this precise, and make it work when n is not a power of 2, however, takes more work.

The rest of the paper is organized as follows: in Section 3, we prove that our regular expression is in fact optimal, assuming one result that is proven at the end of the paper. In Section 4, we establish some inequalities related to Stirling's formula. In Section 5, we connect these inequalities to $f(n)$ and obtain the estimate mentioned in the abstract. Finally, in Section 6 we use our obtained bounds on $f(n)$ to provide the missing piece in our optimality proof.

3 Optimality

In order to show that our regular expression has minimum possible length, we use the following property of $f(n)$ that we prove in Section 6:

▶ **Lemma 1.** *If $n \geq 1$, then every integer $0 < k < n$ satisfies $\binom{n}{k}(f(k) + f(n-k)) \geq f(n)$. Equality occurs if and only if $k = \lfloor n/2 \rfloor$ or $k = \lceil n/2 \rceil$.*

For $n \geq 1$ and $1 \leq k \leq n!$, define $\ell(n, k)$ to be the minimum alphabetic length of a regular expression specifying a subset of P_n, where the subset has cardinality at least k.

▶ **Lemma 2.** *If $n \geq 1$ and $1 \leq k \leq n!$, then $\ell(n, k)/k \geq \ell(n, n!)/n! \geq f(n)/n!$.*

Proof. We prove this by induction over the lexicographical ordering of pairs (n, k). This is easy for our base case $n = k = 1$, as the best regular expression is a single character. We thus suppose $n \geq 2$.

Consider a regular expression for a subset of P_n of cardinality at least $k \geq 1$ that has minimum alphabetic length. Clearly no such expression will involve ϵ or $\emptyset$. We now consider the possibilities for the last (outermost) operation in the regular expression. Clearly the only relevant possibilities are union and concatenation.

If the last operation is a union, then it is the union of two subsets of P_n of cardinalities $k_1, k_2 \geq 1$ where $k_1 + k_2 \geq k$, and $k_1, k_2 < k$ by minimality. Then we get

$$\begin{aligned}\frac{\ell(n,k)}{k} &\geq \frac{\ell(n,k_1)+\ell(n,k_2)}{k_1+k_2} \\ &\geq \min\left\{\frac{\ell(n,k_1)}{k_1}, \frac{\ell(n,k_2)}{k_2}\right\} \\ &\geq \frac{\ell(n,n!)}{n!} \geq \frac{f(n)}{n!}.\end{aligned}$$

If the last operation is a concatenation, then it is the concatenation of two regular expressions for subsets of P_{n_1} and P_{n_2} of cardinalities k_1 and k_2 respectively (possibly after changing alphabets) where $n_1 + n_2 = n$ and $k_1 k_2 \geq k$. By minimality, we have n_1, n_2, k_1, k_2 all positive, so $n_1, n_2 < n$. We now obtain

$$\begin{aligned}\frac{\ell(n,k)}{k} &\geq \frac{\ell(n_1,k_1)+\ell(n_2,k_2)}{k_1 k_2} \\ &\geq \frac{\ell(n_1,n_1!)+\ell(n_2,k_2)}{n_1!\; k_2} \\ &\geq \frac{\ell(n_1,n_1!)+\ell(n_2,n_2!)}{n_1!\; n_2!} \\ &\geq \frac{f(n_1)+f(n_2)}{n_1!\; n_2!} \\ &= \frac{1}{n!}\binom{n}{n_1}(f(n_1)+f(n-n_1)) \\ &\geq \frac{1}{n!}\binom{n}{\lfloor n/2\rfloor}(f(\lfloor n/2\rfloor)+f(\lceil n/2\rceil)) \qquad \text{(by Lemma 1)} \\ &= \frac{f(n)}{n!}.\end{aligned}$$

In both cases, we get the desired inequalities for these choices of n and k, completing our induction. ◀

▶ **Theorem 3.** *Let $n \geq 1$. Over all regular expressions for the permutation language P_n, the regular expression R_n given by our divide-and-conquer strategy achieves the minimum alphabetic length.*

Proof. By our construction from Section 2, the regular expression R_n specifies the entirety of P_n and has alphabetic length $f(n)$. We thus get the upper bound $\ell(n, n!) \leq f(n)$. By Lemma 2, we have the matching lower bound $\ell(n, n!) \geq f(n)$. Thus, R_n has minimum possible alphabetic length for a regular expression specifying P_n. ◀

4 Analysis

In what follows we use ln to denote the natural logarithm, and lg to denote logarithms to the base 2.

Define $S : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ to be the usual Stirling approximation [14]:

$$S(x) = \sqrt{2\pi x}(x/e)^x.$$

▶ **Lemma 4.** *For every $x \geq 1$, we have the bounds*

$$S\big(x+\frac{1}{2}\big)^2 \leq S(x)S(x+1) \leq e^{1/(2x)}S\big(x+\frac{1}{2}\big)^2.$$

Proof. The first two derivatives of $\ln S(x)$ are

$$\frac{d}{dx}\ln S(x) = \frac{1}{2x} + \ln x$$

$$\frac{d^2}{dx^2}\ln S(x) = -\frac{1}{2x^2} + \frac{1}{x},$$

and so we see that $\ln S(x)$ is convex (that is, its derivative is increasing) for all $x > 1/2$. Thus, by Jensen's inequality [10] and exponentiating, we obtain the lower bound

$$S(x)S(x+1) \geq S\big(x+\frac{1}{2}\big)^2$$

for all $x \geq 1$.

Now, using the mean value theorem twice, we get that

$$(\ln S(x)) + \frac{1}{2}\mu \leq \ln S(x+\frac{1}{2}) \tag{2}$$

$$\ln S(x+1) \leq (\ln S(x+\frac{1}{2})) + \frac{1}{2}M, \tag{3}$$

where

$$\mu = \inf_{z\in[x,x+\frac{1}{2}]} (\ln S(z))' = \frac{1}{2x} + \ln x$$

$$M = \sup_{z\in[x+\frac{1}{2},x+1]} (\ln S(z))' = \frac{1}{2(x+1)} + \ln(x+1).$$

Adding the inequalities (2) and (3), we get

$$\begin{aligned}(\ln S(x)) + (\ln S(x+1)) &\leq (2\ln S(x+1/2)) - \mu/2 + M/2\\ &\leq (2\ln S(x+1/2)) + \frac{1}{2}\ln(\frac{x+1}{x})\\ &\leq (2\ln S(x+1/2)) + \frac{1}{2x}.\end{aligned}$$

This gives us the inequality

$$S(x)S(x+1) \leq e^{1/(2x)}S(x+1/2)^2$$

for all $x \geq 1$. ◀

Next, for $\alpha \in \mathbb{R}$, define the function $g_\alpha : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ by

$$g_\alpha(x) = \frac{4^x}{x^{(\lg x)/4}} x^\alpha.$$

Our goal is to show that f can be approximated by g_α for some choice of α.

▶ **Lemma 5.** *Let $\alpha > 0$. Then for every $x \geq 4^\alpha$ we have*

$$e^{-1/(2\sqrt{x})} \frac{5}{2} g_\alpha(x + \frac{1}{2}) \leq g_\alpha(x) + g_\alpha(x+1) \leq e^{1/(2\sqrt{x})} \frac{5}{2} g_\alpha(x + \frac{1}{2}).$$

Proof. We again compute the logarithmic derivative:

$$\frac{d}{dx} \ln g_\alpha(x) = \frac{\alpha - (\lg x)/2}{x} + \ln 4.$$

For $x \geq 4^\alpha$, this derivative is at most $\ln 4$, so by the mean value theorem,

$$\ln g_\alpha(x+1) - \ln 2 \leq \ln g_\alpha(x + \frac{1}{2}) \leq \ln g_\alpha(x) + \ln 2.$$

Exponentiating, we get

$$\frac{1}{2} g_\alpha(x+1) \leq g_\alpha(x + \frac{1}{2}) \leq 2 g_\alpha(x). \tag{4}$$

Next, we note that the derivative of $\ln x$ exceeds that of $\sqrt{x}$ for $0 < x < 4$ and is less for $x > 4$. So, since $\ln 4 < \sqrt{4}$, we have $\ln x < \sqrt{x}$ for all $x > 0$. Hence

$$\begin{aligned} \frac{d}{dx} \ln g_\alpha(x) &= \frac{\alpha}{x} - \frac{\ln x}{2x \ln 2} + \ln 4 \\ &\geq \ln 4 - \frac{\sqrt{x}}{(2 \ln 2) x} \\ &\geq \ln 4 - 1/\sqrt{x} \end{aligned}$$

for all $x > 0$. Thus, by the mean value theorem and exponentiating again, we get

$$\frac{1}{2} e^{1/(2\sqrt{x})} g_\alpha(x+1) \geq g_\alpha(x + \frac{1}{2}) \geq 2 e^{-1/(2\sqrt{x})} g_\alpha(x). \tag{5}$$

We can now combine the inequalities (4) and (5) for $x \geq 4^\alpha$ to get

$$\begin{aligned} e^{-1/(2\sqrt{x})} \frac{5}{2} g_\alpha(x + \frac{1}{2}) &\leq \frac{1}{2} g_\alpha(x + \frac{1}{2}) + 2 e^{-1/(2\sqrt{x})} g_\alpha(x + \frac{1}{2}) \\ &\leq g_\alpha(x) + g_\alpha(x+1) \\ &\leq \frac{1}{2} e^{1/(2\sqrt{x})} g_\alpha(x + \frac{1}{2}) + 2 g_\alpha(x + \frac{1}{2}) \\ &\leq e^{1/(2\sqrt{x})} \frac{5}{2} g_\alpha(x + \frac{1}{2}), \end{aligned}$$

which gives us both desired bounds. ◀

We now show an identity relating g_α and S.

▶ **Lemma 6.** *Suppose $x > 0$ and $\beta > 0$. If $\alpha = \lg \beta + 1/4 - (\lg \pi)/2$, then*

$$\beta \frac{S(2x)}{S(x)^2} g_\alpha(x) = g_\alpha(2x).$$

Proof. We have

$$\begin{aligned}
\beta\frac{S(2x)}{S(x)^2}g_\alpha(x) &= \beta\frac{\sqrt{4\pi x}(2x/e)^{2x}}{(\sqrt{2\pi x}(x/e)^x)^2}\frac{4^x}{x^{\lg x/4}}x^\alpha \\
&= \beta\frac{4^x}{\sqrt{\pi x}}\frac{4^x}{x^{\lg x/4}}x^\alpha \\
&= \frac{2^{\lg\beta+1/4}}{\pi^{1/2}x^{1/4}x^{1/4}2^{1/4}}\frac{4^{2x}}{x^{\lg x/4}}x^\alpha \\
&= 2^{\lg\beta+1/4-\lg\pi/2}\frac{4^{2x}}{2^{(1+\lg x)/4}x^{(1+\lg x)/4}}x^\alpha \\
&= \frac{4^{2x}}{(2x)^{(\lg 2x)/4}}(2x)^\alpha \\
&= g_\alpha(2x).
\end{aligned}$$

◀

5 Bounds on $f(n)$

In this section we obtain an estimate for $f(n)$, the size of the optimal regular expression for P_n.

▶ **Theorem 7.** *For all $n \geq 1$ we have*

$$0.195\frac{4^n}{n^{(\lg n)/4}}n^{5/4-(\lg\pi)/2} \leq f(n) \leq \frac{1}{4}\frac{4^n}{n^{(\lg n)/4}}n^{(\lg 5)-3/4-(\lg\pi)/2}.$$

Further, when n is a power of two, we get the following upper bound, matching the general lower bound.

$$f(n) \leq \frac{1}{4}\frac{4^n}{n^{\lg n/4}}n^{5/4-(\lg\pi)/2}.$$

Proof. Recall the Stirling approximation

$$e^{1/(12n+1)}S(n) \leq n! \leq e^{1/12n}S(n); \tag{6}$$

see [14]. Now suppose that $f(n) \leq r_n g_\alpha(n)$ and $f(n+1) \leq r_{n+1}g_\alpha(n+1)$, where $n \geq \max\{1, 4^\alpha\}$, for some non-decreasing function $r : \mathbb{N} \to \mathbb{R}_{>0}$. Then by combining Lemma 4, Lemma 5, and equation (6), we get

$$\begin{aligned}
f(2n+1) &= \binom{2n+1}{n}(f(n)+f(n+1)) \\
&\leq e^{\frac{1}{12(2n+1)}-\frac{1}{12n+1}-\frac{1}{12(n+1)+1}}\frac{S(2n+1)}{S(n)S(n+1)}(r_n g_\alpha(n)+r_{n+1}g_\alpha(n+1)) \\
&\leq \frac{5}{2}r_{n+1}e^{1/(2\sqrt{n})}\frac{S(2n+1)}{S(n+1/2)^2}g_\alpha(n+1/2)
\end{aligned}$$

and

$$\begin{aligned}
f(2n) &= \binom{2n}{n}(f(n)+f(n)) \\
&\leq 2r_n e^{\frac{1}{12(2n)}-2\frac{1}{12n+1}}\frac{S(2n)}{S(n)^2}g_\alpha(n) \\
&\leq 2r_n\frac{S(2n)}{S(n)^2}g_\alpha(n).
\end{aligned}$$

For the case where n is a power of two only, we use $\beta = 2$ and $r_n = C$, we set

$$\alpha = \lg\beta + 1/4 - (\lg\pi)/2 = 5/4 - (\lg\pi)/2,$$

so $\alpha > 0$ and $4^\alpha < 2$. Now Lemma 6 gives us the identity $2\frac{S(2x)}{S(x)^2} g_\alpha(x) = g_\alpha(2x)$. Then by induction we have $f(n) \le C g_\alpha(n)$ for all $n \ge 1$, where C is any constant that satisfies this bound for $n < 4$. In particular, $C = \frac{1}{4}$ works, so we have $f(n) \le \frac{1}{4} g_{5/4-\lg\pi/2}(n)$ for all $n \ge 1$ that are powers of two.

Next, for general n, we use $\beta = 5/2$ and $r_n = Ce^{-\frac{\sqrt{5}}{(4-\sqrt{10})\sqrt{n}}}$, we set

$$\alpha = \lg\beta + 1/4 - \lg\pi/2 = \lg 5 - 3/4 - \lg\pi/2,$$

so $\alpha > 0$ and $4^\alpha < 4$. Now Lemma 6 gives us the identity $\frac{5}{2}\frac{S(2x)}{S(x)^2} g_\alpha(x) = g_\alpha(2x)$. For $n \ge 4$, we get

$$\begin{aligned}
r_{n+1} e^{1/(2\sqrt{n})} &= Ce^{\frac{1}{2\sqrt{n}} - \frac{\sqrt{5}}{(4-\sqrt{10})\sqrt{n+1}}} \\
&\le C(e^{\frac{1}{\sqrt{n}}})^{\frac{1}{2} - \frac{\sqrt{5}}{4-\sqrt{10}}\frac{\sqrt{4}}{\sqrt{5}}} \\
&= C(e^{\frac{1}{\sqrt{n}}})^{-\frac{\sqrt{10}}{2(4-\sqrt{10})}} \\
&= Ce^{-\frac{\sqrt{5}}{(4-\sqrt{10})\sqrt{2n}}} \\
&\le r_{2n+1}.
\end{aligned}$$

Easily, we also get $2r_n \le \frac{5}{2} r_{2n}$. Thus, by induction we have $f(n) \le r_n g_\alpha(n)$ for all $n \ge 12$, where C is chosen to make this work for $12 \le n < 24$. In particular, $C = \frac{1}{4}$ works again. Further, since we have $r_n < C$ for all $n \ge 1$, we also have $f(n) \le \frac{1}{4} g_{\lg 5 - 3/4 - \lg\pi/2}(n)$ for all $n \ge 12$. Finally, we check manually that this last inequality holds for $1 \le n < 12$ too, and thus for all $n \ge 1$.

All that remains is the lower bound. We get similar recurrences, supposing $f(n) \ge r_n g_\alpha(n)$ and $f(n+1) \ge r_{n+1} g_\alpha(n+1)$, where $n \ge \max\{1, 4^\alpha\}$ for some non-increasing $r : \mathbb{N} \to \mathbb{R}_{>0}$. Then by a similar argument as for the upper bounds, we have

$$\begin{aligned}
f(2n+1) &= \binom{2n+1}{n}(f(n) + f(n+1)) \\
&\ge \frac{5}{2} r_{n+1} e^{\frac{1}{12(2n+1)+1} - \frac{1}{12n} - \frac{1}{12(n+1)}} e^{-1/(2\sqrt{n})} e^{-1/(2n)} \frac{S(2n+1)}{S(n+1/2)^2} g_\alpha(n+1/2) \\
&\ge \frac{5}{2} r_{n+1} e^{-1/(2\sqrt{n}) - 2/(3n)} \frac{S(2n+1)}{S(n+1/2)^2} g_\alpha(n+1/2)
\end{aligned}$$

and

$$\begin{aligned}
f(2n) &= \binom{2n}{n}(f(n) + f(n)) \\
&\ge 2 r_n e^{\frac{1}{24n+1} - \frac{2}{12n}} \frac{S(2n)}{S(n)^2} g_\alpha(n) \\
&\ge 2 r_n e^{-1/(6n)} \frac{S(2n)}{S(n)^2} g_\alpha(n).
\end{aligned}$$

This time, we set $\beta = 2$ with $r_n = Ce^{1/3n}$ (indeed non-increasing), and $\alpha = 5/4 - \lg\pi/2$, noting $4^\alpha < 4$. Now for $n = 16$, we have $\ln\frac{5}{4} \ge \frac{17}{96} = \frac{1}{2\sqrt{n}} + \frac{5}{6n}$. Since this right-hand side is non-increasing in n, we in fact have $\ln\frac{5}{4} \ge \frac{1}{2\sqrt{n}} + \frac{5}{6n}$ for all $n \ge 17$ too. This implies

$$\begin{aligned}
\frac{5}{2} r_{n+1} e^{-1/(2\sqrt{n}) - 2/(3n)} &= 2Ce^{\ln\frac{5}{4} + \frac{1}{3(n+1)} - \frac{1}{2\sqrt{n}} - \frac{5}{6n} + \frac{1}{6n}} \\
&\ge 2Ce^{\frac{1}{3(n+1)} + \frac{1}{6n}} \\
&\ge 2r_{2n+1}.
\end{aligned}$$

Further, $2r_n e^{-1/6n} = 2r_{2n}$, so by induction we have $f(n) \geq r_n g_\alpha(n)$ for all $n \geq 17$, where C is chosen to satisfy this bound for $17 \leq n < 34$. In particular, $C = 0.195$ works. Since $r_n > C$ for all $n \geq 17$, we also have $f(n) \geq 0.195 g_{5/4-\lg \pi/2}(n)$ for all $n \geq 17$. Finally, we check manually that this works for all $1 \leq n < 17$ too, and thus for all $n \geq 1$. ◀

6 Optimality revisited

We now give a simple lower bound on the growth of f.

▶ **Lemma 8.** *We have $f(n+1) \geq 3f(n)$ for all $n \geq 1$.*

Proof. We prove this by induction on n. It is easy to verify the base case $f(2) = 4 \geq 3 = 3f(1)$. Otherwise $n > 1$. Suppose the desired inequality holds for all smaller values of n. If $n \geq 2$ is odd, then let $1 \leq m < n$ satisfy $2m + 1 = n$. Then

$$\begin{aligned}
f(n+1) &= f(2m+2) \\
&= 2\binom{2m+2}{m+1} f(m+1) \\
&= 2\frac{2m+2}{m+1}\binom{2m+1}{m} f(m+1) \\
&= \binom{2m+1}{m}(f(m+1) + 3f(m+1)) \\
&\geq \binom{2m+1}{m}(3f(m) + 3f(m+1)) \\
&= 3f(2m+1) \\
&= 3f(n).
\end{aligned}$$

Otherwise, if $n \geq 2$ is even, then let $1 \leq m < n$ satisfy $2m = n$. We note that $4m+2 \geq 3m+3$, so $2\frac{2m+1}{m+1} \geq 3$. We then have

$$\begin{aligned}
f(n+1) &= f(2m+1) \\
&= \binom{2m+1}{m}(f(m) + f(m+1)) \\
&= \frac{2m+1}{m+1}\binom{2m}{m}(f(m) + f(m+1)) \\
&\geq \frac{2m+1}{m+1}\binom{2m}{m}(f(m) + 3f(m)) \\
&= 2\frac{2m+1}{m+1} f(2m) \\
&\geq 3f(2m) \\
&= 3f(n).
\end{aligned}$$

◀

Armed with this inequality and the bounds given by Theorem 7 of Section 5, we are ready to complete our proof of the optimality of R_n. We recall Lemma 1, which is what we have left to show:

▶ **Lemma 1.** *If $n \geq 1$, then every integer $0 < k < n$ satisfies $\binom{n}{k}(f(k) + f(n-k)) \geq f(n)$. Equality occurs if and only if $k = \lfloor n/2 \rfloor$ or $k = \lceil n/2 \rceil$.*

Proof. We easily check the cases $n < 12$ by hand. Let $n \geq 12$ be arbitrary. It suffices to consider the cases where $k \leq \lfloor n/2 \rfloor$, as those where $k \geq \lceil n/2 \rceil$ are symmetric. Equality for the case $k = \lfloor n/2 \rfloor$ is given by the definition of $f(n)$.

Suppose that $n/6 \leq k < \lfloor n/2 \rfloor$. Then $n > 9$, so $3n - 3 > 2n + 6$. Hence we get

$$\frac{n/2 - 1/2}{n/2 + 3/2} > 2/3$$

and so

$$\frac{\lfloor n/2 \rfloor}{\lceil n/2 \rceil + 1} > 2/3. \tag{7}$$

Also, from $k \geq n/6$ we get $3k + 3 \geq \lceil n/2 \rceil + 2$, and so

$$\frac{k+1}{\lceil n/2 \rceil + 2} \geq 1/3. \tag{8}$$

Then

$$\begin{aligned}
&\binom{n}{k}(f(k) + f(n-k)) \\
&\geq \binom{n}{k} f(n-k) \\
&\geq \binom{n}{k} 3^{\lfloor n/2 \rfloor - k} f(\lceil n/2 \rceil) && \text{(by Lemma 8)} \\
&\geq 3^{\lfloor n/2 \rfloor - k} \binom{n}{k} \frac{f(\lfloor n/2 \rfloor) + f(\lceil n/2 \rceil)}{2} \\
&= \frac{3^{\lfloor n/2 \rfloor - k}}{2} \prod_{k \leq j < \lfloor n/2 \rfloor} \frac{j+1}{n-j} \binom{n}{\lfloor n/2 \rfloor} (f(\lfloor n/2 \rfloor) + f(\lceil n/2 \rceil)) \\
&= \frac{3^{\lfloor n/2 \rfloor - k}}{2} \frac{\prod_{k < j \leq \lfloor n/2 \rfloor} j}{\prod_{\lceil n/2 \rceil < j \leq n-k} j} f(n) \\
&= \frac{3^{\lfloor n/2 \rfloor - k}}{2} \frac{\lfloor n/2 \rfloor}{\lceil n/2 \rceil + 1} \frac{\prod_{k+1 \leq j \leq \lfloor n/2 \rfloor - 1} j}{\prod_{\lceil n/2 \rceil + 2 \leq j \leq n-k} j} f(n) \\
&> \frac{3^{\lfloor n/2 \rfloor - k}}{2} (2/3) \prod_{1 \leq j \leq \lfloor n/2 \rfloor - k - 1} \frac{k+j}{\lceil n/2 \rceil + 1 + j} f(n) && \text{(by (7))} \\
&\geq 3^{\lfloor n/2 \rfloor - k - 1} \left(\frac{k+1}{\lceil n/2 \rceil + 2} \right)^{\lfloor n/2 \rfloor - k - 1} f(n) \\
&\geq 3^{\lfloor n/2 \rfloor - k - 1} (1/3)^{\lfloor n/2 \rfloor - k - 1} f(n) && \text{(by (8))} \\
&= f(n).
\end{aligned}$$

Next, suppose $1 \leq k < n/6$. Then $4k < n - n/3$ and so $4 < \frac{n-1}{k}$. Then if $k > 1$,

$$\begin{aligned}
\binom{n}{k} &= n \prod_{2 \leq j \leq k} \frac{n - 1 - k + j}{j} \\
&\geq n \left(\frac{n-1}{k} \right)^{k-1} \\
&\geq n 4^{k-1}.
\end{aligned}$$

We note also that $\binom{n}{1} = n4^0$, so we have $\binom{n}{k} \geq n4^{k-1}$ for all $1 \leq k < n/6$. We note from our proof of Lemma 5 that the derivative of $\ln g_\alpha(x)$ is at most $\ln 4$ for $x \geq 4^\alpha$. In particular, for $\alpha = 5/4 - \lg \pi/2$, this derivative is at most $\ln 4$ for all $x \geq 2$, and so $4^k g_\alpha(n-k) \geq g_\alpha(n)$ here (as $n - k > 5n/6 \geq 10$). Thus

$$\begin{aligned}
&\binom{n}{k}(f(k) + f(n-k)) \\
&\geq \binom{n}{k} f(n-k) \\
&\geq n4^{k-1}\left(0.195 g_{5/4-\lg\pi/2}(n-k)\right) && \text{(by Theorem 7)} \\
&\geq n4^{k-1}\left(0.195 \frac{g_{5/4-\lg\pi/2}(n)}{4^k}\right) \\
&= 0.195 n \frac{1}{4} g_{5/4-\lg\pi/2}(n) \\
&= 0.195 n \frac{1}{4} n^{2-\lg 5} g_{\lg 5 - 3/4 - \lg\pi/2}(n) \\
&\geq 0.195 n^{3-\lg 5} f(n) && \text{(by Theorem 7)} \\
&> f(n) && \text{(since } n \geq 12 > 0.195^{-\frac{1}{3-\lg 5}}\text{).}
\end{aligned}$$

◀

References

1. Rakesh Agrawal, Giuseppe Psaila, Edward L. Wimmers, and Mohamed Zait. Querying Shapes of Histories. Technical Report RJ 9962 (87921), IBM Almaden Research Center, June 26 1995.
2. P. R. J. Asveld. Generating all permutations by context-free grammars in Chomsky normal form. *Theoret. Comput. Sci.*, 354:118–130, 2006.
3. P. R. J. Asveld. Generating all permutations by context-free grammars in Greibach normal form. *Theoret. Comput. Sci.*, 409:565–577, 2008.
4. D. Chistikov, S. Ivan, A. Lubiw, and J. Shallit. Fractional Coverings, Greedy Coverings, and Rectifier Networks. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science (STACS 2017)*, volume 66 of *LIPIcs*, pages 23:1–23:14. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2017.
5. Da Jung Cho, Daniel Goč, Yo-Sub Han, Sang Ki Ko, Alexandros Palioudakis, and Kai Salomaa. State complexity of permutation on finite languages over a binary alphabet. *Theoretical Computer Science*, 682:67–78, June 2017. doi:10.1016/j.tcs.2017.03.007.
6. A. Ehrenfeucht and P. Zeiger. Complexity measures for regular expressions. *J. Comput. System Sci.*, 12:134–146, 1976.
7. K. Ellul, B. Krawetz, J. Shallit, and M.-w. Wang. Regular expressions: new results and open problems. *J. Autom. Lang. Combin.*, 10:407–437, 2005.
8. Y. Filmus. Lower bounds for context-free grammars. *Info. Proc. Letters*, 111:895–898, 2011.
9. G. Gramlich and G. Schnitger. Minimizing nfa's and regular expressions. *J. Comput. System Sci.*, 73:908–923, 2007.
10. J. L. W. V. Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Math.*, 30:175–193, 1906. doi:10.1007/BF02418571.
11. T. Jiang and B. Ravikumar. Minimal NFA problems are hard. *SIAM J. Comput.*, 22:1117–1141, 1993.
12. A. R. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proc. 13th Ann. IEEE Symp. on Switching and Automata Theory*, pages 125–129. IEEE, 1972.

13 H. Mousavi. Lower bounds on regular expression size. Preprint available at `arXiv:1712.00811`, 2017.

14 H. Robbins. A Remark on Stirling's Formula. *Amer. Math. Monthly*, 62:26–29, 1955.

15 N. J. A. Sloane et al. The On-Line Encyclopedia of Integer Sequences. Available at `https://oeis.org`, 2018.

Equivalence of Finite-Valued Streaming String Transducers Is Decidable

Anca Muscholl
LaBRI, University of Bordeaux, France

Gabriele Puppis
CNRS, LaBRI, Bordeaux, France

Abstract

In this paper we provide a positive answer to a question left open by Alur and and Deshmukh in 2011 by showing that equivalence of finite-valued copyless streaming string transducers is decidable.

2012 ACM Subject Classification Theory of computation → Transducers

Keywords and phrases String transducers, equivalence, Ehrenfeucht conjecture

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.122

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version A full version of the paper is available at `https://arxiv.org/abs/1902.06973`.

Funding DeLTA project (ANR-16-CE40-0007)

1 Introduction

Finite transducers are simple devices that allow to reason about data transformations in an effective, and even efficient way. In their most basic form they transform strings using finite control. Unlike automata, their power heavily depends on various parameters, like non-determinism, the capability of scanning the input several times, or the kind of storage they may use. The oldest transducer model, known as generalized sequential machine, extends finite automata by outputs. Inspired by an approach that applies to arbitrary relational structures [9], logic-based transformations (also called transductions) were considered by Engelfriet and Hoogeboom [12]. They showed that two-way transducers and monadic-second order (MSO) definable transductions are equivalent in the deterministic case (and even if the transduction is single-valued, which is more general than determinism). This equivalence supports thus the notion of "regular" functions, in the spirit of classical results on regular word languages from automata theory and logics due to Büchi, Elgot, Trakhtenbrot, Rabin, and others. A one-way transducer model that uses write-only registers as additional storage was proposed a few years ago by Alur and Cerný [2], and called streaming string transducer (SST). SST were shown equivalent to two-way transducers and MSO definable transductions in the deterministic setting, and again, even in the single-valued case.

In the relational case the picture is less satisfactory, as expressive equivalence is only preserved for SST and non-deterministic MSO transductions [5], which extend the original MSO transductions by existentially quantified monadic parameters. On the other hand, two-way transducers and SST are incomparable in the relational case. Between functions and relations there is however one class of transductions that exhibits a better behavior, and this is the class of finite-valued transductions. Being finite-valued means that there exists some constant k such that every input belonging to the domain has at most k outputs.

Finite-valued transductions were intensively studied in the setting of one-way and two-way transducers. For one-way transducers, k-valuedness can be checked in PTIME [17]. In addition, every k-valued one-way transducer can be effectively decomposed into a union of k

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 122; pp. 122:1–122:15


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

unambiguous one-way transducers of exponential size [24, 23]. For both two-way transducers and SST, checking k-valuedness is in PSPACE.

Besides expressiveness, another fundamental question concerning transducers is the equivalence problem, that is, the problem of deciding whether two transducers define the same relation (or the same partial function if we consider the single-valued case). The equivalence problem turns out to be PSPACE-complete for deterministic two-way transducers [16], single-valued two-way transducers, as well as for single-valued SST [5]. For deterministic SST, equivalence is in PSPACE [3], but it is open whether this complexity upper bound is optimal. For arbitrary SST, and in fact even for non-deterministic one-way transducers over a unary output alphabet, equivalence is undecidable [13, 18]. The equivalence problem for k-valued one-way transducers was shown to be decidable by Culik and Karhumäki using an elegant argument based on Ehrenfeucht's conjecture [10], and the authors noted that the same proof goes through for two-way transducers as well. The decidability status for the equivalence problem for k-valued SST was first stated as an open problem in [5]. Another open problem is whether SST and two-way transducers are equivalent in the finite-valued case, like in the single-valued case. It is worth noting, however, that in the full relational case SST and two-way transducers are incomparable. Concerning this last open question, a partial positive answer was given in [14], by decomposing any finite-valued SST with only one register into a finite union of unambiguous SST. This decomposition result also entails the decidability of the equivalence problem for the considered class.

The main result of this paper is a positive answer to the first question left open in [5]:

▶ **Theorem 1.** *The equivalence problem for finite-valued SST is decidable.*

We show the above result with a proof idea due to Culik and Karhumäki [10], based on the Ehrenfeucht conjecture. Our proof is much more involved, because SST produce their outputs piece-wise, in contrast to one-way and two-way transducers, that produce output linearly while reading the input. We manage to overcome this obstacle using some (mild) word combinatorics and word equations, by introducing a suitable normalization procedure for SST. We believe that our technique will also allow to solve the second problem left open in [5], which is the expressive equivalence between finite-valued SST and two-way transducers.

Related work

The equivalence problem for transducers has recently raised interest for more complex types of transducers in the single-valued case: Filiot and Reynier showed that equivalence of copyful, deterministic SST is decidable by showing them equivalent to HDT0L systems and applying [10], which contains the above-mentioned result as a special case. Subsequently, Benedikt et al. showed that equivalence of copyful, deterministic SST has Ackerman complexity, with a proof based on polynomial automata and ultimately on Hilbert's basis theorem [6]. Interestingly, the use of Hilbert's basis theorem goes back to the proof of Ehrenfeucht's conjecture [1, 15]. A similar approach was used by Boiret et al. in [7] to show that bottom-up register automata over unordered forests have a decidable equivalence problem, see also the nice survey [8].

Overview

Section 2 introduces the transducer model, then Section 3 sets up the technical machinery that allows to normalize finite-valued SST. Section 4 shows the major normalization result, which holds for left quotients of SST. Finally Section 5 recalls the Ehrenfeucht-based proof for equivalence and the application to finite-valued SST. A full version of the paper is available at `https://arxiv.org/abs/1902.06973`.

2 Streaming string transducers

A *streaming string transducer* (*SST*) is a tuple $T = (\Sigma, \Gamma, X, Q, U, I, E, F, x_{\text{out}})$, where Σ and Γ are finite input and output alphabets, X is a finite set of registers (usually denoted x, x', x_1, x_2, etc.), Q is a finite set of states, U is a finite set of register updates, that is, functions from X to $(X \uplus \Gamma)^*$, $I, F \subseteq Q$ are subsets of states, defining the initial and final states, $E \subseteq Q \times \Sigma \times U \times Q$ is a transition relation, describing, for each state and input symbol, the possible register updates and target states, and finally $x_{\text{out}} \in X$ is a register for the output. Note that, compared to the original definition from [2], here we forbid for simplicity the use of final production rules, that perform ad additional register update after the end of the input. This simplification is immaterial with respect to the decidability of the equivalence problem. For example, it can be enforced, without loss of generality, by assuming that all well-formed inputs are terminated by a special marker, say $\dashv$, on which the transducer can apply a specific transition. We assume here that *all inputs of a transducer are non-empty and of the form $u \dashv$, with $\dashv$ not occurring in u.*

Copyless restriction and capacity

An SST as above is *copyless* if for all register updates $f \in U$, every register $x \in X$ appears at most once in the word $f(x_1) \dots f(x_m)$, where $X = \{x_1, \dots, x_m\}$. For a copyless SST, every output has length at most linear in the length of the input. More precisely, every output associated with an input u has length at most $c|u|$, where $c = \max_{f \in U} \sum_{x \in X} |f(x)|_\Gamma$ is the maximum number of letters that the SST can add to its registers along a single transition (this number c is called *capacity* of the SST).

Hereafter, we assume that all SST are copyless.

Register updates and flows

Every register update, and in general every function $f : X \to (X \uplus \Gamma)^*$ is naturally extended to a morphism on $(X \uplus \Gamma)^*$, by defining it as identity over Γ. When reasoning with register updates, it is sometimes possible to abstract away the specific words over Γ, and only consider how the contents of the registers flows into other registers. Formally, the *flow* of an update $f : X \to (X \uplus \Gamma)^*$ is the bipartite graph that consists of two ordered sequences of nodes, one on the left and one on the right, with each node in a sequence corresponding to a specific register, and arrows that go from the node corresponding to register x to a right node corresponding to register x whenever x occurs in $f(x)$. For example, the flow of the update f defined by $f(x_1) = a\, x_1\, aa\, x_3$, $f(x_2) = b\, a$, and $f(x_3) = x_2\, b$ is the second bipartite graph in the figure on page 5.

Note that there are finitely many flows on a fixed number of registers. Moreover, flows can be equipped with a natural composition operation: given two flows F_1 and F_2, $F_1 \cdot F_2$ is the bipartite graph obtained by glueing the right nodes of F_1 with the left nodes of F_2, and by shortcutting pairs of consecutive arrows. We call *flow monoid* of an SST T the monoid of flows generated by the updates of T, with the composition operation as associative product.

Transitions, runs, and loops

A transition (q, a, f, q') of an SST T is conveniently denoted by the arrow $q \xrightarrow[T]{a/f} q'$, and the subscript T is often omitted when clear from the context. A *run* on $w = a_1 \dots a_n$ is a sequence of transitions of the form $q_0 \xrightarrow{a_1/f_1} q_1 \xrightarrow{a_2/f_2} \dots \xrightarrow{a_n/f_n} q_n$. Sometimes, a run as

above is equally denoted by $q_0 \xrightarrow{w/f} q_n$, so as to highlight the underlying input w and the induced register update $f = f_1 \circ \cdots \circ f_n$. A run is *initial* (resp. *final*) if it begins with an initial (resp. final) state; it is *successful* if it is both initial and final.

Given two registers x, x' and a run $\rho : q \xrightarrow{w/f} q'$, we say that x *flows into* x' *along* ρ if x occurs in $f(x')$. Note that this property depends only on the flow of the induced update f.

An SST is said to be *trimmed* is every state occurs in at least one successful run, so every state is reachable from the initial states and co-reachable from the final states. This property can be easily enforced with a polynomial-time preprocessing.

When reasoning with automata, it is common practice to use pumping arguments. Pumping will also be used here, but the notion of loop needs to be refined as to take into account the effect of register updates. Formally, a *loop* of a run ρ of an SST is any non-empty factor of ρ of the form $\gamma : q \xrightarrow{w/f} q$, that starts and ends in the same state q, and induces a *flow-idempotent* update, namely, an update f such that f and $f \circ f$ have the same flow.

Outputs and finite-valuedness

The *output* of a successful run $\rho : q_0 \xrightarrow{w/f} q_n$ is defined as $\mathrm{out}(\rho) = (f_0 \circ f)(x_{\mathrm{out}})$, where $f_0(x) = \varepsilon$ for all $x \in X$. Sometimes, we write $\mathrm{out}(f)$ in place of $\mathrm{out}(\rho)$. The *relation realized by an SST* is the set of pairs $(u, v) \in \Sigma^* \times \Gamma^*$, where u is a well-formed input (namely, terminating with $\dashv$) and v is the output associated with some successful run on u. An SST is *k-valued* if for every input u, there are at most k different outputs associated with u. It is *single-valued* (resp. *finite-valued*) if it is k-valued for $k = 1$ (resp. for some $k \in \mathbb{N}$). The domain an SST T, denoted $\mathrm{Dom}(T)$, is the set of input words that have some successful run in T. Two SST T_1, T_2 are *equivalent*, denoted as $T_1 \equiv T_2$, if they realize the same relation over $\Sigma^* \times \Gamma^*$.

Register valuations

A *register valuation* is a function from X to Γ^*. Given a successful run $\rho : q_0 \xrightarrow{a_1/f_1} q_1 \xrightarrow{a_2/f_2} \ldots \xrightarrow{a_n/f_n} q_n$ and a position $i \in \{0, \ldots, n\}$ in it, *the register valuation at position i in ρ* is the function $\mathrm{val}_{\rho,i}$ that is defined inductively on i as follows: $\mathrm{val}_{\rho,0}(x) = \varepsilon$, for all $x \in X$, and $\mathrm{val}_{\rho,i+1} = \mathrm{val}_{\rho,i} \circ f_i$. Note that $\mathrm{val}_{\rho,n}(x_{\mathrm{out}})$ coincides with the final output $\mathrm{out}(f_1 \circ \cdots \circ f_n)$ produced by ρ.

3 Normalizations

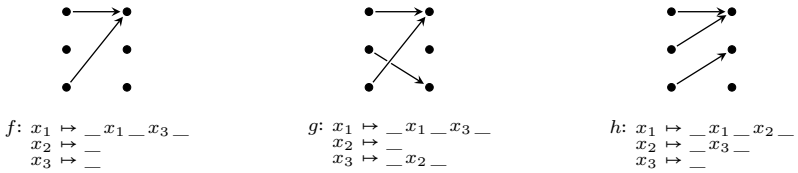
A major stumbling block in deciding equivalence of SST, as well as other crucial problems, lies in the fact that the same output can be produced by very different runs. This phenomenon already appears with much simpler transducers, e.g. with one-way transducers, where runs may produce the same output, but at different speeds. However, the phenomenon is more subtle for SST, as the output is produced piece-wise, and not sequentially: runs with same output may appear to be different in many ways, e.g. in terms of the flows of the register updates, or in terms of shifts of portions of the output. The goal of this section is to provide suitable normalization steps that remove, one at a time, the above mentioned degrees of freedom in producing the same output.

Another issue that we will be concerned with is the compatibility of the normalization steps with constructions on transducers that shortcut arbitrary long runs into a single transition. Essentially, we aim at having an effective notion of equivalence w.r.t. final outputs that works not only for transitions but also for runs.

Normalization of flows

In this section, *m will always denote the number of registers of an SST and $X = \{x_1, \dots, x_m\}$ the set of registers.* It is convenient to equip X with a total order, say $x_1 < \cdots < x_m$. Accordingly, we let $\chi = x_1 \dots x_m$ be the juxtaposition of all register names, and $f(\chi) = f(x_1) \dots f(x_m)$ for every register update f.

We say that a register update f is *non-erasing* if for every register x, $f(\chi)$ contains at least an occurrence of x (in fact, exactly one, since T is copyless). This can be rephrased as a property of the flow of f, where every node on the left must have an outgoing arrow. In a similar way, we say that f is *non-permuting* if registers appear in $f(\chi)$ with their natural order and without jumps, that is, $f(\chi) \in \Gamma^* x_1 \Gamma^* \dots \Gamma^* x_k \Gamma^*$, for some $k \le m$. As before, this can be rephrased by saying that the arrows in the flow of f must not be crossing, and the target nodes to the right must form a prefix of χ. Below are some examples of updates with their flows: the first update f is erasing, the second update g is non-erasing but permuting, and the third update h is non-erasing and non-permuting.



We say that T is *flow-normalized* if all its register updates are non-erasing and non-permuting. Note that a flow-normalized SST with m registers can have at most 2^m different flows.

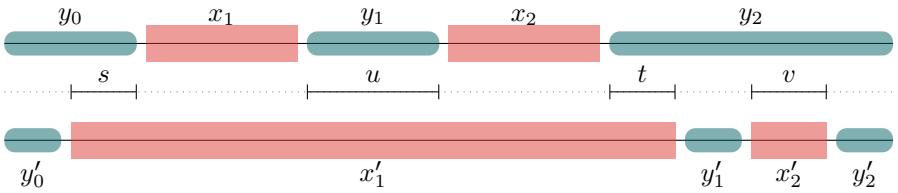
▶ **Proposition 2.** *One can transform any SST into an equivalent flow-normalized one.*

Recall that a register valuation is a function from X to Γ^*. With a flow-normalized SST, one can also define a dual notion of valuation, representing "gaps" between registers that shrink along the run. For this we introduce $m+1$ fresh variables $y_0, y_1, \dots, y_m$, called *gaps*. Hereafter, *$Y = \{y_0, y_1, \dots, y_m\}$ will always denote the set of gaps.* We use the term *valuation* to generically denote a register/gap valuation, that is, a function from $X \uplus Y$ to Γ^*.

The idea is that a gap y_j represents a word that is inserted between register x_j (if $j > 0$) and register x_{j+1} (if $j < n$) so as to form the final output. Formally, given a word $w \in \Gamma^* x_1 \Gamma^* \dots \Gamma^* x_k \Gamma^*$, with $k \le m$, and given two registers x_i, x_j, with $i < j$, we denote by $w\langle x_i, x_j\rangle$ the maximal factor of w strictly between the unique occurrence of x_i and the unique occurrence of x_j, using the following conventions for the degenerate cases: if $i = 0$, then $w\langle x_i, x_j\rangle$ is a maximal prefix of w; if $i > 0$ but there is no occurrence of x_i, then $w\langle x_i, x_j\rangle = \varepsilon$; finally, if there is an occurrence of x_i but no occurrence of x_j in w, then $w\langle x_i, x_j\rangle$ is a maximal suffix. Given a run $\rho : q_0 \xrightarrow{a_1/f_1} q_1 \xrightarrow{a_2/f_2} \dots \xrightarrow{a_n/f_n} q_n$ and a position i in it, *the valuation at position i of ρ* is the function $\mathrm{val}_{\rho,i} : X \uplus Y \to \Gamma^*$ such that

- $\mathrm{val}_{\rho,i}$ restricted to X is the register valuation at position i of ρ,
- $\mathrm{val}_{\rho,i}$ maps every gap y_j to the word $(f_{i+1} \circ \cdots \circ f_n)(\chi)\langle x_j, x_{j+1}\rangle$.

By definition, the image of the word $\zeta = y_0\, x_1\, y_1 \dots x_m\, y_m$ via the valuation $\mathrm{val}_{\rho,i}$ is always equal to the final output $\mathrm{out}(\rho)$, for all positions i. In this sense, the sequence of valuations $\mathrm{val}_{\rho,0}, \mathrm{val}_{\rho,1}, \dots, \mathrm{val}_{\rho,n}$ can be identified with a sequence of factorizations of $\mathrm{out}(\rho)$. For example, below are the factorizations of the output before and after a transition with register update f such that $f(x_1) = s\, x_1\, u\, x_2\, t$ and $f(x_2) = v$, for $s, u, t, v \in \Gamma^*$:



This also suggests the principle that gaps, like registers, are updated along transitions via suitable morphisms, but in a symmetric way, that is, from right to left. For instance, in the above picture, the gaps y_0, y_1, y_2 are updated by the function $f^\star$ such that $f^\star(y_0) = y_0\, s$, $f^\star(y_1) = u$, and $f^\star(y_2) = t\, y_1\, v\, y_2$. In general, the function $f^\star$, called *gap update*, is uniquely determined by the register update f, and vice versa, f is uniquely determined by the gap update $f^\star$. Another perhaps interesting phenomenon is that the gap update $f^\star$ is also non-erasing and non-permuting (the notion of non-permuting gap assignment is defined w.r.t. the reverse order $y_m < \cdots < y_0$).

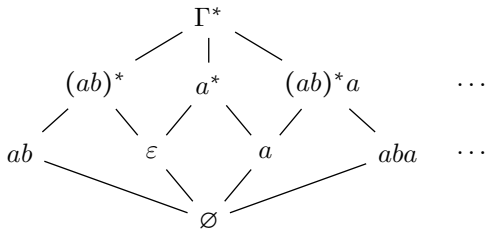
Normalization of states

The next normalization step splits the states of an SST in such a way that it becomes possible to associate with each state an over-approximation of the possible register/gap valuations witnessed when the state is visited along a successful run. These over-approximations are very simple languages over the output alphabet Γ, e.g. singleton languages like $\{aba\}$ and periodic languages like $\{ab\}^*\{a\}$ (often denoted $(ab)^*a$ to improve readability). Basically our over-approximations refer to length and period constraints. The *period* of a word w is the least number $0 < p \leq |w|$ such that w is a prefix of $(w[1,p])^\omega$. For example, the period of $w = abcab$ is 3.

For a given parameter $\alpha \in \mathbb{N}$ we define the family $\mathcal{L}_\alpha$ that contains:

- the empty language $\varnothing$,
- the singleton languages $\{u\}$, with $u \in \Gamma^*$ and $|u| \leq \alpha$,
- the periodic languages u^*v, with $u \in \Gamma^+$ *primitive* (i.e. $u = w^k$ only if $k = 1$), $|u| \leq \alpha$, and $v \in \Gamma^*$ strict prefix of u,
- the universal language Γ^*.

The languages in $\mathcal{L}_\alpha$, partially ordered by containment, form a finite meet semi-lattice, where the meet is the intersection $\cap$. We depict here part of the lattice $\mathcal{L}_\alpha$ for a parameter $\alpha \geq 3$:



The semi-lattice structure allows to derive a best over-approximation in $\mathcal{L}_\alpha$ of any language $L \subseteq \Gamma^*$, that is: $L^{\uparrow\alpha} = \bigcap\{L' \in \mathcal{L}_\alpha : L' \supseteq L\}$. We will mostly use the approximation operator $^{\uparrow\alpha}$ on singleton languages. For example, for $\alpha = 3$, we have $\{aba\}^{\uparrow\alpha} = \{aba\}$, $\{ababa\}^{\uparrow\alpha} = (ab)^*a$, and $\{abbb\}^{\uparrow\alpha} = \Gamma^*$. Note also that if $|w| \leq \alpha$ then $w^{\uparrow\alpha} = \{w\}$. A useful property is the compatibility of $^{\uparrow\alpha}$ with concatenation, which immediately extends to compatibility with word morphisms:

▶ **Lemma 3.** $(L_1 \cdot L_2)^{\uparrow\alpha} = (L_1{}^{\uparrow\alpha} \cdot L_2{}^{\uparrow\alpha})^{\uparrow\alpha}$ *for every* $\alpha \in \mathbb{N}$ *and* $L_1, L_2 \subseteq \Gamma^*$.

Recall that X, Y denote, respectively, the sets of registers and gaps of a flow-normalized SST. Given a valuation $\nu : X \uplus Y \to \Gamma^*$, its *$\alpha$-approximant* is the function $\nu^{\uparrow\alpha} : X \uplus Y \to \mathcal{L}_\alpha$ that maps any $z \in X \uplus Y$ to the language $\{\nu(z)\}^{\uparrow\alpha}$. The set of α-approximants is denoted $\mathcal{L}_\alpha^{X \uplus Y}$, and consists of all maps from $X \uplus Y$ to $\mathcal{L}_\alpha$. Further let

$$\mathrm{Val}_q = \{\mathrm{val}_{\rho,i} \,:\, \rho \text{ successful run visiting } q \text{ at any position } i\}$$

be the set of possible valuations induced by an arbitrary successful run when visiting state q.

A first desirable property is that all valuations in Val_q have the *same* α-approximant, which is thus determined by the state q. Formally, given a flow-normalized SST T with trimmed state space Q, we say that T *admits α-approximants* if every state $q \in Q$ can be effectively annotated with an α-approximant $A_q \in \mathcal{L}_\alpha^{X \uplus Y}$ in such a way that

$$\forall \nu \in \mathrm{Val}_q : \qquad \nu^{\uparrow\alpha} = A_q. \tag{1}$$

This condition is best understood as an invariant on lengths and periods that can be enforced on valuations of registers and gaps when visiting a particular state. For instance, when the approximant A_q guarantees a certain period, then this period will be the same for all valuations occurring at q, independently of the specific initial run that may lead to q (for registers), and of the run that may lead from q to an accepting state (for gaps).

The proposition below shows that it is always possible to refine any SST T so as to admit α-approximants, for any parameter α. The proof for α-approximants that concern only registers could be understood as unfolding the SST T, and merging nodes corresponding to any two inputs u and v, with u prefix of v, whenever the induced α-approximants at u and v are the same for every register x. In general, the resulting SST can be seen a covering of the original SST T, in the sense formalized by Sakarovitch and de Souza in [22]: T' is a *covering* of T if the states of T' can be mapped homomorphically to states of T, while preserving transitions and the distinction into initial and final states, and, moreover, the outgoing transitions of every state of T' map one-to-one to outgoing transitions of a corresponding state of T. This implies that the successful runs of T and those of T' are in one-to-one correspondence.

▶ **Proposition 4.** *Let T be a flow-normalized SST, and let $\alpha \in \mathbb{N}$. One can construct an equivalent flow-normalized SST T' that admits α-approximants and that is a covering of T.*

Notation. Whenever an SST admits α-approximants A_q as above, it is convenient to denote its states by triples of the form (q, A_X, A_Y), where A_X (resp. A_Y) is the restriction of the α-approximant A_q of state q to registers (resp. gaps).

Note that the smaller the parameter α, the weaker is the property required for α-approximants (in particular, for $\alpha = 0$ the lattice $\mathcal{L}_\alpha$ collapses to $\varnothing$ and Γ^*). Choosing α to be at least the capacity of the SST is already a reasonable choice, as it gives a nice characterization of equivalence of transitions w.r.t. the produced outputs (cf. Lemma 5 below). However, we will see that it is desirable to have even finer approximants, in such a way that our results will be compatible with left quotients of SST, that shortcut arbitrary long runs into single transitions. We postpone the technical details to Section 4, and only provide a rough intuition underlying the choice of the appropriate parameter α. We will choose α much larger than the capacity of the SST, so that, by pumping arguments, one can show that, for every state q and every parameter $\beta \geq \alpha$, the β-approximant cannot be strictly smaller than the α-approximant on *all* valuations from Val_q.

Normalization of transitions

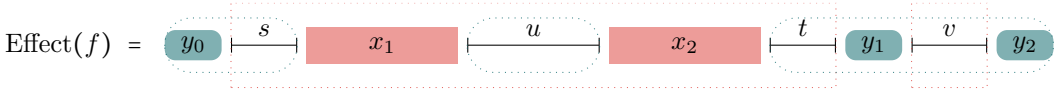
We finally turn to studying a notion of equivalence on transitions that is similar to the two-sided Myhill-Nerode equivalence on words. We will only compare transitions that consume the same input letter and link the same pair of states. Instead of using words as two-sided contexts, we will use initial and final runs that can be attached to the considered transitions in order to form successful runs, and instead of comparing membership in a language, we will compare the effect on the produced outputs.

Consider two transitions $\tau_1 : q \xrightarrow{a/f_1} q'$ and $\tau_2 : q \xrightarrow{a/f_2} q'$. We say that τ_1 and τ_2 are *equivalent* if for every initial run ρ leading to q and every final run σ starting in q', the outputs $\mathrm{out}(\rho\,\tau_1\,\sigma)$ and $\mathrm{out}(\rho\,\tau_2\,\sigma)$ are equal. We often refer to (ρ, σ) as a *context* for τ_1, τ_2.

In general, two transitions of an SST having the same source, target and Σ-label might turn out to be non-equivalent, and still produce the same output within specific contexts. However, Lemma 5 below shows that this is not the case with α-approximants at hand, provided that α is at least the capacity of the SST. More precisely, we will show that the equivalence of two transitions $\tau_1 : (q, A_X, A_Y) \xrightarrow{a/f_1} (q', A'_X, A'_Y)$ and $\tau_2 : (q, A_X, A_Y) \xrightarrow{a/f_2} (q', A'_X, A'_Y)$, where f_1, f_2 have the same flow, only depends on the α-approximants A_X, A'_Y that annotate the source and target states. This will imply that τ_1, τ_2 either always produce the same output or always produce different outputs, independently of the surrounding contexts. To prove the statement, we have to consider register valuations induced by initial runs, and symmetrically gap valuations induced by final runs. It helps to introduce the following:

Notation. Given an initial run ρ, $\mathrm{val}_{\rho\bullet}$ is the register valuation induced at the end of ρ; symmetrically, $\mathrm{val}_{\bullet\sigma}$ is the gap valuation induced at the beginning of a final run σ.

We also recall a consequence of the flow normalization: the effect on the final output of an update f that occurs in a successful run can be described by a word $\mathrm{Effect}(f)$ over the alphabet $X \uplus Y \uplus \Gamma$, defined as $\mathrm{Effect}(f) = y_0\, f(x_1)\, y_1 \dots f(x_m)\, y_{m+1}$. Note that in $\mathrm{Effect}(f)$ each register (resp. gap) occurs exactly once, according to the order $x_1 < \dots < x_m$ (resp. $y_0 < \dots < y_m$) – the occurrences of registers and gaps, however, may not be strictly interleaved. In $\mathrm{Effect}(f)$, an occurrence of $x_i \in X$ represents an abstract valuation for register x_i *before* applying the update f, while an occurrence of $y_j \in Y$ represents an abstract valuation for the gap y_j *after* applying f. In particular, note that the x's and the y's refer to valuations induced at different positions of a run. The maximal factors of $\mathrm{Effect}(f)$ that are entirely over Γ represent the words that need to be added in order to get the register valuation *after* f, or equally the gap valuation *before* f. For instance, by reusing the example update f from page 5, where $f(x_1) = s\,x_1\,u\,x_2\,t$ and $f(x_2) = v$, the effect of f is described by the word $\mathrm{Effect}(f) = y_0\,s\,x_1\,u\,x_2\,t\,y_1\,v\,y_2$, suggestively depicted as



(here the lengths of the blocks labeled with variables are immaterial). Note that the y_j above are in fact the y'_j from the picture at page 5. In the above figure we have also highlighted with dotted rectangles the factors that represent gap valuations before the update (e.g. $y_0\,s$), and register valuations after the update (e.g. $s\,x_1\,u\,x_2\,t$).

Given a valuation ν and two approximants $A \in \mathcal{L}^X_\alpha$ and $A' \in \mathcal{L}^Y_\alpha$, one for register valuations and the other for gap valuations, we write $\nu \in A \uplus A'$ to mean that $\nu(x) \in A(x)$ and $\nu(y) \in A'(y)$ for all $x \in X$ and $y \in Y$.

▶ **Lemma 5.** *Let T be a trimmed flow-normalized SST. Given two transitions $\tau_i : q \xrightarrow{a/f_i} q'$, with $i \in \{1,2\}$, a context (ρ,σ) for them, and the α-approximants $A = ((val_{\rho\bullet})^{\uparrow\alpha})|_X$ and $A' = ((val_{\bullet\sigma})^{\uparrow\alpha})|_Y$, with $\alpha \in \mathbb{N}$, the following holds:*

1. *If $Effect(f_1) = Effect(f_2)$ holds on all valuations $\nu \in A \uplus A'$, then $out(\rho\,\tau_1\,\sigma) = out(\rho\,\tau_2\,\sigma)$.*
2. *If τ_1, τ_2 have the same flow, $out(\rho\,\tau_1\,\sigma) = out(\rho\,\tau_2\,\sigma)$, and $\alpha \geq c$, where c is the capacity of T, then $Effect(f_1) = Effect(f_2)$ holds on all valuations $\nu \in A \uplus A'$.*

Recall that in an SST that admits α-approximants, states are of the form (q, A_X, A_Y), and we have $((\mathrm{val}_{\rho\bullet})^{\uparrow\alpha})|_X = A_X$ (resp. $((\mathrm{val}_{\bullet\sigma})^{\uparrow\alpha})|_Y = A_Y$) for every initial run ρ that ends in (q, A_X, A_Y) (resp. for every final run σ that starts in (q, A_X, A_Y)). By pairing this with Lemma 5, we immediately obtain the following corollary:

▶ **Corollary 6.** *Let T be a trimmed flow-normalized SST. One can decide in polynomial time whether two given transitions τ_1, τ_2 of T with the same flow are equivalent. Moreover, if T has capacity c and admits α-approximants for some $\alpha \geq c$, then only two cases can happen:*

1. *either $out(\rho\,\tau_1\,\sigma) = out(\rho\,\tau_2\,\sigma)$ for every context (ρ,σ) (so τ_1, τ_2 are equivalent),*
2. *or $out(\rho\,\tau_1\,\sigma) \neq out(\rho\,\tau_2\,\sigma)$ for* every *context (ρ,σ) (so τ_1, τ_2 are not equivalent).*

Another important consequence is the following theorem, that normalizes finite-valued SST in order to bound the maximum number of transitions linking the same pair of states and consuming the same input letter. This number is called *edge ambiguity* for short.

▶ **Theorem 7.** *Let T be a k-valued, flow-normalized SST that has m registers, capacity c, and that admits α-approximants, for some $\alpha \geq c$. One can construct an equivalent SST T', with the same states and the same registers as T, that has edge ambiguity at most $k \cdot 2^m$.*

Proof. By Corollary 6, T has at most k pairwise non-equivalent transitions with the same input letter, the same source and target states, and the same flow. Moreover equivalence of such transitions can be decided. We can then normalize T by removing in each equivalence class all but one transitions with the same flow. Since T has at most 2^m flows, the normalization results in an equivalent SST T' with edge ambiguity at most $k \cdot 2^m$. ◀

The next section is devoted to prove a very similar result as above, but for all SST that can be obtained by shortcutting runs into single transitions, and that thus have arbitrary large capacity. This will be the main technical ingredient for establishing the decidability of the equivalence problem for k-valued SST.

4 Shortcut construction

Here we focus on a transformation of relations that absorbs the first input letter when this is equal to a specific element, say $a \in \Sigma \setminus \{\dashv\}$. Such a transformation maps any relation R to the relation $R_a = \{(u,v) : (au,v) \in R\}$. Observe that $R = R_\varepsilon \cup \bigcup_{a\in\Sigma} R_a$, where $R_\varepsilon = R \cap (\{\dashv\} \times \Gamma^*)$.

It is easy to see that the class of relations realized by SST is effectively closed under the transformation $R \mapsto R_a$. To prove this closure property, it is convenient to restrict, without loss of generality, to SST with *transient initial states*, namely, SST where no transition reaches an initial state. Under this assumption, the closure property also preserves the state space (though some states may become useless), the set of registers, the property of being k-valued, as well as the α-approximants, if they are admitted by the original SST. However, the transformation does not preserves the capacity, which may increase.

▶ **Lemma 8.** *Given a flow-normalized SST T with transient initial states, and given a letter $a \in \Sigma \setminus \{\dashv\}$, one can construct an SST T_a with transient initial states such that*

- *$Dom(T_a) = \{u \in \Sigma^* : au \in Dom(T)\}$,*
- *T_a on input u produces the same outputs as T on input au.*

Moreover, T_a has the same states and the same registers as T; if T has capacity c, then T_a has capacity $2c$; if T admits α-approximants, then so does T_a (via the same annotation).

The above construction can be applied inductively to compute an SST for any *left quotient* $R_u = \{(v, w) : (u\,v, w) \in R\}$ of R. For $u = a_1 \dots a_n \in (\Sigma \setminus \{\dashv\})^*$, we let $T_u = (\dots (T_{a_1})_{a_2} \dots)_{a_n}$.

The last and most technical step consists in proving that edge ambiguity can be uniformly bounded in every SST T_u, provided that the initial SST T is finite-valued, flow-normalized, and admits α-approximants for a large enough α. More precisely, we aim at establishing that, for α much larger than the capacity of T, the notion of α-approximant, besides satisfying Equation (1), also satisfies the following property:

$$\begin{aligned} \forall \beta \geq \alpha \quad \exists \rho \ : \qquad (\mathrm{val}_{\rho\bullet})^{\uparrow\beta} &= (\mathrm{val}_{\rho\bullet})^{\uparrow\alpha} \\ \forall \beta \geq \alpha \quad \exists \sigma \ : \qquad (\mathrm{val}_{\bullet\sigma})^{\uparrow\beta} &= (\mathrm{val}_{\bullet\sigma})^{\uparrow\alpha}. \end{aligned} \tag{2}$$

In this case we say that the α-approximants are *tight*.

Intuitively, the above property can be explained as follows. When considering an SST with states annotated with α-approximants, it may happen that for some larger parameter β some initial runs induce register valuations at a state (q, A_X, A_Y) whose β-approximants are strictly included in A_X (e.g. possibly entailing new periodicities). These runs should be thought of as exceptional cases, and there is a way of pumping them so as to restore the equality between A_X and the induced β-approximant.

Let us first see how tight approximants are used. The theorem below assumes that there is an SST T' with tight approximants (later we will show how to compute such an SST), and bounds the edge ambiguity of the SST T'_u that realizes a left quotient of T'.

▶ **Theorem 9.** *Let T' be a k-valued, flow-normalized SST realizing R, with transient initial states and tight α-approximants. For every $u \in \Sigma^*$, one can construct an SST T'_u realizing R_u, with the same states and the same registers as T', and with edge ambiguity at most $k \cdot 2^m$.*

Proof. The crux is to show that the SST T'_u obtained from Lemma 8 has at most k pairwise non-equivalent transitions with the same flow (for any given source/target state and label). Once this is proven, one can proceed as in the proof of Theorem 7, by removing all but one transition with the same flow in each equivalence class. By way of contradiction, assume that T'_u has $k+1$ pairwise non-equivalent transitions $\tau_1, \dots, \tau_{k+1}$ with the same flow. Since T' admits tight α-approximants, by Lemma 8 we know that the source and target state, respectively, of the previous transitions are annotated with tight α-approximants, say (A_X, A_Y) and (A'_X, A'_Y), respectively.

We begin by applying the first claim of Lemma 5, implying that the equation $\mathrm{Effect}(f_i) = \mathrm{Effect}(f_j)$ is violated for some valuation $\nu \in A_X \uplus A'_Y$. Then, we let $\beta = \max(\alpha, |u|c)$ and use Equations (1) and (2) to get a context (ρ, σ) such that $(\mathrm{val}_{\rho\bullet})^{\uparrow\beta} = A_X$ and $(\mathrm{val}_{\bullet\sigma})^{\uparrow\beta} = A'_Y$. Finally, knowing that β is at least the capacity of T'_u, we apply the second claim of Lemma 5 to get $\mathrm{out}(\rho\,\tau_i\,\sigma) \neq \mathrm{out}(\rho\,\tau_j\,\sigma)$, thus witnessing non-equivalence of all pairs of transitions τ_i, τ_j at the same time. This contradicts the assumption that T'_u (and hence T') is k-valued. ◀

Now, let T be a flow-normalized SST with m registers, capacity c, and trimmed state space Q. Below, we show how to compute, with the help of Proposition 4, an SST T' equivalent to T that admits tight approximants. For simplicity, we will mostly focus on

register valuations induced by initial runs, even though similar results can be also stated for gap valuations induced by final runs. We begin by giving a few technical results based on pumping arguments. We say that register x is *productive* along ρ if the update induced by ρ maps x to a word that contains at least one letter from Γ. We also recall that a loop of a run needs to induce a flow-idempotent update.

▶ **Lemma 10.** *If $\rho = \rho_1 \gamma \rho_2$ is an initial run of T, with γ loop, then for every $n > 0$ the pumped run $\rho^{(n)} = \rho_1 \gamma^n \rho_2$ induces valuations $val_{\rho^{(n)}\bullet}$ mapping any register x to a word of the form $u_0 v_1^{n-1} u_1 \dots v_{2m}^{n-1} u_{2m}$, where $u_0, \dots, u_{2m}, v_1, \dots, v_{2m} \in \Gamma^*$ depend on ρ and x, but not on n. Moreover, we have $v_i \neq \varepsilon$ for some i if there is a register x' that is productive along γ and that flows into x along ρ_2.*

Given a tuple of pairwise disjoint loops $\overline{\gamma} = \gamma_1, \dots, \gamma_\ell$ in a run ρ, we write $\rho' \trianglerighteq_{\overline{\gamma}} \rho$ when ρ' is obtained from ρ by *simultaneously* pumping n times every loop γ_i, for some $n > 0$. When using this notation, we often omit the subscript $\overline{\gamma}$; in this case we tacitly assume that $\overline{\gamma}$ is *uniquely determined* from ρ. In this way, when writing, for instance, $\rho', \rho'' \trianglerighteq \rho$, we will know that ρ', ρ'' are obtained by pumping the same loops of ρ. We also say that a property on runs holds *for all but finitely many* $\rho' \trianglerighteq \rho$ if it holds on runs ρ' that are obtained from ρ by pumping n times the loops in a fixed tuple $\overline{\gamma}$, for all $n > n_0$ and for a sufficiently large n_0.

▶ **Lemma 11.** *Let ρ be an initial run and x a register. If $val_{\rho\bullet}(x)$ has length (resp. period) larger than $\alpha = m\, c\, |Q|\, 2^{3 \cdot 2^m}$, then for every $\beta \geq \alpha$ and for all but finitely many $\rho' \trianglerighteq \rho$, $val_{\rho'\bullet}(x)$ has length (resp. period) larger than β.*

Using the previous lemmas and the fact that the type of quantification "for all but finitely many runs" commutes with conjunctions (e.g. those used to enforce properties on each register $x \in X$), we obtain that α-approximants are tight for sufficiently large α:

▶ **Proposition 12.** *Let T' be the SST admitting α-approximants that is obtained from T using Proposition 4, for any $\alpha \geq m\, c\, |Q|\, 2^{3 \cdot 2^m}$, where Q is the set of states of T. The α-approximants of T' are tight.*

5 Equivalence algorithm

The equivalence algorithm for k-valued SST follows a classical approach of Culik and Karhumäki [10] that is based on so-called *test sets*. A test set for two SST T_1, T_2 over input alphabet Σ is a set $F \subseteq \Sigma^*$ such that T_1, T_2 are equivalent if and only if they are equivalent over F. The main contribution of [10] is to show that *finite* test sets exist and be computed effectively for k-valued one-way transducers.The key ingredient of their proof is to show the existence of a test set that works for *all* transducers with fixed number of states. An essential observation is that for k-valued one-way, or even two-way, transducers one can assume that the edge ambiguity is at most k. The reason for this is simply that the output is generated sequentially. For SST the situation is far more complex because the output is generated piecewise. The purpose of the normalizations performed in Section 3 was precisely to restore the property of bounded edge ambiguity.

In a nutshell, the existence of a test set for transducers is a consequence of Ehrenfeucht's conjecture, whereas the effectiveness is based on the resolution of word equations due to Makanin (see e.g. the survey [11]).

Ehrenfeucht's conjecture was originally stated as a conjecture about formal languages: for every language $L \subseteq \Sigma^*$, there is a finite subset $F \subseteq L$ such that for all morphisms $f, g : \Sigma^* \to \Delta^*$, $f(w) = g(w)$ for every $w \in L$ if and only if $f(w) = g(w)$ for every $w \in F$. Such a set F is called a *test set* for L.

There is an equivalent formulation of Ehrenfeucht's conjecture in terms of a compactness property of word equations [20]. Let Σ and Ω be two alphabets, where the elements in Ω are called unknowns. A word equation is a pair $(u, v) \in \Omega^* \times \Omega^*$, and a solution is a morphism $\sigma : \Omega^* \to \Sigma^*$ such that $\sigma(u) = \sigma(v)$. Ehrenfeucht's conjecture is equivalent to saying that any system of equations over a finite set Ω of unknowns has a finite, equivalent subsystem, where equivalence means that the solution sets are the same. The latter compactness property was proved in [1, 15] by encoding words by polynomials and using Hilbert's basis theorem.

In view of Propositions 2, 4, and 12, we can restrict without loss of generality to SST that are flow-normalized and that admit tight approximants. Hereafter, we shall tacitly assume that all transducers are of this form. Given some integers k, n, m, and e, let $\mathcal{C}_k(n, m, e)$ be the class of k-valued SST with at most n states, m registers, and edge-ambiguity at most e. Note that if T is k-valued, then by Theorem 7 it belongs to $\mathcal{C}_k(n, m, e)$, where n, m are the number of states and registers of T and $e = k \cdot 2^m$. Similarly, by Lemma 8 and Theorem 9, every left quotient T_u also belongs to $\mathcal{C}_k(n, m, e)$.

Now, let us fix k, n, m, e and consider an arbitrary SST T from $\mathcal{C}_k(n, m, e)$. Following [10] we first build an abstraction of T by replacing each maximal factor from Γ^* occurring in some update function of T, by a distinct unknown from Ω. The SST $\Delta(T)$ obtained in this way is called a *schema*; its outputs are words over Ω. Note that the assumption of bounded edge ambiguity is essential here to get a uniform bound on the number of unknowns required for a schema. Clearly, there are only finitely many schemas of SST in $\mathcal{C}_k(n, m, e)$. We denote by $\phi_T : \Omega \rightharpoonup \Gamma^*$ the partial mapping (concretization) that associates with each unknown the corresponding word from Γ^* as specified by the updates of T.

We can rephrase the equivalence $T_1 \equiv T_2$ of two arbitrary SST from $\mathcal{C}_k(n, m, e)$ as an infinite "system" of word equations[1] $\mathcal{S} = \bigwedge_{u \in \Sigma^*} \bigvee_\pi S_\pi$ over set of unknowns $\Omega \uplus \Omega'$. The unknowns from Ω are used for the schema $\Delta(T_1)$, whereas those from Ω' are used for $\Delta(T_2)$; in particular, $\phi_{T_1} : \Omega \to \Gamma^*$ and $\phi_{T_2} : \Omega' \to \Gamma^*$. The disjunctions in $\mathcal{S}$ are finite, with π ranging over the possible schemas Δ_1, Δ_2 (for T_1 and T_2, respectively) and the possible partitions of the set of runs of Δ_1 and Δ_2 over the input u, into at most k groups (one for each possible output). Finally, S_π is a (finite) system of word equations, stating the equality of the words from $\Omega^* \cup \Omega'^*$ that belong to the same group according to π.

The following lemma was stated in [10] for k-valued one-way transducers, but it holds as well for two-way transducers and for SST (even copyful, with a proper definition for $\mathcal{C}_k(n, m, e)$):

▶ **Lemma 13.** *Given two SST T_1, T_2 from $\mathcal{C}_k(n, m, e)$, the system $\mathcal{S} = \bigwedge_{u \in \Sigma^*} \bigvee_\pi S_\pi$ has $\phi_{T_1} \uplus \phi_{T_2}$ as solution if and only if $T_1 \equiv T_2$.*

As shown in [10], the Ehrenfeucht conjecture can be used to show that any infinite system $\mathcal{S}$ as in Lemma 13 is equivalent to some *finite* sub-system $\mathcal{S}_N = \bigwedge_{u \in \Sigma^{\leq N}} \bigvee_\pi S_\pi$. This gives:

▶ **Lemma 14.** *Given $n, m, e \in \mathbb{N}$, there is $N \in \mathbb{N}$ such that $\Sigma^{\leq N}$ is a test set for every pair of SST T_1, T_2 from $\mathcal{C}_k(n, m, e)$.*

Using Theorem 7 and Lemma 14 we can derive immediately the existence of a finite test set for any two k-valued SST. The last question is how to compute such a test set effectively. For this we will use the shortcut construction provided in Section 4.

[1] Formally, $\mathcal{S}$ depends on n and k, but for simplicity we leave out the indices.

▶ **Lemma 15.** *Assume that the formulas $\mathcal{S}_N$ and $\mathcal{S}_{N+1}$ are equivalent, i.e., they have the same solutions. Then $\Sigma^{\leq N}$ is a test set for any pair of SST from $\mathcal{C}_k(n,m,e)$.*

Proof. Let $T_1 \equiv_r T_2$ denote equivalence of T_1 and T_2 relativized to $\Sigma^{\leq r}$. The goal is to prove that $\Sigma^{\leq N}$ is a test set, namely, for all $r > N$ and all $T_1, T_2 \in \mathcal{C}_k(n,m,e)$, $T_1 \equiv_r T_2$ holds if and only if $T_1 \equiv_N T_2$. Clearly, for any $r \geq 0$, $T_1 \equiv_{r+1} T_2$ is equivalent to $T_{1,a} \equiv_r T_{2,a}$ for every $a \in \Sigma$, and $T_1 \equiv_0 T_2$ (the latter being abbreviated as $(*)$ below). Moreover, by Theorem 9, we have $T_{1,a}, T_{2,a} \in \mathcal{C}_k(n,m,e)$. This enables the following proof by induction on r:

$$\begin{array}{ccccc} T_1 \equiv_{r+1} T_2 & \Leftrightarrow & T_{1,a} \equiv_r T_{2,a}\ (\forall a \in \Sigma) \text{ and } (*) & & T_1 \equiv_N T_2. \\ & & \Updownarrow \text{(ind. hyp.)} & & \Updownarrow \\ & & T_{1,a} \equiv_N T_{2,a}\ (\forall a \in \Sigma) \text{ and } (*) & \Leftrightarrow & T_1 \equiv_{N+1} T_2 \end{array}$$ ◀

Using Makanin's algorithm for solving word equations (and even for deciding the existential theory of word equations, see e.g. [11] for a modern presentation) we obtain:

▶ **Proposition 16.** *Given $n,m,e \in \mathbb{N}$, there is $N \in \mathbb{N}$ such that $\Sigma^{\leq N}$ is a test set for every pair of SST from $\mathcal{C}_k(n,m,e)$, and such an N can be effectively computed.*

Proof. By Lemma 14 we know that N exists, and Makanin's algorithm allows to determine whether $\mathcal{S}_N, \mathcal{S}_{N+1}$ are equivalent, so to determine N by Lemma 15. ◀

We finally obtain the main result:

▶ **Theorem 1.** *The equivalence problem for finite-valued SST is decidable.*

Of course, Theorem 1 does not come with any complexity upper bound, mainly because of the Ehrenfeucht conjecture. The only known lower bound is PSPACE-hardness, which holds even for single-valued SST over unary output alphabets, and follows from a simple reduction from universality of NFA.

Quite surprisingly, the exact complexity of equivalence is not known even for *deterministic* SST, where the problem is known to be between NLOGSPACE and PSPACE [3]. We also recall that equivalence of deterministic SST with *unary output* can be checked in PTIME using invariants [4]. Finally, we recall that the currently best upper bound for solving word equations is PSPACE [21] (with even linear space requirement, as shown in [19]).

6 Conclusions

Our paper answers to a question left open in [5], showing that the equivalence problem for finite-valued SST is decidable. We followed a proof for one-way transducers due to Culik and Karhumäki [10], that is based on the Ehrenfeucht conjecture. The main contribution of the paper is to provide the technical development that allows to follow the proof scheme of [10]. We believe that this development will also allow to obtain stronger results. We conjecture that finite-valued SST can be effectively decomposed into finite unions of unambiguous SST. This would entail that in the finite-valued setting, two-way transducers and SST have the same expressive power, as is it the case for single-valued transducers. If this holds with elementary complexity, then the equivalence of single-valued SST (or two-way transducers) could also be solved with elementary complexity. We believe that the complexity is indeed elementary, and leave this for future work.

References

1 M.H. Albert and J. Lawrence. A proof of Ehrenfeucht's conjecture. *Theor. Comput. Sci.*, 41(1):121–123, 1985.

2 Rajeev Alur and Pavel Cerný. Expressiveness of streaming string transducer. In *IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS'10)*, volume 8 of *LIPIcs*, pages 1–12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010.

3 Rajeev Alur and Pavol Cerný. Streaming Transducers for Algorithmic Verification of Single-pass List-processing Programs. In *POPL'11*. ACM, 2011.

4 Rajeev Alur, Loris D'Antoni, Jyotirmoy Deshmukh, Mukund Raghothaman, and Yifei Yuan. Regular functions and cost register automata. In *Proc. of Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2013)*, pages 13–22. IEEE, 2013.

5 Rajeev Alur and Jyotirmoy Deshmukh. Nondeterministic streaming string transducers. In *International Colloquium on Automata, Languages and Programming (ICALP'11)*, volume 6756 of *LNCS*. Springer, 2011.

6 Michael Benedikt, Timothy Duff, Aditya Sharad, and James Worrell. Polynomial automata: Zeroness and applications. In *Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'17)*, pages 1–12. IEEE, 2017.

7 Adrien Boiret, Radoslaw Piórkowski, and Janusz Schmude. Reducing Transducer Equivalence to Register Automata Problems Solved by "Hilbert Method". In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'18)*, volume 122 of *LIPIcs*, pages 48:1–48:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

8 Mikolaj Bojańczyk. The Hilbert Method for Transducer Equivalence. *ACM SIGLOG News*, January 2019.

9 Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*, volume 138 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2012.

10 Karel Culik II and Juhani Karhumäki. The equivalence of finite valued transducers (on HDT0L languages) is decidable. *Theor. Comput. Sci.*, 47:71–84, 1986.

11 Volker Diekert. Makanin's Algorithm. In M. Lothaire, editor, *Algebraic combinatorics on words*, volume 90 of *Encyclopedia of mathematics and its applications*, chapter 12, pages 387–442. Cambridge University Press, 2002.

12 Joost Engelfriet and Hendrik Jan Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Log.*, 2(2):216–254, 2001.

13 Patrick C. Fischer and Arnold L. Rosenberg. Multi-tape one-way nonwriting automata. *J. Comput. and System Sci.*, 2:88–101, 1968.

14 Paul Gallot, Anca Muscholl, Gabriele Puppis, and Sylvain Salvati. On the Decomposition of Finite-Valued Streaming String Transducers. In *Annual Symposium on Theoretical Aspects of Computer Science (STACS'17)*, volume 66 of *LIPIcs*, pages 34:1–34:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

15 Victor S. Guba. Equivalence of infinite systems of equations in free groups and semigroups to finite subsystems. *Mat. Zametki*, 40(3):688—-690, 1986.

16 Eitan M. Gurari. The equivalence problem for deterministic two-way sequential transducers is decidable. *SIAM Journal of Computing*, 448–452, 1982.

17 Eitan M. Gurari and Oscar H. Ibarra. A note on finite-valued and finitely ambiguous transducers. *Math. Syst. Theory*, 16(1):61–66, 1983.

18 Oscar H. Ibarra. The unsolvability of the equivalence problem for e-free NGSM's with unary input (output) alphabet and applications. *SIAM J. of Comput.*, 7(4):524–532, 1978.

19 Artur Jez. Word Equations in Nondeterministic Linear Space. In *Proc. International Colloquium on Automata, Languages, and Programming (ICALP'17)*, volume 80 of *LIPIcs*, pages 95:1–95:13. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017.

20 Juhani Karhumäki. The Ehrenfeucht conjecture: a compactness claim for finitely generated free monoids. *Theor. Comput. Sci.*, 29:285–308, 1984.

21 Wojciech Plandowski. Satisfiability of word equations with constants is in PSPACE. *JACM*, 51(3):483–496, 2004.

22 Jacques Sakarovitch and Rodrigo de Souza. On the decomposition of k-valued rational relations. In *Annual Symposium on Theoretical Aspects of Computer Science (STACS'08)*, volume 1 of *LIPIcs*, pages 621–632. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2008.

23 Jacques Sakarovitch and Rodrigo de Souza. Lexicographic decomposition of K-valued transducers. *Theory Comput. Sci.*, 47:758–785, 2010.

24 Andreas Weber. Decomposing A k-Valued Transducer into k Unambiguous Ones. *RAIRO-ITA*, 30(5):379–413, 1996.

From Normal Functors to Logarithmic Space Queries

Lê Thành Dũng Nguyễn
LIPN, UMR 7030 CNRS, Université Paris 13, Sorbonne Paris Cité, France
https://nguyentito.eu/
nltd@nguyentito.eu

Pierre Pradic
ENS de Lyon, Université de Lyon, LIP, France
University of Warsaw, Faculty of Mathematics, Informatics and Mechanics, Poland
http://perso.ens-lyon.fr/pierre.pradic/
pierre.pradic@ens-lyon.fr

Abstract

We introduce a new approach to implicit complexity in linear logic, inspired by functional database query languages and using recent developments in effective denotational semantics of polymorphism. We give the first sub-polynomial upper bound in a type system with impredicative polymorphism; adding restrictions on quantifiers yields a characterization of logarithmic space, for which extensional completeness is established via descriptive complexity.

2012 ACM Subject Classification Theory of computation → Linear logic; Theory of computation → Complexity theory and logic; Theory of computation → Finite Model Theory

Keywords and phrases coherence spaces, elementary linear logic, semantic evaluation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.123

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version A full version of the paper containing the technical appendix is available at https://hal.archives-ouvertes.fr/hal-02024152.

Funding *Lê Thành Dũng Nguyễn*: Partially supported by the Elica project (ANR-14-CE25-0005). *Pierre Pradic*: Partially supported by the the RAPIDO project (ANR-14-CE25-0007).

Acknowledgements L. T. D. Nguyễn wishes to thank Damiano Mazza, Thomas Seiller and Kazushige Terui for highly instructive discussions. P. Pradic thanks Alexis Ghyselen for his valuable feedback on a first draft of this paper.

1 Introduction

Machine-free complexity. We pursue here a research theme advocated by Leivant [26]: using type systems and the proofs-as-programs correspondence to define functional languages whose expressible functions are exactly those of a given complexity. This usually consists of two independent parts: *soundness* – all those functions admit such complexity bounds – and *extensional completeness* – for every algorithm with this complexity, there is an expressible program computing the same function. This is part of the general area of *implicit computational complexity* (ICC), whose goal is to obtain characterizations of complexity classes by programming languages, without explicit resource bounds on a machine model (other methods in ICC include, for instance, recursive function algebras).

On the other hand, *descriptive complexity* is closer to a declarative programming paradigm: it consists in characterizing complexity classes as sets of *queries* – predicates over finite first-order relational structures – written in some logic. (Such structures often go by the

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 123; pp. 123:1–123:15


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

name of *finite models*; see Definition 3.) The field was launched by Fagin's result that NP queries correspond to existential second-order logic [11]. For our purposes, an useful example is Immerman's characterization of *deterministic logarithmic space* (L) (Theorem 13).

This idea of representing inputs as finite first-order structures also appeared in the early history of ICC: Gurevich [17] showed in 1983 that in this setting, a form of primitive recursion captures L. But unlike in descriptive complexity, Gurevich considers endofunctions instead of relations and queries.

Queries in the λ-calculus. Hillebrand's PhD thesis [18] is a junction point between implicit and descriptive complexity. The idea was to represent finite models inside the simply typed λ-calculus (STλ), using them to represent the inputs to programs. By doing so, Hillebrand et al. managed to characterize P [19], PSPACE [1] and k-EXPTIME/k-EXPSPACE[1] [20] – the extensional completeness for the first two being established through descriptive complexity.

Keeping in mind the connections between finite model theory and relational databases, this can also be seen as using STλ as a functional language for database queries, expressive enough to admit translations from other languages such as Datalog, as is done in [21].

The present paper could then be motivated as looking for a *sub-polynomial*[2] functional query language, filling a gap in the aforementioned work.

Linear logic for ICC. Here it is natural to turn to *linear logic*, a constructive logic born from the proofs-as-programs correspondence, in which several characterizations of sub-polynomial complexity classes have already been devised [38, 36, 7, 28, 29]. From its inception, linear logic has indeed had the ambition to "help us improve the efficiency of programs" [13, p. 3], and a landmark result in that direction was characterizing P through Light Linear Logic [16].

In this paper, we will use Elementary Linear Logic (ELL) [16, 8], which was originally introduced to capture the class ELEMENTARY[3]. A recent line of work by Baillot et al. [2, 3, 4] shows that one can define, inside variants of ELL, types of programs which compute smaller complexity classes, such as P. We follow this approach, by introducing a type `Inp` which is essentially an abstract data type[4] for finite models. Our main result is (writing $\mathtt{Bool} = 1 \oplus 1$):

▶ **Theorem 1.** *The class of queries computed by the proofs of* $\mathtt{Inp} \multimap !!\mathtt{Bool}$ *in second-order Elementary Linear Logic* (ELL_2) *is* between L and NL. *Furthermore, a suitable restriction on the existential witnesses in the proof gives an* exact characterization of L.

Here NL stands for *non-deterministic logarithmic space*. Actually, we obtain a better upper bound than NL in the unrestricted case, namely the class L^{UL} which will be defined later. But we believe that this is still not optimal:

▶ **Conjecture 1.** *Even without the restriction, the class of queries obtained is* exactly L.

---

[1] k-EXPTIME(resp. k-EXPSPACE) is the class of functions which can be computed in time (resp. space) $2 \uparrow^k (p(n))$, where p is a polynomial and n is the size of the input. (We use Knuth's up-arrow notation [24] for iterated exponentials: $2 \uparrow^{k+1} (n) = 2^{2\uparrow^k(n)}$, and $2 \uparrow^0 (n) = n$.)

[2] That is, capturing a complexity class below P. To be fair, Hillebrand's thesis does define a characterization of the sub-polynomial class of *first-order queries* (FO) in STλ, but this class has very little expressivity, and our work captures a class still well above FO.

[3] This is the class of elementary recursive functions, i.e. the union over $k \in \mathbb{N}$ of the classes k-EXPTIME.

[4] This term is the programming language counterpart of existential formulas in logic, cf. infra.

Our characterization has a few distinctive features with respect to the previous variants of linear logic capturing logarithmic space [36, 7, 28]: it takes place in a simple pre-existing logical system, which contains only usual logical connectives, and no primitive datatypes[5]; at the price of a more involved encoding of inputs, the `Inp` type. But a main novelty, in our opinion, is the unrestricted case: to our knowledge, it is the first[6] sub-polynomial bound in a type system with *impredicative polymorphism*.

This forces our approach to be significantly different to these previous works: they all exploit some form of the Geometry of Interaction (GoI) [14, 9] as a space-efficient evaluator, whereas in our case this does not work[7] because of impredicative quantication. In the predicative case, there is still an obstruction to the GoI: the *additive* connectives of linear logic. Instead, our tool of choice will be *denotational semantics*.

Semantic evaluation and polymorphism. This is indeed the sequel to a previous paper [32] which studied the semantics of second-order Multiplicative-Additive Linear Logic (MALL_2) with applications in mind; in particular it proved that Girard's model of MALL_2 in *coherence spaces* [12, 13] is finite and effective. In order to establish our upper complexity bounds, we will compute the denotation of a program applied to its input in the coherence space model.

This *semantic evaluation* technique has been very successful before for establishing complexity bounds in STλ: it is how soundness is established in the aforementioned works of Hillebrand et al., and also underlies Terui's more recent result on the complexity of β-reduction in STλ at fixed order [39]. Beyond STλ, it has been applied to System T and PCF, see [25] and references therein. However, these applications had been confined to monomorphic type systems[8] until the prequel showed:

▶ **Theorem 2** ([32]). *The languages decided by proofs of* $!\texttt{Str} \multimap !!\texttt{Bool}$ *in* ELL_2*, where* `Str` *is the type of* ELL *Church encodings of strings, are exactly the* regular languages.

An analysis of the proof also suggested that to increase the expressivity[9] while keeping $!!\texttt{Bool}$ as output, one should replace `Str` by an *existential* input type. Hence the `Inp` type.

To perform semantic evaluation in a polymorphic language, one needs an effective model of polymorphism, and such models are not easy to build. First, one must first restrict to a purely linear language[10] such as MALL_2 to make a non-trivial finitary semantics possible. Even then, obstacles remain: for instance, the prequel [32] proved that no degenerate model of MALL_2 (in which $\otimes$ and $\parr$ are identified) can satisfy a desirable "constancy property", so this excludes the Scott model of linear logic used by [39]. Girard managed to build a

[5] Given the special status granted to unary Church integers by the "skewed iteration" rule in Schöpp's SBAL [36], it is fair to consider them to be primitive datatypes.

[6] Excluding the characterization of regular languages in ELL_2, cf. infra, but regular languages do not form a well-behaved complexity class (for instance they are not closed under uniform AC^0 reductions).

[7] We will not enter into details here, but essentially, the GoI works by "following paths" inside a proof, and in our case, the length of these paths would be super-polynomial.

[8] That said, there have been some uses of rather different semantic techniques for implicit complexity in presence of polymorphism, e.g. realizability [6].

[9] This was also a major motivation in the work of Hillebrand et al.: they wanted to overcome limits in STλ such as Statman's classical result that equality cannot be defined on STλ Church integers (see the introduction to [21]). Hillebrand and Kanellakis [20] later proved that the languages decided by STλ predicates over Church-encoded strings are regular (this inspired the analogous result on ELL_2). Such restrictions seem drastic since the β-equivalence problem for STλ is not in ELEMENTARY [37, 27], hinting that its computational power should be much greater. By using finite models as inputs, Hillebrand, Kanellakis and Mairson [21] manage to express all ELEMENTARY queries.

[10] The type $\forall X.\ X \to (X \to X) \to X$ of polymorphic Church integers – more generally, any infinite data type whose destructors are definable – has an inifinite denotation in any semantics of System F.

semantics for System F [12] which later turned out to be finite and effective for MALL_2 by representing types depending on type parameters as *normal functors*[11]. Although we will not have to study the properties of normal functors here – the semantic groundwork has been laid in the prequel – we consider that this ingredient is crucial enough to deserve inclusion in the title.

New complexity phenomena in MALL. The bottleneck for this L^{UL} bound is the complexity of an iterated composition problem: given a MALL_2 type A and k proofs $f_1, \ldots, f_k$ of $A \vdash A$, compute their composition $f_1 \circ \ldots \circ f_k$. To illustrate the kind of complexity constraint induced by the linearity of the f_i, consider the types $\mathtt{Bool} \otimes \ldots \otimes \mathtt{Bool}$ (n times) and $\mathtt{Bool} \,\&\, \ldots \,\&\, \mathtt{Bool}$ (n times). A non-linear function does not distinguish them, whereas for linear functions:

- an iteration over $\mathtt{Bool} \otimes \ldots \otimes \mathtt{Bool}$ can simulate a Turing machine running in space n (minus $O(1)$ bits for the control state);
- an iteration over $\mathtt{Bool} \,\&\, \ldots \,\&\, \mathtt{Bool}$ can be computed in space $O(\log(nk))$.

This kind of phenomenon surfaced when we tried to obtain bounds on our ELL_2 queries; we are not aware of a previous mention in the literature. Coherence spaces are sensitive to this (e.g. the interpretation of $\otimes$ and & bit vectors have respective sizes 2^n and $2n$) and thus manage to give a systematic sub-polynomial (but not L) bound on iterations.

For now, we have only managed to find a logarithmic space algorithm for those iterations in very specific cases of A, subsuming the above example. These cases still leave enough room for an extensional completeness result, leading to our exact characterization of L. But even in propositional MALL, the complexity of iterations remains mysterious.

Plan of the paper. In Section 2 we introduce the necessary definitions and state the main theorems. The lower bound on expressivity is established using descriptive complexity in Section 3, while our upper bounds are both proved in Section 4 via semantic evaluation.

2 Elementary Linear Logic as a query language

2.1 Linear Logic

In this paper, we assume some familiarity with the basic ideas of the proofs-as-programs paradigm and more specifically of linear logic. The formulas and the sequent calculus of second-order Multiplicative-Additive Linear Logic (MALL_2) are recalled in the full paper, in Appendix A. Recall that MALL_2 forbids using the structural rules of contraction and weakening, enforcing *linearity* whose computational meaning is that data cannot be duplicated or erased.

In order to allow the use of the structural rules in a controlled manner, the grammar of full Linear Logic extends the syntax of MALL_2 with *exponential modalities* $!F$ and $?F$ which allow to tag duplicable assumptions and conclusions. (Second-order) *Elementary Linear Logic* (ELL_2) corresponds to the subsystem whose rules governing the exponential connectives are given in Figure 1; this makes the principles of digging ($!A \multimap !!A$) and dereliction ($!A \multimap A$) invalid in ELL_2 while they are provable in full Linear Logic.

[11] A remark: the fact that our L^{UL} upper bound involves *unambiguous* nondeterminism, as we shall see, is related to the *stability* of linear maps in coherence spaces; stable maps are the "lower-dimensional analogue" of normal functors, and interestingly, it seems that stability is required for the construction of models of polymorphism based on normal functors.

$$\text{(functorial promotion)}\frac{\vdash \Gamma, A}{\vdash ?\Gamma, !A} \qquad \text{(weakening)}\frac{\vdash \Gamma}{\vdash \Gamma, ?A} \qquad \text{(contraction)}\frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}$$

Figure 1 Exponential rules for the ELL_2 sequent calculus. In the functorial promotion rule, when $\Gamma = B_1, \dots, B_k$, $?\Gamma$ stands for $?B_1, \dots, ?B_k$.

ELL_2 thus satisfies a *stratification* property: the *depth* of a given connective – i.e. the number of !/? modalities it is in the scope of – does not change during cut-elimination (key cut-elimination rules are also recalled in Appendix A). As a consequence, this notion of depth is of the utmost relevance for the computational complexity properties of ELL_2.

LL notations. When π and ρ have respective conclusions $\vdash \Gamma, A$ and $\vdash A^\perp, \Delta$, we write $\mathbf{cut}(\pi, \rho)$ for the proof of $\vdash \Gamma, \Delta$ consisting of a cut-rule with premises π and ρ. Given a proof $\pi : A$, $!\pi$ denotes the proof of $!A$ obtained by applying the promotion rule to π. As we formally use one-sided sequents, $A_1, \dots, A_n \vdash B$ is a notation for $\vdash A_1^\perp, \dots, A_n^\perp, B$.

2.2 Finite models

▶ **Definition 3.** *Let Σ be a first-order relational signature, i.e. a list of relation symbols $\{\mathcal{R}_0, \dots, \mathcal{R}_k\}$ with their respective arities $r_0, \dots, r_k$.*

A finite model $\mathfrak{D}$ *over Σ consists of a finite set D and an interpretation $\mathcal{R}_i^{\mathfrak{D}} \subseteq D^{r_i}$ for each relation symbol. It is* totally ordered *when $\mathcal{R}_0 = \leq$, $r_0 = 2$ and $\mathcal{R}_0^{\mathfrak{D}}$ is a total order.*

We write $\mathrm{FinMod}(\Sigma)$ *for the set of totally ordered finite models over Σ.*

As an example, a possible signature for binary strings is $\{\leq, S\}$ with arities 2 and 1. Finite models consist of a totally ordered set $(D, \leq^{\mathfrak{D}})$ with a unary predicate $S^{\mathfrak{D}}$; we interpret $(D, \leq^{\mathfrak{D}})$ as the indices of the string, and $S^{\mathfrak{D}}(d)$ as "the dth bit is set to 1".

▶ Remark 4. The "totally ordered" assumption is common in descriptive complexity (see e.g. Theorem 13) and will be often kept implicit in the paper. Indeed, there are order-independent queries requiring a total order to be expressed.

To use finite models as inputs for ELL_2 programs, we represent the elements of $\mathrm{FinMod}(\Sigma)$ as proofs of an ELL_2 formula $\mathtt{Inp}_\Sigma$.

▶ **Definition 5.** *We define the types with a free variable δ:*

$$\mathtt{List}[\delta] = \forall X.\, !(\delta \multimap X \multimap X) \multimap !(X \multimap X) \qquad \mathtt{C}[\delta] = \delta \multimap \delta \otimes \delta \qquad \mathtt{W}[\delta] = \delta \multimap 1$$

$$\mathtt{Ctx}[\delta] = !\mathtt{List}[\delta] \otimes !!\mathtt{C}[\delta] \otimes !!\mathtt{W}[\delta] \qquad \mathtt{Bool} = 1 \oplus 1 \qquad \mathtt{Rel}_r[\delta] = \delta^r \multimap \mathtt{Bool}$$

Given a signature $\Sigma = \{\leq, \mathcal{R}_1, \dots, \mathcal{R}_k\}$ with arities $r_0 = 2, r_1, \dots, r_k$, we also define:

$$\mathtt{Inp}_\Sigma[\delta] = \mathtt{Ctx}[\delta] \otimes \bigotimes_{0 \leq i \leq k} !!\mathtt{Rel}_{r_i}[\delta] \qquad \mathtt{Inp}_\Sigma = \exists \delta.\, \mathtt{Inp}_\Sigma[\delta]$$

We now define the encoding $\overline{\mathfrak{D}}$ of any totally ordered finite model $\mathfrak{D}$ over Σ as a proof of $\mathtt{Inp}_\Sigma[\mathtt{Fin}(n)]$, where $\mathtt{Fin}(n) = 1 \oplus \dots \oplus 1$ with n summands, n being the domain size.

Let $\mathfrak{D} = (D, \leq^{\mathfrak{D}}, \mathcal{R}_1^{\mathfrak{D}}, \dots, \mathcal{R}_k^{\mathfrak{D}}) \in \mathrm{FinMod}(\Sigma)$ with $\mathrm{Card}(D) = n$. Choose a bijection between D and the n proofs of $\mathtt{Fin}(n)$.

- *We represent D as a Church-encoded list of type* $\mathtt{List}[\mathtt{Fin}(n)]$ *enumerating the n elements of* $\mathtt{Fin}(n)$.
- *Each relation* $\mathcal{R}_i^{\mathfrak{D}}$ *can be represented by an element of* $\mathtt{Rel}_{r_i}[\mathtt{Fin}(n)]$.
- *Finally, since* $\mathtt{Fin}(n)$ *is a positive type, there are canonical elements of type* $\mathtt{C}[\mathtt{Fin}(n)]$ *and* $\mathtt{W}[\mathtt{Fin}(n)]$ *implementing the structural rules.*

▶ **Definition 6.** *A proof* π *of* $\mathtt{Inp}_\Sigma \multimap \mathtt{!!Bool}$ *defines the query which evaluates to true on* $\mathfrak{D} \in \mathrm{FinMod}(\Sigma)$ *iff the application of* π *to the encoding* $\overline{\mathfrak{D}}$ *reduces to* $\mathtt{!!true}$ *(where* $\mathtt{true}$ *is the proof of* $\mathtt{Bool} = 1 \oplus 1$ *proving the left occurrence of* 1*).*

2.3 Complexity classes and the main theorems

For the rest of the paper, we fix a signature $\Sigma = \{\mathcal{R}_0 = \leq, \mathcal{R}_1, \dots, \mathcal{R}_N\}$ with arities $r_0 = 2, r_1 \dots, r_N$.

As we said in the introduction, we write L (resp. NL) for the class of decision problems solvable in deterministic (resp. non-deterministic) logarithmic space. The *unambiguous* logarithmic space class UL [33] consists of the problems which can be solved by a NL Turing machine whose accepting runs are guaranteed to be *unique*: for each input, if the machine accepts, there is a single sequence of non-deterministic choices leading to the accepting state. (So $\mathsf{UL} \subseteq \mathsf{NL}$.) L^{UL} denotes L with an UL oracle; as usual we use the Ruzzo–Simon–Tompa definition[12] of space-bounded oracle machines [35, §4].

We can now state our result in the unrestricted case.

▶ **Theorem 7.** *The class of queries computed by the proofs of* $\mathtt{Inp}_\Sigma \multimap \mathtt{!!Bool}$ *in* ELL_2 *is between* L *and* L^{UL}.

It is known that $\mathsf{NL}^{\mathsf{NL}} = \mathsf{NL}$ (as noted in [23, Corollary 2], it follows from $\mathsf{NL} = \mathsf{coNL}$), so $\mathsf{L}^{\mathsf{UL}} \subseteq \mathsf{NL}$, hence the statement in the introduction. Furthermore, while $\mathsf{NL} \subset \mathsf{P}$, it is commonly believed that $\mathsf{NL} \neq \mathsf{P}$, so our class of queries is presumably strictly sub-polynomial.

To state the second main theorem, we now introduce a fragment of ELL_2 with an ad-hoc restriction on existential witnesses.

▶ **Definition 8.** *The set of* positive polynomial formulas PP *is the subset of* MALL_2 *formulas generated by the grammar* $P, Q, \dots ::= 0 \mid 1 \mid X \mid P \otimes Q \mid P \oplus Q$.

We define PP3 *to be the set of formulas of the form* $P \otimes (Q \multimap R)$*, where* $P, Q, R \in \mathrm{PP}$.

The logic $\mathrm{ELL}_2^{\mathrm{PP3}}$ *is defined by the same rules as* ELL_2 *except that we exclude the cut rule, and restrict the* $\exists$*-rule as follows: the witness must belong to* PP3.

The "cut-free" part is necessary because *a cut between two* $\mathrm{ELL}_2^{\mathrm{PP3}}$ *proofs does not necessarily normalize into a* $\mathrm{ELL}_2^{\mathrm{PP3}}$ *proof.* However, we do have:

▶ **Proposition 9.** *Let* π *and* ρ *be* $\mathrm{ELL}_2^{\mathrm{PP3}}$ *proofs with respective conclusions* $\vdash \Gamma, A$ *and* $\vdash A^\perp, \Delta$*. If* A *is* quantifier-free*, then* $\mathbf{cut}(\pi, \rho)$ *is in* $\mathrm{ELL}_2^{\mathrm{PP3}}$.

With $\mathrm{ELL}_2^{\mathrm{PP3}}$, we obtain an exact characterization of L:

▶ **Theorem 10.** *The class of queries computed by proofs of* $\mathtt{Inp}_\Sigma \multimap \mathtt{!!Bool}$ *in* $\mathrm{ELL}_2^{\mathrm{PP3}}$ *is* L.

[12] A remark on notation: they would write $\mathsf{L}^{\langle \mathsf{UL} \rangle}$ instead of L^{UL} and use the latter to denote a naive notion of oracle machine. See [35, Example 1] for an example of the subtleties involved: without a careful definition, $\mathsf{NL}^{\mathsf{NL}}$ would include NP.

3 The lower bound: encoding logarithmic space queries

In this section, we use descriptive complexity to get the lower bound in both theorems above (so, for the second one, this is an extensional completeness proof).

3.1 Reminder: Immerman's characterization of L

Descriptive complexity considers queries given by formulas in extensions of classical first-order logic. The first-order formulas over Σ are generated by the grammar $\phi, \psi, \ldots ::= \mathcal{R}_i(x_1, \ldots, x_{r_i}) \mid \neg\phi \mid \phi \vee \psi \mid \exists x.\phi$, where the x_j are variables.

As usual, the semantics of these formulas is specified by a "satisfaction" relation $\mathfrak{D} \models \phi[\sigma]$ for $\mathfrak{D} \in \mathrm{FinMod}(\Sigma)$, defined by induction over ϕ, where σ assigns elements of the domain D of $\mathfrak{D}$ to the free variables of ϕ: e.g. $\mathfrak{D} \models (\exists x.\phi)[\sigma]$ iff $\mathfrak{D} \models \phi[\sigma + (x \mapsto d)]$ for some $d \in D$. Thus, when such a formula ϕ is closed, it defines the query $\mathfrak{D} \mapsto (\mathfrak{D} \models \phi)$.

To express all logarithmic space queries, we need to extend our language of formulas with a *deterministic transitive closure* operator.

▶ **Definition 11.** *The formulas of* first-order logic with deterministic transitive closure (FO+DTC) *are generated by the above grammar extended with a new clause:*

$$\phi, \psi, \ldots ::= \ldots \mid \mathrm{DTC}_{\vec{x},\vec{y}}(\phi) \quad (\vec{x} \text{ and } \vec{y} \text{ are lists of variables of same length})$$

The definition of the satisfaction relation is extended with the following induction case: $\mathfrak{D} \models \mathrm{DTC}_{\vec{x},\vec{y}}(\phi)[\sigma] \iff \sigma(\vec{x})\ R^*\ \sigma(\vec{y})$ *where*

- R^* *is the reflexive transitive closure of the binary relation* $R \subseteq D^k \times D^k$*;*
- D *is the domain of* $\mathfrak{D}$ *and* $\vec{x}, \vec{y}$ *have length* k*;*
- $\vec{a}\ R\ \vec{b} \iff \mathfrak{D} \models \phi_d[\sigma + (\vec{x} \mapsto \vec{a}) + (\vec{y} \mapsto \vec{b})]$[13] *with* ϕ_d *defined as* $\phi \wedge (\forall \vec{z}.\, \phi[\vec{z}/\vec{y}] \Rightarrow \vec{z} = \vec{y})$.

▶ Remark 12. In the above definition, the relation R defined by ϕ_d is *deterministic*, i.e. it is the graph of a partial function $D^k \rightharpoonup D^k$, hence the name. Indeed, it is a "determinization" of the relation defined by ϕ.

▶ **Theorem 13** (Immerman [22])**.** *The* L *queries over totally ordered finite models are exactly those expressible in* FO+DTC*.*

3.2 An encoding of FO+DTC

Thus, it suffices to compile FO+DTC formulas, by induction, to $\mathrm{ELL}_2^{\mathrm{PP3}}$ proofs. For this purpose, it is convenient to interpret formulas with free variables as relation-valued queries:

▶ **Theorem 14.** *Let* $\phi(\vec{x})$ *be an* FO+DTC *formula with* k *free variables. Then there exists an* $\mathrm{ELL}_2^{\mathrm{PP3}}$ *proof* π_ϕ *of* $\mathtt{Inp}[\delta] \vdash !!\mathtt{Rel}_k[\delta]$ *such that, for all* $\mathfrak{D} \in \mathrm{FinMod}(\Sigma)$ *with a domain* D *of size* n*,* $\mathbf{cut}(\overline{\mathfrak{D}}, \pi_\phi[\mathtt{Fin}(n)/\delta])$ *reduces to the encoding of* $\{\vec{a} \in D^k \mid \mathfrak{D} \models \phi[\vec{x} \mapsto \vec{a}]\}$*.*

▶ **Corollary 15** (Lower bound for Theorem 7 and Theorem 10)**.** *All* FO+DTC *queries – and therefore all* L *queries – over* $\mathrm{FinMod}(\Sigma)$ *can be computed by* $\mathrm{ELL}_2^{\mathrm{PP3}}$ *proofs of* $\mathtt{Inp}_\Sigma \multimap \mathtt{Bool}$*.*

Proof. Note that $\mathtt{Rel}_0[\delta] \cong \mathtt{Bool}$, and apply a $\parr$-rule and a $\forall$-rule to the $\mathrm{ELL}_2^{\mathrm{PP3}}$ proof given by the previous theorem. ◀

[13] The new assignments for $\vec{x}$ and $\vec{y}$ override the pre-existing ones in σ.

The detailed proof of Theorem 14 is given in the full paper, in Appendix B. As stated before, it works by induction on the FO+DTC formula, the bulk of the work for the induction being the case $\phi = \mathrm{DTC}_{\vec{x},\vec{y}}(\psi)$. The remainder of the section gives a rough summary of the ideas involved.

Let $R \subseteq D^k \times D^k$, and define $\psi_R : Q \mapsto \{(x,z) \mid x = z \vee (\exists y : x\, R\, y \wedge y\, Q\, z)\}$. Then ψ_R is a monotone function over $\mathcal{P}(D^k \times D^k)$, a lattice of height $n^{2k}+1$ ($n = \mathrm{Card}(D)$). Its least fixpoint $\psi_R^{n^{2k}+1}(\emptyset)$ is exactly the reflexive transitive closure of R. To compute $\psi_R^{n^{2k}+1}$, we use an iterator of type $\mathtt{Nat} = \forall X.!(X \multimap X) \multimap !(X \multimap X)$ derived from the $\mathtt{List}[\delta]$.

But this only allows us to iterate *linear* functions. This is where we use the assumption that R is *deterministic*: if $f_R : D^k \rightharpoonup D^k$ is the partial function associated to R, then $\psi_R(Q) = \{(x,z) \mid x = z \vee (f_R(x) \text{ defined} \wedge f_R(x)\, Q\, z)\}$. In this reformulation, the existential quantifier, which was a source of non-linearity, has disappeared: now, for each (x,z), the evaluation of $(x,z) \in \psi_R(Q)$ uses Q at most once, on $(f_R(x), z)$. In the end, we manage to write a function of type $\mathtt{Rel}_{2k}[\delta] \multimap \mathtt{Rel}_{2k}[\delta]$ representing ψ_R, which we feed to the $\mathtt{Nat}$.

A not-quite-trivial step is to define a proof of $\mathtt{Ctx}[\delta], !!\mathtt{Rel}_{2k}[\delta] \vdash !!(\delta^k \multimap 1 \oplus \delta^k)$ sending a relation ϕ to the partial function associated to its determinization ϕ_d. To do so, at one point, we need to instantiate the input $\mathtt{List}[\delta]$ at the type $\delta^{k-1} \otimes (\delta^k \multimap 1 \oplus \delta^k \oplus 1)$; this is our most complicated existential witness, and it is in PP3. We refer the reader to Appendix B of the full paper again for details.

4 The upper bounds: semantic evaluation

We now give space-efficient algorithms for queries defined by proofs of $\mathtt{Inp}_\Sigma \multimap !!\mathtt{Bool}$ in ELL_2 (resp. $\mathrm{ELL}_2^{\mathrm{PP3}}$). First, we analyse the shape of such a proof, to obtain alternative definitions of the same predicates involving only MALL_2 types and proofs. This puts us in a position to evaluate our queries in a finite, effective semantics of MALL_2: the model of coherence spaces and normal functors which we recall next. Then, we quickly derive the unrestricted L^{UL} bound for Theorem 7, and finally prove L soundness for Theorem 10 thanks to a tricky combinatorial algorithm on coherence spaces.

4.1 Syntactic analysis

Purely syntactic arguments suffice to show that our ELL_2 queries can be captured by a kind of function algebra, defined below. Though it bears some similarities with Gurevich's characterization of L [17] by primitive recursion on finite models, a major difference is that our functions may take arguments which are not just domain elements (that can be coded on $O(\log n)$ bits) but also higher-order data of polynomial size in n, such as relations. Indeed, linearity serves mainly to tame the complexity in presence of higher-order features, while it is mostly meaningless on first-order data.

▶ **Definition 16.** *We define inductively, simultaneously for all $(k+1)$-tuples $(A_1, \ldots, A_k, B)$ of MALL_2 types with at most one free type variable δ, the classes of functions $\mathcal{C}(A_1, \ldots, A_k; B)$ taking as input:*

- *a closed MALL_2 type T (i.e. without free variables)*
- *a list $L = [\tau_1, \ldots, \tau_n]$ of proofs of T*
- *a k-tuple of proofs $(\rho_1, \ldots, \rho_k)$ with $\rho_i : A_i[T/\delta]$*

and returning a proof of $B[T/\delta]$ as follows:

- *if π is a proof of $A_1, \ldots, A_k \vdash B$, then*
  $[(T; L; \rho_1, \ldots, \rho_k) \mapsto \mathbf{cut}(\rho_1, \ldots \mathbf{cut}(\rho_k, \pi[T/\delta]) \ldots)] \in \mathcal{C}(A_1, \ldots, A_k; B)$

- (projection) $\Pi_i^k = [(T; L; \rho_1, \dots, \rho_k) \mapsto \rho_i] \in \mathcal{C}(A_1, \dots, A_k; A_i)$
- (composition) *if* $f_i \in \mathcal{C}(A_1, \dots, A_k; B_i)$ *for* $i \in \{1, \dots, l\}$ *and* $g \in \mathcal{C}(B_1, \dots, B_l; C)$*, then* $[(T; L; \vec{\rho}) \mapsto g(T; L; f_1(T; L; \vec{\rho}), \dots, f_l(T; L; \vec{\rho}))] \in \mathcal{C}(A_1, \dots, A_k; C)$
- (iteration) *if* $f \in \mathcal{C}(A_1, \dots, A_k; \delta \multimap B \multimap B)$*, then* $[(T; L = [\tau_1, \dots, \tau_n]; \vec{\rho}) \mapsto f(T; L; \vec{\rho})\langle\tau_1\rangle \circ \dots \circ f(T; L; \vec{\rho})\langle\tau_n\rangle] \in \mathcal{C}(A_1, \dots, A_k; B \multimap B)$ *where*
  - $\pi\langle\tau\rangle$ *is the partial application of* $\pi : T \multimap B[T/\delta] \multimap B[T/\delta]$ *to* $\tau : T$*, to produce a proof of* $B[T/\delta] \multimap B[T/\delta]$*;*
  - $\circ$ *is the composition of proofs of* $B[T/\delta] \multimap B[T/\delta]$ *seen as endomorphisms of* $B[T/\delta]$*.*

▶ **Proposition 17.** *Let* $(A_1, \dots, A_k, B)$ *be a* $(k+1)$*-tuple of* MALL_2 *types and* π *be an* ELL_2 *proof of* $\forall\delta.((!\mathtt{List}[\delta] \otimes !!A_1 \otimes \dots \otimes !!A_k) \multimap !!B)$*. Then there exists a function* $f \in \mathcal{C}(A_1, \dots, A_k; B)$ *such that for all* $\rho_i : A_i[T/\delta]$ *(*$i \in \{1, \dots, m\}$*) and* $\tau_1, \dots, \tau_n : T$*,* $\mathbf{cut}(![\tau_1, \dots, \tau_n] \otimes !!\rho_1 \otimes \dots \otimes !!\rho_k, \pi) = !!f(T; [\tau_1, \dots, \tau_n]; \rho_1, \dots, \rho_k)$ *(where the* $[\tau_1, \dots, \tau_n]$ *on the left is a Church-encoded list in* ELL_2 *of type* $\mathtt{List}[T]$*).*

Moreover, if π *is in* $\mathrm{ELL}_2^{\mathrm{PP3}}$*, then there is an inductive derivation for* f *in which all instances of the iteration scheme use a type of accumulators in* PP3*: that is, they are applied to functions in* $\mathcal{C}(\dots; \delta \multimap P \multimap P)$ *with* $P \in \mathrm{PP3}$*.*

Though the proof of this proposition presents no conceptual difficulty, it is cumbersome and so is relegated to the full paper, in Appendix C. Importantly, it is thanks to the stratification property of ELL_2 that the types involved in the function algebra can be taken in MALL_2: the argument uses the "truncation at depth 2" operation introduced in the prequel to prove Theorem 2. Note that they may still contain impredicative quantifications, making its finite interpretation essential to our approach.

▶ Remark 18. The converse also holds: one can map functions in our algebra to ELL_2 proofs.

This can now be specialized to the case $\pi : \mathtt{Inp}_\Sigma \multimap !!\mathtt{Bool}$; indeed,

$$\mathtt{Inp}_\Sigma \multimap !!\mathtt{Bool} \cong \forall\delta.\ !\mathtt{List}[\delta] \otimes !!\mathtt{C}[\delta] \otimes !!\mathtt{W}[\delta] \otimes \bigotimes_{0 \le i \le N} !!\mathtt{Rel}_{r_i}[\delta] \multimap !!\mathtt{Bool}$$

Our ELL_2-definable (resp. $\mathrm{ELL}_2^{\mathrm{PP3}}$-definable) queries can therefore be specified, equivalently, by functions in $\mathcal{C}(\mathtt{C}[\delta], \mathtt{W}[\delta], \mathtt{Rel}_{r_0}[\delta], \dots, \mathtt{Rel}_{r_N}[\delta]; \mathtt{Bool})$. The next step is to evaluate these functions in the coherence space model.

4.2 The finite semantics of second-order MALL in coherence spaces

We recall key facts about the denotational model of MALL_2 in which we will carry out our semantic evaluation. A comprehensive introduction to this model for propositional MALL may be found in [15], and the extension to MALL_2 is taken from the prequel [32].

In this semantics, a formula/type is interpreted as a *coherence space*: an undirected reflexive graph, i.e. a pair $X = (|X|, \frown_X)$ of a set $|X|$ – customarily called the *web* of X – and a symmetric and reflexive relation $\frown_X \subseteq |X| \times |X|$ – its *coherence relation*. Elements $x, y \in |X|$ are called *coherent* when $x \frown_X y$. A *clique* is a subset of pairwise coherent elements of $|X|$; we write $c \sqsubset X$ when c is a clique of X. The denotation of a closed type A is a coherence space, and a proof/program $\pi : A$ is interpreted as a clique $[\![\pi]\!] \sqsubset [\![A]\!]$.

$[\![A]\!]$ is defined by induction on A, the connectives $\otimes, \parr, \&, \oplus, (-)^\perp$ being mapped to operations on coherence spaces. The base case depends on an assignment of type variables. So, if A has n type variables, $[\![A]\!]$ is actually a map from n-tuples of coherence spaces to coherence spaces. Similarly, $[\![\pi]\!]$ also depends on such an assignment, and one should

write $[\![\pi]\!](X_1, \ldots, X_n) \sqsubset [\![A]\!](X_1, \ldots, X_n)$. To extend the semantics to MALL_2, we interpret quantifiers as sending such "$(n+1)$-parameter spaces" to "n-parameter spaces". The following proposition sums up the properties that will be necessary for our purposes.

▶ **Proposition 19** ([32]). *Let A be a MALL_2 type with a single free type variable.*
- *$[\![A]\!](X)$ is* finite*, with size polynomial in the size of X when A is fixed.*
- *$[\![\pi]\!](X)$ can be computed in logarithmic space when $\pi : A$ is fixed.*

Finally, we need to recall the semantic counterpart of cut-elimination, that is, composition of morphisms. A first remark is that $|X \multimap Y| = |X| \times |Y|$. So a clique $c \sqsubset X \multimap Y$ can in fact be seen as a *binary relation* $c \subseteq |X| \times |Y|$. The composition of c with some $c' \sqsubset Y \multimap Z$, seen as morphisms of coherence spaces, is then none other than their *relational composition*. Additionally, the coherence relation ensures the well-known fact that:

▶ **Proposition 20.** *Let $c \sqsubset X \multimap Y$, $c' \sqsubset Y \multimap Z$, $x \in |X|$ and $z \in |Z|$. Then there exists* at most one *$y \in |Y|$ such that $(x, y) \in c$ and $(y, z) \in c'$.*

4.3 The unrestricted case: an unambiguous logarithmic space bound

In this subsection and in the next one, we abbreviate for convenience $[\![\mathtt{Fin}(n)]\!]$, i.e. the n-vertex coherence space with no edges, as $\mathtt{Fin}(n)$. So, if A is a MALL_2 type with a single variable δ, then $[\![A[\mathtt{Fin}(n)/\delta]]\!] = [\![A]\!](\mathtt{Fin}(n))$. Our main theorem here is:

▶ **Theorem 21.** *Let $f \in \mathcal{C}(A_1, \ldots, A_k; B)$. Then $[\![f(T; L; \rho_1, \ldots, \rho_k)]\!]$ is determined by $[\![T]\!]$, $[\![L]\!] = [[\![\tau_1]\!], \ldots, [\![\tau_n]\!]]$ (where $L = [\tau_1, \ldots, \tau_n]$) and $[\![\rho_1]\!], \ldots, [\![\rho_k]\!]$. Furthermore, when f is fixed, $[\![f(T; L; \rho_1, \ldots, \rho_k)]\!]$ can be computed from these denotations in L^{UL}.*

Proof. By structural induction on Definition 16; the first part is an immediate consequence of the functoriality/compositionality of $[\![-]\!]$, so we focus on the complexity. We take care of the base case, where the function comes from a proof $\pi : (A_1, \ldots, A_k \vdash B)$, with Proposition 19 and the fact that relational composition is in L. For the composition scheme, we use the closure of[14] L^{UL} under composition. The iteration scheme is handled by Lemma 22 below. ◀

▶ **Lemma 22.** *Let A be a MALL_2 type with a single type variable. Given $n, k \in \mathbb{N}$, $f_1, \ldots, f_k \sqsubset [\![A \multimap A]\!](\mathtt{Fin}(n))$ and $(u, v) \in |[\![A]\!](\mathtt{Fin}(n))|^2$, whether $(u, v) \in (f_k \circ \ldots \circ f_1)$ can be decided in UL (in the size of the input, which is polynomial[15] in n and k).*

Proof. Thanks to Proposition 20, if $v \in (f_k \circ \ldots \circ f_1)(\{u\})$ then there is a *unique* sequence $u_0 = u, u_1, \ldots u_k = v$ such that $u_{i+1} \in f(\{u_i\})$. We successively guess the u_i; at each point, we need only store (u_i, u_{i+1}) to check its presence in f_i. This can be done by a UL Turing machine because each u_i can be stored in space $O(\log n)$: indeed, $|[\![A]\!](\mathtt{Fin}(n))|$ has cardinality polynomial in n (Proposition 19) and there is a natural representation of its points of using $O(1)$ variables in $|\mathtt{Fin}(n)| = \{1, \ldots, n\}$, see [32, Section IV.D]. (Notice that we do not even make use of the coherence relation of $[\![A]\!](\mathtt{Fin}(n))$; its mere existence ensures that the naive NL algorithm is actually UL.) ◀

The upper bound of Theorem 7 follows immediately from Theorem 21 together with:

[14] Strictly speaking, L^{UL} denotes a class of decision problems, and it is the associated class of function problems $\mathsf{FL}^{\mathsf{UL}}$ which is closed under composition (the usual proof for FL relativizes).

[15] The f_i are cliques in the graph $[\![A \multimap A]\!](\mathtt{Fin}(n))$, which has a polynomial size in n by virtue of Proposition 19. Note that we always have $k \geq 1$.

▶ **Lemma 23.** *Let* $\mathfrak{D} \in \mathrm{FinMod}(\Sigma)$*. Its* ELL_2 *encoding* $\overline{\mathfrak{D}} : \mathtt{Inp}_\Sigma[\mathtt{Fin}(n)]$ *(*n *is the domain size of* $\mathfrak{D}$*) contains* MALL_2 *proofs of* $\mathtt{C}[\mathtt{Fin}(n)]$*,* $\mathtt{W}[\mathtt{Fin}(n)]$ *and* $\mathtt{Rel}_{r_i}[\mathtt{Fin}(n)]$ *(*$i \in \{1, \dots, N\}$*).*

The denotations of these proofs in the coherence space model can all be computed in L*.*

4.4 Iterations in deterministic log space for low-complexity types

As can be seen in the proof of Theorem 21, the single crucial point where the complexity of evaluating a query does not seem to fall squarely in L is Lemma 22. By putting the complexity of this iterated composition problem in L when $A \in \mathrm{PP3}$, we will get the L soundness result for Theorem 10.

A first remark is that for $A \in \mathrm{PP3}$, $A[\mathtt{Fin}(n)/\delta] \cong \mathtt{Fin}(P(n)) \otimes (\mathtt{Fin}(Q(n)) \multimap \mathtt{Fin}(R(n)))$ where P, Q, R are polynomials with integer coefficients. The goal becomes to show:

▶ **Theorem 24.** *Let* $A \cong \mathtt{Fin}(m) \otimes (\mathtt{Fin}(n) \multimap \mathtt{Fin}(p))$ *for some* $m, n, p \in \mathbb{N}$*. Given* $f_1, \dots, f_k \sqsubset [\![A \multimap A]\!]$ *and* $(u, v) \in |[\![A]\!]|^2$*, whether* $(u, v) \in (f_k \circ \dots \circ f_1)$ *can be decided in* L*.*

At this point, the proofs start to involve tricky combinatorics on coherence spaces, so this final section of the paper is written for readers familiar with the coherence space model of MALL (but not necessarily its extension to MALL_2). For instance we will often identify cliques $f \sqsubset A \multimap B$ with linear maps from the cliques of A to the cliques of B.

We start with a lemma solving the case $m = 1$, generalizing the example given at the end of the introduction.

▶ **Lemma 25.** *Let* $A = \mathtt{Fin}(n) \multimap \mathtt{Fin}(p)$*,* $f_1, \dots, f_k \sqsubset A \multimap A$*,* $\nu, \nu' \in |\mathtt{Fin}(n)|$ *and* $\pi \in |\mathtt{Fin}(p)|$*. There exists* at most one π' *such that* $(\nu', \pi') \in (f_k \circ \dots \circ f_1)(\{(\nu, \pi)\})$*.*

Furthermore, there is a logarithmic space algorithm taking $n, p, f_1, \dots, f_k, \nu, \nu', \pi$ *as inputs which decides whether* π' *exists and, if so, finds it.*

Proof. Consider the adjoint maps $f_i^\perp \sqsubset (\mathtt{Fin}(n) \otimes \mathtt{Fin}(p)^\perp \multimap \mathtt{Fin}(n) \otimes \mathtt{Fin}(p)^\perp)$. The graph $\mathtt{Fin}(n) \otimes \mathtt{Fin}(p)^\perp$ has n connected components, which are all cliques (of size p). These $f_i^\perp$ send cliques to (possibly empty) cliques, so for $j \in |\mathtt{Fin}(n)|$, $f_i^\perp(\{j\} \times |\mathtt{Fin}(p)|)$ is either (1) empty or (2) included in some $\{l\} \times |\mathtt{Fin}(p)|$, for l uniquely determined by j. This defines partial maps $\widehat{f_i^\perp} : |\mathtt{Fin}(n)| \rightharpoonup |\mathtt{Fin}(n)|$: in case (1) $\widehat{f_i^\perp}(j)$ is undefined, in case (2) $\widehat{f_i^\perp}(j) = l$.

This allows us to perform a backwards iteration: we define $\nu_k = \nu'$ and, for $i = k, \dots, 1$, $\nu_{i-1} = \widehat{f_i^\perp}(\nu_i)$; ν_0 can be computed in logarithmic space. If ν_0 is undefined or $\nu_0 \neq \nu$, then π' does not exist: we return false.

Otherwise, let us restrict each i-th intermediate $\mathtt{Fin}(n) \multimap \mathtt{Fin}(p)$ to the connected component corresponding to ν_i, and take the corresponding sub-cliques: for $i = 1, \dots, k$, $f_i' = f_i \cap ((\{\nu_{i-1}\} \times |\mathtt{Fin}(p)|) \times (\{\nu_i\} \times |\mathtt{Fin}(p)|))$. Then either $(f_k' \circ \dots \circ f_1')(\{\pi\})$ is empty, and π' does not exist; or it contains a single element, which is then π'.

Each ν_i is computable in logarithmic space, so $(f_1', \dots, f_k')$ also is; additionally, the computation of $(f_k' \circ \dots \circ f_1')(\{\pi\})$ from $(f_1', \dots, f_k')$ and π only needs to store a single point of $\mathtt{Fin}(p)$ in working memory, because the cliques of the latter are subsingletons. Since L is closed under L-reductions, we are done. (Making the interactive composition explicit results in a quadratic time algorithm.) ◀

We would like to L-reduce the problem to the case $m = 1$, by determining the projection to $|\mathtt{Fin}(m)|$ of the unique "path" of $k + 1$ points corresponding to a point of the clique $f_k \circ \dots \circ f_1$. This would involve an iteration analogous to the previous proof, but forwards instead of backwards.

But the image $f_i(\{j\} \times |\mathtt{Fin}(n) \multimap \mathtt{Fin}(p)|)$ is *not necessarily connected*, because $\{j\} \times |\mathtt{Fin}(n) \multimap \mathtt{Fin}(p)|$ is not a clique (though $\mathtt{Fin}(n) \multimap \mathtt{Fin}(p)$ is a connected graph, it is not complete). So one cannot guarantee that this image is included in some $\{l\} \times |\mathtt{Fin}(n) \multimap \mathtt{Fin}(p)|$. An explicit counter-example is the interpretation of the term $\lambda(x \otimes g).\,((g\,x) \otimes \ldots)$ when $m = n$: in some sense, the first component of the output depends on *both x and g being known*, not only x. However, knowing x is enough to determine what argument will be fed to g (x itself, in this example). The intuitive idea is to propagate this backwards.

The following lemma ensures that we can always either carry on with the forwards iteration or start the backwards propagation (Π_1 (resp. Π_2) is the projection on the first (resp. second) component):

▶ **Lemma 26.** *Let $c \sqsubset A \parr B$ be a non-empty clique. Then $\Pi_1(c)$ is included in a connected component of A, or (non-exclusively) $\Pi_2(c)$ is included in a connected component of B.*

Proof sketch. If $\Pi_1(u)$ and $\Pi_1(v)$ are in different connected components for $u, v \in c$, then $\Pi_2(u)$ and $\Pi_2(v)$ are coherent or equal, and all other $\Pi_2(w)$ are coherent or equal to at least one of them: $\Pi_2(c)$ is connected with diameter ≤ 3. ◀

Proof of Theorem 24. We write $A = \mathtt{Fin}(m) \otimes B$ and $B = \mathtt{Fin}(n) \multimap \mathtt{Fin}(p)$. If $n = 0$, $A \cong 0$ and the problem is trivial and if $n = 1$, $A \cong \mathtt{Fin}(m) \otimes \mathtt{Fin}(p)$, so a simple forward propagation solves the problem. From now on, we thus assume that $n > 1$, which makes B connected. Let $f_1, \ldots, f_k \sqsubset A \multimap A$, $(\mu, (\nu, \pi)) \in |A|$ and $(\mu', (\nu', \pi')) \in |A|$. The goal is to decide, in logarithmic space, whether $(\mu', (\nu', \pi')) \in (f_k \circ \ldots \circ f_1)(\{(\mu, (\nu, \pi))\})$.

Let $\mu_0 = \mu$. If the clique $f_1(\{(\mu, (\nu, \pi))\})$ is empty, then the answer is negative; else, let $\{\mu_1\} \times |B|$ be the connected component containing it. For $1 \leq i < k$, assuming that μ_i is defined, then $f_{i+1}(\{\mu_i\} \times |B|)$ is either:

- empty, and the answer is negative;
- non-empty and contained in some $\{\mu_{i+1}\} \times |B|$ – this defines $\mu_{i+1} \in |\mathtt{Fin}(m)|$ uniquely;
- non-empty and disconnected.

Let $f_i^{\ddagger} = f_i \cap ((\{\mu_{i-1}\} \times |B|) \times (\{\mu_i\} \times |B|))$ for all $i \geq 1$ for which μ_i is defined. If the iteration reaches $i = k$, this means that $(f_1^{\ddagger}, \ldots, f_k^{\ddagger})$ can be computed in logarithmic space, and as in Lemma 25 we can use this to L-reduce the problem to the case $m = 1$, so we are done. If it aborts because of emptiness, then the algorithm can immediately return false.

The remaining case is the last item above. Suppose that μ_{i+1} is undefined because of disconnectedness. Let $f_{i+1}^{\dagger} = f_{i+1} \cap ((\{\mu_i\} \times |B|) \times |A|)$; it can be seen as a clique $f_{i+1}^{\dagger} \sqsubset B \multimap A = B^{\perp} \parr A$, with $B^{\perp} = \mathtt{Fin}(n) \otimes \mathtt{Fin}(p)^{\perp}$. The assumption that $\Pi_2(f_{i+1}^{\dagger}) = f_{i+1}(\{\mu_i\} \times |B|)$ is non-empty and disconnected entails, by Lemma 26, that $\Pi_1(f_{i+1}^{\dagger})$ is connected. In other words $\Pi_1(f_{i+1}^{\dagger}) \subseteq \{\nu''\} \times |\mathtt{Fin}(p)|$ for some ν''.

Let us apply the algorithm of Lemma 25 to the inputs $n, p, f_1^{\ddagger}, \ldots, f_i^{\ddagger}, \nu, \nu'', \pi$. This can be done in logarithmic space, and the subroutine either raises a failure or gives us some $\pi'' \in |\mathtt{Fin}(p)|$. In the former case, we can return false; in the latter, we know that $(f_k \circ \ldots \circ f_1)(\{(\mu, (\nu, \pi))\}) = (f_k \circ \ldots \circ f_{i+1})(\{(\mu_i, (\nu'', \pi''))\})$. So all we have to do is to tail-recurse on a suffix of the original input; to implement this in L, it suffices to keep a counter indicating what the current suffix is. This is a strict suffix, because μ_1 is always defined by construction (see above); therefore, our algorithm terminates, while maintaining a logarithmic working space. ◀

▶ Remark 27. $\mathtt{Fin}(m) \otimes (\mathtt{Fin}(n) \multimap \mathtt{Fin}(p)) \cong \bigoplus_{i=1}^{m} \&_{j=1}^{n} \bigoplus_{k=1}^{p} 1$, and such a bicartesian MALL formula can be seen as a game where Player and Opponents alternate choices of branches. Linear implication consists in playing two games in parallel. Morally, Lemma 26 says: if it is your turn to play on both boards, then you must make a choice; and our L algorithm is mostly about scheduling a set of strategies interacting together.

5 Perspective: unrestricted L upper bound through game semantics?

In the extensional completeness proof, strikingly, the *determinism* of a relation corresponds exactly to the *linearity* of its pre-composition operator. This is one reason for which we believe that our class of queries in ELL_2 is exactly L (Conjecture 1) – or at least, that it is strictly containted in NL which corresponds to first-order logic with general transitive closure [22]. Thus, our L^{UL} bound is likely not optimal: it is widely believed that $\mathsf{UL} = \mathsf{NL}$ [34, 33].

To bring down the complexity of the bottleneck – namely the iterated composition – from UL to L, bridging the intuitions of Remark 27 with a proper game semantics of full MALL_2 might be key. In this direction, it is known that the points of the web of a (hyper)coherence space can be seen as external positions of a game [10, 5, 30, 31]. With this point of view, the uniqueness of the intermediate points in the iteration of Lemma 22 reflects the determinism of an underlying interaction which reaches those final positions.

References

1 Serge Abiteboul and Gerd Hillebrand. Space usage in functional query languages. In Gerhard Goos, Juris Hartmanis, Jan Leeuwen, Georg Gottlob, and Moshe Y. Vardi, editors, *Database Theory — ICDT '95*, volume 893, pages 439–454. Springer Berlin Heidelberg, Berlin, Heidelberg, 1995. doi:10.1007/3-540-58907-4_33.

2 Patrick Baillot. On the expressivity of elementary linear logic: Characterizing Ptime and an exponential time hierarchy. *Information and Computation*, 241:3–31, April 2015. doi:10.1016/j.ic.2014.10.005.

3 Patrick Baillot, Erika De Benedetti, and Simona Ronchi Della Rocca. Characterizing polynomial and exponential complexity classes in elementary lambda-calculus. *Information and Computation*, 261:55–77, August 2018. doi:10.1016/j.ic.2018.05.005.

4 Patrick Baillot and Alexis Ghyselen. Combining Linear Logic and Size Types for Implicit Complexity. In *27th EACSL Annual Conference on Computer Science Logic (CSL 2018)*, pages 9:1–9:21, 2018. doi:10.4230/LIPIcs.CSL.2018.9.

5 Pierre Boudes. Projecting Games on Hypercoherences. In *Automata, Languages and Programming*, volume 3142, pages 257–268. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. doi:10.1007/978-3-540-27836-8_24.

6 Ugo Dal Lago and Martin Hofmann. Realizability models and implicit complexity. *Theoretical Computer Science*, 412(20):2029–2047, April 2011. doi:10.1016/j.tcs.2010.12.025.

7 Ugo Dal Lago and Ulrich Schöpp. Computation by interaction for space-bounded functional programming. *Information and Computation*, 248:150–194, June 2016. doi:10.1016/j.ic.2015.04.006.

8 Vincent Danos and Jean-Baptiste Joinet. Linear logic and elementary time. *Information and Computation*, 183(1):123–137, May 2003. doi:10.1016/S0890-5401(03)00010-5.

9 Vincent Danos and Laurent Regnier. Reversible, irreversible and optimal λ-machines. *Theoretical Computer Science*, 227(1):79–97, September 1999. doi:10.1016/S0304-3975(99)00049-3.

10 Thomas Ehrhard. Parallel and serial hypercoherences. *Theoretical Computer Science*, 247(1):39–81, September 2000. doi:10.1016/S0304-3975(00)00173-0.

11 Ronald Fagin. *Contributions to the model theory of finite structures*. PhD thesis, University of California, Berkeley, 1973.

12 Jean-Yves Girard. The system F of variable types, fifteen years later. *Theoretical Computer Science*, 45:159–192, January 1986. doi:10.1016/0304-3975(86)90044-7.

13 Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–101, January 1987. doi:10.1016/0304-3975(87)90045-4.

14 Jean-Yves Girard. Geometry of Interaction 1: Interpretation of System F. In R. Ferro, C. Bonotto, S. Valentini, and A. Zanardo, editors, *Studies in Logic and the Foundations of Mathematics*, volume 127 of *Logic Colloquium '88*, pages 221–260. Elsevier, January 1989.

15 Jean-Yves Girard. Linear logic: its syntax and semantics. In Jean-Yves Girard, Yves Lafont, and Laurent Regnier, editors, *Advances in Linear Logic*, volume 222 of *London Mathematical Society Lecture Notes*. Cambridge University Press, 1995.

16 Jean-Yves Girard. Light Linear Logic. *Information and Computation*, 143(2):175–204, June 1998. doi:10.1006/inco.1998.2700.

17 Yuri Gurevich. Algebras of feasible functions. In *24th Annual Symposium on Foundations of Computer Science (FOCS 1983)*, pages 210–214, Tucson, AZ, USA, November 1983. doi:10.1109/SFCS.1983.5.

18 Gerd G. Hillebrand. *Finite Model Theory in the Simply Typed Lambda Calculus*. PhD thesis, Brown University, Providence, RI, USA, 1994.

19 Gerd G. Hillebrand and Paris C. Kanellakis. Functional Database Query Languages As Typed Lambda Calculi of Fixed Order (Extended Abstract). In *Proceedings of the Thirteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS '94, pages 222–231, New York, NY, USA, 1994. ACM. doi:10.1145/182591.182615.

20 Gerd G. Hillebrand and Paris C. Kanellakis. On the Expressive Power of Simply Typed and Let-Polymorphic Lambda Calculi. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, pages 253–263. IEEE Computer Society, 1996. doi:10.1109/LICS.1996.561337.

21 Gerd G. Hillebrand, Paris C. Kanellakis, and Harry G. Mairson. Database Query Languages Embedded in the Typed Lambda Calculus. *Information and Computation*, 127(2):117–144, June 1996. doi:10.1006/inco.1996.0055.

22 Neil Immerman. Languages that Capture Complexity Classes. *SIAM Journal on Computing*, 16(4):760–778, August 1987. doi:10.1137/0216051.

23 Neil Immerman. Nondeterministic Space is Closed under Complementation. *SIAM Journal on Computing*, 17(5):935–938, October 1988. doi:10.1137/0217058.

24 Donald E. Knuth. Mathematics and Computer Science: Coping with Finiteness. *Science*, 194(4271):1235–1242, 1976. doi:10.1126/science.194.4271.1235.

25 Lars Kristiansen. Higher Types, Finite Domains and Resource-bounded Turing Machines. *Journal of Logic and Computation*, 22(2):281–304, April 2012. doi:10.1093/logcom/exq009.

26 Daniel Leivant. Reasoning about functional programs and complexity classes associated with type disciplines. In *24th Annual Symposium on Foundations of Computer Science (FOCS 1983)*, pages 460–469, Tucson, AZ, USA, November 1983. doi:10.1109/SFCS.1983.50.

27 Harry G. Mairson. A simple proof of a theorem of Statman. *Theoretical Computer Science*, 103(2):387–394, September 1992. doi:10.1016/0304-3975(92)90020-G.

28 Damiano Mazza. Simple Parsimonious Types and Logarithmic Space. In *24th EACSL Annual Conference on Computer Science Logic (CSL 2015)*, pages 24–40, 2015. doi:10.4230/LIPIcs.CSL.2015.24.

29 Damiano Mazza and Kazushige Terui. Parsimonious Types and Non-uniform Computation. In *Automata, Languages, and Programming*, Lecture Notes in Computer Science, pages 350–361. Springer, Berlin, Heidelberg, July 2015. doi:10.1007/978-3-662-47666-6_28.

30 Paul-André Melliès. Sequential algorithms and strongly stable functions. *Theoretical Computer Science*, 343(1):237–281, October 2005. doi:10.1016/j.tcs.2005.05.015.

31 Paul-André Melliès. On dialogue games and coherent strategies. In Simona Ronchi Della Rocca, editor, *Computer Science Logic 2013 (CSL 2013)*, pages 540–562, 2013. doi:10.4230/LIPIcs.CSL.2013.540.

32 Lê Thành Dũng Nguyễn. Around finite second-order coherence spaces. *CoRR*, abs/1902.00196, 2019. arXiv:1902.00196.

33 A. Pavan, Raghunath Tewari, and N. V. Vinodchandran. On the power of unambiguity in log-space. *computational complexity*, 21(4):643–670, December 2012. doi:10.1007/s00037-012-0047-3.

34 K. Reinhardt and E. Allender. Making Nondeterminism Unambiguous. *SIAM Journal on Computing*, 29(4):1118–1131, January 2000. doi:10.1137/S0097539798339041.

35 Walter L. Ruzzo, Janos Simon, and Martin Tompa. Space-bounded hierarchies and probabilistic computations. *Journal of Computer and System Sciences*, 28(2):216–230, April 1984. doi:10.1016/0022-0000(84)90066-7.

36 Ulrich Schöpp. Stratified Bounded Affine Logic for Logarithmic Space. In *22nd Annual IEEE Symposium on Logic in Computer Science (LICS 2007)*, pages 411–420, July 2007. doi:10.1109/LICS.2007.45.

37 Richard Statman. The typed λ-calculus is not elementary recursive. *Theoretical Computer Science*, 9(1):73–81, July 1979. doi:10.1016/0304-3975(79)90007-0.

38 Kazushige Terui. Proof Nets and Boolean Circuits. In *19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings*, pages 182–191, 2004. doi:10.1109/LICS.2004.1319612.

39 Kazushige Terui. Semantic Evaluation, Intersection Types and Complexity of Simply Typed Lambda Calculus. In *23rd International Conference on Rewriting Techniques and Applications (RTA'12)*, pages 323–338, 2012. doi:10.4230/LIPIcs.RTA.2012.323.

Automatic Semigroups vs Automaton Semigroups

Matthieu Picantin
IRIF UMR 8243 CNRS & Univ Paris Diderot, 75013 Paris, France
picantin@irif.fr

Abstract

We develop an effective and natural approach to interpret any semigroup admitting a special language of greedy normal forms as an automaton semigroup, namely the semigroup generated by a Mealy automaton encoding the behaviour of such a language of greedy normal forms under one-sided multiplication. The framework embraces many of the well-known classes of (automatic) semigroups: free semigroups, free commutative semigroups, trace or divisibility monoids, braid or Artin–Tits or Krammer or Garside monoids, Baumslag–Solitar semigroups, etc. Like plactic monoids or Chinese monoids, some neither left- nor right-cancellative automatic semigroups are also investigated, as well as some residually finite variations of the bicyclic monoid. It provides what appears to be the first known connection from a class of automatic semigroups to a class of automaton semigroups. It is worthwhile noting that, "being an automatic semigroup" and "being an automaton semigroup" become dual properties in a very automata-theoretical sense. Quadratic rewriting systems and associated tilings appear as the cornerstone of our construction.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects; Theory of computation → Rewrite systems

Keywords and phrases Mealy machine, semigroup, rewriting system, automaticity, self-similarity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.124

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version A full version of the paper [52] is available at `https://arxiv.org/abs/1609.09364`.

Funding This work was partially supported by the French *Agence Nationale pour la Recherche*, through the project MealyM ANR-JS02-012-01.

1 Introduction

The half century long history of the interactions between (semi)group theory and automata theory went through a pivotal decade from the mid-eighties to the mid-nineties. Contemporaneously but independently, two new theories truly started to develop and thrive: automaton (semi)groups on the one hand with the works of Aleshin [2, 3] and Grigorchuk [27, 28] and the book [47], and automatic (semi)groups on the other hand with the work of Cannon and Thurston and the book [24]. We refer to [55] for a clear and short survey on the known interactions between groups and automata. A deeper and more extended survey by Bartholdi and Silva can be found out in two chapters [7, 8] of the forthcoming AutoMathA handbook. We can refer to [14, 45] for automaton semigroups and to [17, 33] for automatic semigroups.

Remote siblings. As their very name indicates, automaton (semi)groups and automatic (semi)groups share a same defining object: the *automaton* or the letter-to-letter transducer in this case. Beyond this common origin, these two topics until now happened to remain largely distant both in terms of community and in terms of tools or results. Typically, any paper on one or the other topic used to contain a sentence like "it should be emphasised that, despite their similar names, the notions of automaton (semi)groups are entirely separate from

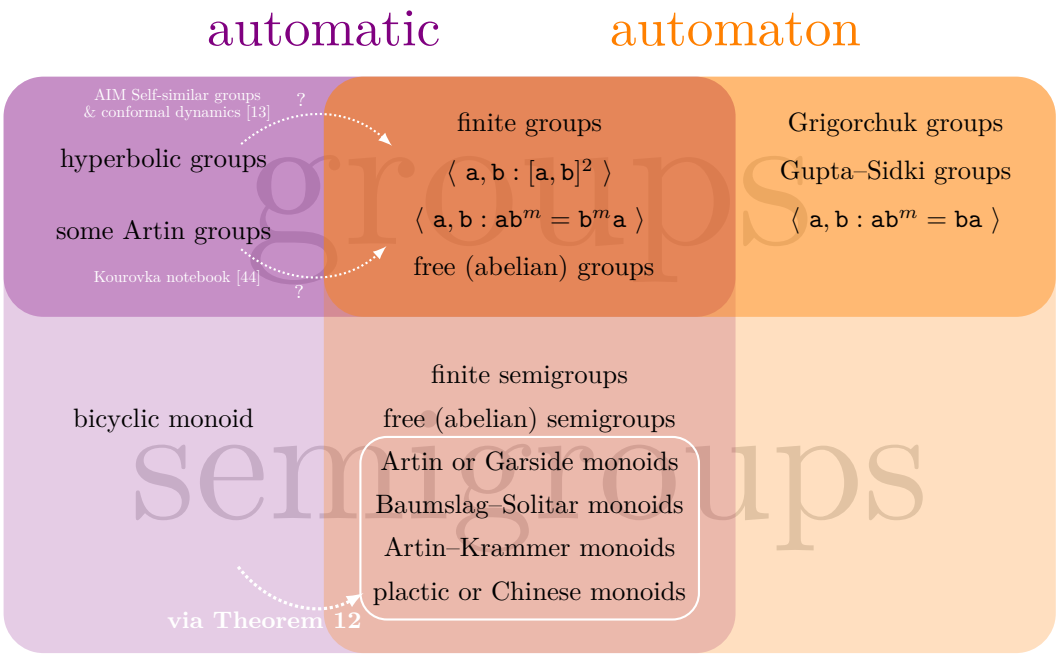



Figure 1 The big picture: comparing the classes of automatic *vs* automaton (semi)groups.

the notions of automatic (semi)groups". This was best evidenced by the above-mentioned valuable handbook chapter [7] which splits into exactly two sections (automatic groups and automaton groups) without any reference between one and the other appearing explicitly.

Related open problems. A significant problem is to recognise whether a given (semi)group is self-similar, that is, an automaton (semi)group, see Figure 1. Amongst the thirty-odd listed problems from [13], we can pick the one with the number 1.1:

> **Problem A.** It seems quite difficult to show whether a given group is self-similar. Are Gromov hyperbolic groups self-similar? Find obstructions to self-similarity.

Amongst the unsolved problems in group theory from the Kourovka Notebook [44], the one (with number 16.84) asked by Sushchanskii (see also [40, 51]) can be formulated as follows:

> **Problem B.** Is the n-strand braid group $\mathbf{B}_n$ a subgroup of some automaton group?

All these questions can be meaningfully rephrased in terms of semigroups or monoids.

Our contributions. The aim here is to establish a possible connection between being an automatic semigroup and being an automaton semigroup. Preliminary observations are that these classes intersect non trivially and that neither is included in the other (see Figure 1). Like the Grigorchuk group for instance, many automaton groups are infinite torsion groups, hence cannot be automatic groups. By contrast, it is an open question whether every automatic group is an automaton group. The latter is related to the question whether every automatic group is residually finite, which remains open despite the works by Wise [58] and Elder [23]. Like the bicyclic monoid, some automatic semigroups are not residually finite, hence cannot be automaton semigroups (see [14] for instance). As for the intersection, we know that at least finite semigroups, free semigroups (of rank at least 2, see [11, 12]), free abelian semigroups happen to be both automatic semigroups and automaton semigroups.

We propose here a new and natural way to interpret algorithmically each semigroup from a wide class of automatic semigroups – encompassing all the above-mentioned classes – as an automaton semigroup (Theorem 12). Furthermore, it is worthwhile noting that, in all these cases, "being an automatic semigroup" and "being an automaton semigroup" become dual properties in a very automata-theoretical sense (Corollary 15).

Occurring as the very first bridge between two hitherto irreconcilable research areas, Theorem 12 allows us to also provide a (more than) positive answer to the monoidal version of Problem **B**. While the n-strand braid monoid $\mathbf{B}^1_{n+}$ is the paradigmatic example of an automatic monoid from [24, Chapter 9], Theorem 12 implies that $\mathbf{B}^1_{n+}$ is (not only a submonoid of) an automaton monoid as well, see Example 23. From this significant milestone arise various new questions and perspectives, that will hopefully pervade both research fields.

Organisation. The structure of the paper is as follows. As a simple preliminary, Section 2 illustrates in a deliberately informal manner how a single Mealy automaton can be used in order to define both self-similar structures and automatic structures (via a principle of duality). In Section 3, we set up the notations for Mealy automata and recall necessary notions of dual automata, cross-diagrams, and self-similar structures. In Section 4, we recall basics about normal forms and automatic structures, and we give necessary notions of quadratic normalisations and square-diagrams. Section 5 is devoted to our main results (Theorem 12 and Corollary 15), while Section 6 finally gathers several carefully selected examples, counterexamples, and open problems.

2 A preliminary example

As their very name indicates, automaton (semi)groups and automatic (semi)groups share a same defining object. In both cases, a *Mealy automaton* (see Definition 1) basically transforms words into words.

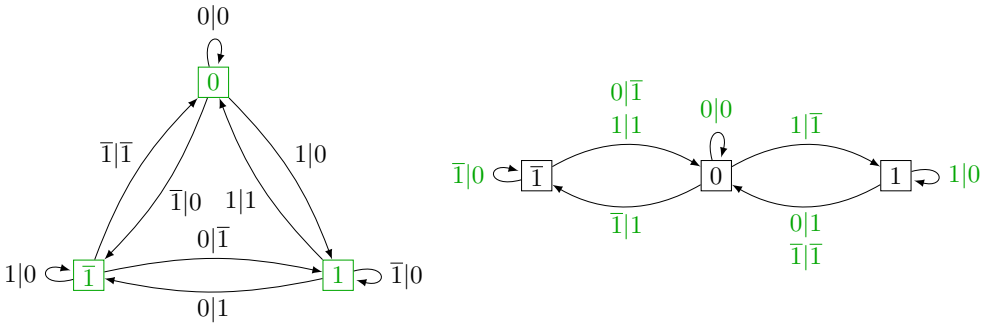


Figure 2 Two (dual) Mealy automata: the left-hand one computing the division by 3 in base 2 (most significant digit first) *vs* the right-hand one computing the multiplication by 2 in base 3 (least significant digit first).

The Mealy automaton displayed on Figure 2 (left) is some signed-digit version of one of the most classical examples of a transducer (see [53, Prologue] for a delightfully alternate history). Signed-digit numeration systems [4, 18] are not the topic, now they provide a special opportunity to illustrate our purpose. When starting from the state 0 and reading any binary word or any $\{\overline{1}, 0, 1\}$-word $\mathbf{u}$ (most significant digit first), it computes the division by 3 in base 2 by outputting the (quotient) $\{\overline{1}, 0, 1\}$-word $\mathbf{v}$ (most significant digit first) satisfying

$$(\mathbf{u})_2 = 3 \times (\mathbf{v})_2 + f \qquad (\%)$$

where the (remainder) $f \in \{\overline{1}, 0, 1\}$ corresponds to the arrival state of the run, and where $(\mathbf{w})_b$ denotes by convention the number that is represented by $\mathbf{w}$ in base b.

For the current preliminary section, let us now focus on this basic example and consider the two different viewpoints described as follows.

On the one hand, it seems natural to consider the set of those functions (from $\{\overline{1}, 0, 1\}$-words to $\{\overline{1}, 0, 1\}$-words) thus associated with each state, then to compose them with each other, and finally to study the (semi)group which is generated by such functions.

For instance, the function $\mathbf{u} \mapsto \mathbf{v}$ associated with the state 0 (satisfying Equation (%)) can be squared, cubed, and so on, to obtain functions, which can be again interpreted as the division by $9, 27, \ldots$ (in base 2 with most significant digit first), or can be composed with the functions induced by the other two states. The generated semigroup happens to be the rank 3 free semigroup $\{\overline{1}, 0, 1\}^+$ (provided that the three states and their induced functions are identified). This simple idea coincides with the notion of automaton (semi)groups or self-similar structures (see Definition 2). With this crucial standpoint, we can compute (semi)group operations by manipulating the corresponding Mealy automaton (see [5, 7, 35, 46]), and hopefully foresee some combinatorial and dynamical properties by examining its shape (see [6, 9, 10, 19, 25, 26, 34, 36, 37, 38, 51, 56] for instance).

On the other hand, it may be also natural to simply iterate the runs. The starting language is again over the (input/output) alphabet, now the images of the transformations are some languages over the stateset.

For instance, restarting again from the state 0, the previously output word $\mathbf{v}$ (satisfying Equation (%)) can be read in turn, and so on. The successive arrival states can be then collected and concatenated in order to obtain here the decomposition of $(\mathbf{u})_2$ in base 3 (least significant digit first). The whole process has thus inherently a quadratic time complexity.

This second idea coincides with the fundamental notion of automatic (semi)groups (see Definition 3 and [7, 17, 24, 33, 51]), for which Mealy automata can compute normal forms.

To conclude this preliminary section, let us mention that states and letters of any Mealy automaton play a symmetric role, and that several properties can be beneficially derived from the so-called *dual (Mealy) automaton*, obtained by exchanging the stateset and the alphabet (see [36] for an overview).

For instance, Figure 2 displays a pair of dual automata. While the left-hand automaton allows to compute the division by 3 in base 2 (most significant digit first) as we have seen just above, its dual automaton (right) essentially computes the multiplication by 2 in base 3 (least significant digit first). More precisely, its state 0 induces the function $x \mapsto 2 \times x$ on $\mathbb{Z}$, while its states $\overline{1}$ and 1 induce the functions $x \mapsto 2 \times x - 1$ and $x \mapsto 2 \times x + 1$ respectively: they together generate the semigroup $\langle\ \overline{1}, 0, 1 : 01 = 1\overline{1}, 0\overline{1} = \overline{1}1\ \rangle_+$. Let us also mention that the induced functions happen to be invertible and to generate a group which is isomorphic with the so-called Baumslag–Solitar group $\mathbf{BS}(2, 1) = \langle\ \mathtt{a}, \mathtt{b} : \mathtt{ab}^2 = \mathtt{ba}\ \rangle$ (see Figure 1, Example 20, and [51]).

Besides, such a Mealy automaton (right) can be used to compute the base 2 from the base 3 representation of the fractional part of any rational number, by iterating runs as explained above. For instance, finitely iterated runs from the state 0 and the initial word $0001\overline{1}$ produce the infinite word $(0\overline{1}0010)^\omega$, both words representing (the fractional part of) the rational $-\frac{2}{9}$ in base 3 (least significant digit first) and in base 2 (most significant digit first) respectively.

This innocuous example allows to illustrate the quite simple machineries associated both with automaton semigroups and with automatic semigroups. It also aims to give an informal glimpse of their behaviours through the duality principle: for instance, division *vs* multiplication, factor *vs* base, least *vs* most significant digit first, integer part *vs* fractional part.

3 Mealy automata and self-similar structures

We first recall the formal definition of an automaton. Possible references are [7, 14, 45, 47].

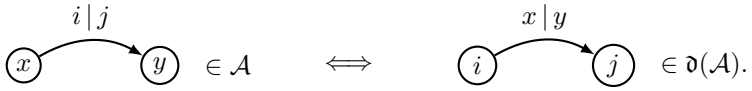
▶ **Definition 1.** *A* (finite, deterministic, and complete) automaton *is a triple* $\big(Q, \Sigma, \tau = (\tau_i \colon Q \to Q)_{i\in\Sigma}\big)$*, where the* stateset Q *and the* alphabet Σ *are non-empty finite sets, and where the* τ_i*'s are functions.*

A Mealy automaton *is a quadruple* $\big(Q, \Sigma, \tau = (\tau_i \colon Q \to Q)_{i\in\Sigma}, \sigma = (\sigma_x \colon \Sigma \to \Sigma)_{x\in Q}\big)$ *such that both* (Q, Σ, τ) *and* (Σ, Q, σ) *are automata.*

In other terms, a Mealy automaton is a complete, deterministic, letter-to-letter transducer with the same input and output alphabet.

The graphical representation of a Mealy automaton is standard, see Figures 2 and 7.

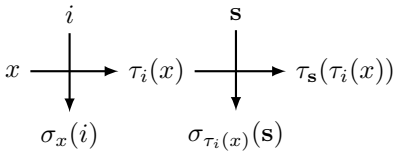
In a Mealy automaton $\mathcal{A} = (Q, \Sigma, \tau, \sigma)$, the sets Q and Σ play dual roles. So we may consider the *dual (Mealy) automaton* defined by $\mathfrak{d}(\mathcal{A}) = (\Sigma, Q, \sigma, \tau)$:



We view $\mathcal{A} = (Q, \Sigma, \tau, \sigma)$ as an automaton with an input and an output tape, thus defining mappings from input words over Σ to output words over Σ. Formally, for $x \in Q$, the map $\sigma_x \colon \Sigma^* \to \Sigma^*$, extending $\sigma_x \colon \Sigma \to \Sigma$, is defined recursively by:

$$\forall i \in \Sigma,\ \forall \mathbf{s} \in \Sigma^*, \qquad \sigma_x(i\mathbf{s}) = \sigma_x(i)\sigma_{\tau_i(x)}(\mathbf{s})\,.$$

The above equation can be easier to understood when depicted by a *cross-diagram* (see [1]):



By convention, the image of the empty word is itself. The mapping σ_x for each $x \in Q$ is length-preserving and prefix-preserving. We say that σ_x is the *production function* associated with $(\mathcal{A}, x)$. For $\mathbf{x} = x_1 \cdots x_n \in Q^n$ with $n > 0$, set $\sigma_{\mathbf{x}} \colon \Sigma^* \to \Sigma^*, \sigma_{\mathbf{x}} = \sigma_{x_n} \circ \cdots \circ \sigma_{x_1}$. Denote dually by $\tau_i \colon Q^* \to Q^*, i \in \Sigma$, the production functions associated with the dual automaton $\mathfrak{d}(\mathcal{A})$. For $\mathbf{s} = s_1 \cdots s_n \in \Sigma^n$ with $n > 0$, set $\tau_{\mathbf{s}} \colon Q^* \to Q^*,\ \tau_{\mathbf{s}} = \tau_{s_n} \circ \cdots \circ \tau_{s_1}$.

▶ **Definition 2.** *The semigroup of mappings from* Σ^* *to* Σ^* *generated by* $\{\sigma_x, x \in Q\}$ *is called the* semigroup generated by $\mathcal{A}$ *and is denoted by* $\langle\ \mathcal{A}\ \rangle_+$*. When* $\mathcal{A}$ *is invertible, its production functions are permutations on words of the same length and thus we may consider the corresponding group instead; this group is the* group generated by $\mathcal{A}$ *and is denoted by* $\langle\ \mathcal{A}\ \rangle$*. A (semi)group is called an* automaton (semi)group *whenever it can be generated by some Mealy automaton. The term* self-similar *is used as a synonym.*

4 Quadratic normalisations and automatic structures

This section gathers the definitions of some classical notions like normal form or automatic structure (see [24, 17, 33]), together with the slighly more specific notion of a quadratic normalisation (see [20, 22]).

For any set $\mathcal{Q}$, we denote by $\mathcal{Q}^+$ the free semigroup over $\mathcal{Q}$ (*resp.* by $\mathcal{Q}^*$ the free monoid and by $\mathtt{1}$ its unit element) and call its elements $\mathcal{Q}$-*words*. We write $|\mathbf{w}|$ for the length of a $\mathcal{Q}$-word $\mathbf{w}$, and $\mathbf{ww'}$ for the product of two $\mathcal{Q}$-words $\mathbf{w}$ and $\mathbf{w'}$.

▶ **Definition 3.** *Let S be a semigroup with a generating set $\mathcal{Q}$. A* normal form *for $(S, \mathcal{Q})$ is a (set-theoretic) section of the canonical projection* EV *from the language of $\mathcal{Q}$-words onto S, that is, a map* NF *that assigns to each element of S a distinguished representative $\mathcal{Q}$-word with* $\mathrm{EV} \circ \mathrm{NF} = \mathrm{Id}_S$*:*

$$\mathrm{EV} : \mathcal{Q}^+ \longrightarrow S \qquad \mathrm{NF}$$

Whenever $\mathrm{NF}(S)$ *is regular, it provides a* right-automatic structure *for S if the language* $\mathcal{L}_q = \{ \ (\ \mathrm{NF}(a)\#^{|\mathrm{NF}(aq)|},\ \mathrm{NF}(aq)\#^{|\mathrm{NF}(a)|}\ ) : a \in S \ \}$ *over the alphabet* $(\mathcal{Q} \sqcup \{\#\})^2$ *is regular for each $q \in \mathcal{Q}$, where the normal forms of a pair are right-padded with an extra symbol $\# \notin \mathcal{Q}$ to equalise the lengths. The semigroup S can then be called a* (right-)automatic semigroup*.*

We mention here the thorough and precious study in [32] of the different notions (right- or left-reading-padding *vs* right- or left-multiplication) of automaticity for semigroups.

▶ Remark 4. In his seminal work [24, Chapter 9], Thurston shows how the whole set of these different automata recognizing the multiplication – that is, recognizing the languages $\mathcal{L}_q$ – in Definition 3 can be replaced with advantage by a single letter-to-letter transducer over the alphabet $\mathcal{Q}$ (see Definition 14) that computes the normal forms via iterated runs: each run both provides one symbol of the final normal form and outputs a word still to be normalised.

One will often consider the associated normalisation $\mathrm{N} = \mathrm{NF} \circ \mathrm{EV}$ over $\mathcal{Q}$.

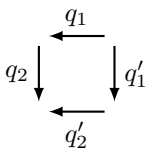
▶ **Definition 5.** *A* normalisation *is a pair $(\mathcal{Q}, \mathrm{N})$, where $\mathcal{Q}$ is a set and* N *is a map from $\mathcal{Q}^+$ to itself satisfying, for all $\mathcal{Q}$-words $\mathbf{u}, \mathbf{v}, \mathbf{w}$:*

- $|\mathrm{N}(\mathbf{w})| = |\mathbf{w}|$,
- $|\mathbf{w}| = 1 \Rightarrow \mathrm{N}(\mathbf{w}) = \mathbf{w}$,
- $\mathrm{N}(\mathbf{u}\,\mathrm{N}(\mathbf{w})\,\mathbf{v}) = \mathrm{N}(\mathbf{uwv})$.

A $\mathcal{Q}$-word $\mathbf{w}$ satisfying $\mathrm{N}(\mathbf{w}) = \mathbf{w}$ is called N-normal*. If S is a semigroup, we say that $(\mathcal{Q}, \mathrm{N})$ is a* normalisation for *S if S admits the presentation*

$$\langle\ \mathcal{Q} : \{\, \mathbf{w} = \mathrm{N}(\mathbf{w}) \mid \mathbf{w} \in \mathcal{Q}^+ \,\}\ \rangle_+.$$

We associate with every element $q \in \mathcal{Q}$ a q-labeled edge and with a product the concatenation of the associated edges, and represent equalities in the ambient semigroup using commutative diagrams, that we shall often organise as tilings and that we call here *square-diagram*. For instance, the following square illustrates an equality $q_1 q_2 = q_1' q_2'$.



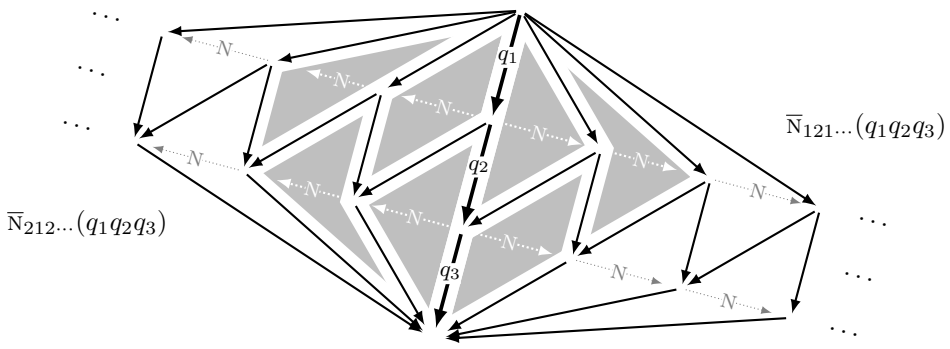


Figure 3 From an initial $\mathcal{Q}$-word $q_1q_2q_3$, one applies normalisations on the first and the second 2-factors alternatively up to stabilisation, beginning either on the first 2-factor q_1q_2 (right-hand side here) or on the second q_2q_3. The gray zone corresponds to Condition ( ) as defined in Definition 7.

For a normalisation $(\mathcal{Q}, \mathrm{N})$, we denote by $\overline{\mathrm{N}}$ the restriction of N to $\mathcal{Q}^2$ and, for $i \geq 1$, by $\overline{\mathrm{N}}_i$ the (partial) map from $\mathcal{Q}^+$ to itself that consists in applying $\overline{\mathrm{N}}$ to the entries in position i and $i+1$. For any finite sequence $\mathbf{i} = i_1 \cdots i_n$ of positive integers, we write $\overline{\mathrm{N}}_{\mathbf{i}}$ for the composite map $\overline{\mathrm{N}}_{i_n} \circ \cdots \circ \overline{\mathrm{N}}_{i_1}$ (so $\overline{\mathrm{N}}_{i_1}$ is applied first).

▶ **Definition 6.** *A normalisation* $(\mathcal{Q}, \mathrm{N})$ *is* quadratic *if the two following conditions hold:*

- *a* $\mathcal{Q}$*-word* $\mathbf{w}$ *is* N*-normal if, and only if, every length-two factor of* $\mathbf{w}$ *is;*
- *for every* $\mathcal{Q}$*-word* $\mathbf{w}$*, there exists a finite sequence* $\mathbf{i}$ *of positions, depending on* $\mathbf{w}$*, such that* $\mathrm{N}(\mathbf{w})$ *is equal to* $\overline{\mathrm{N}}_{\mathbf{i}}(\mathbf{w})$*.*

▶ **Definition 7.** *As illustrated in Figure 3, with any quadratic normalisation* $(\mathcal{Q}, \mathrm{N})$ *is associated its* breadth (d, p) *(called minimal left and right classes in [20, 22]) defined as:*

$$d = \max_{(q_1,q_2,q_3)\in\mathcal{Q}^3} \min\{\,\ell : \mathrm{N}(q_1q_2q_3) = \overline{\mathrm{N}}\underbrace{_{212\cdots}}_{\text{length } \ell}(q_1q_2q_3)\},\ \textit{and}$$

$$p = \max_{(q_1,q_2,q_3)\in\mathcal{Q}^3} \min\{\,\ell : \mathrm{N}(q_1q_2q_3) = \overline{\mathrm{N}}\underbrace{_{121\cdots}}_{\text{length } \ell}(q_1q_2q_3)\}.$$

Such a breadth need to be finite provided that $\mathcal{Q}$ *is finite, and then satisfies* $|d-p| \leq 1$*. For* $p \leq 3$ *(and* $d \leq 4$*), the quadratic normalisation* $(\mathcal{Q}, \mathrm{N})$ *is said to satisfy Condition* ( ) *(its corresponds with the so-called* domino rule *in [21] but with a different reading direction).*

The first main result of [22] is an axiomatisation of these quadratic normalisations satisfying Condition ( ) in terms of their restrictions to length-two words: any idempotent map $\overline{\mathrm{N}}$ on $\mathcal{Q}^2$ that satisfies $\overline{\mathrm{N}}_{2121} = \overline{\mathrm{N}}_{121} = \overline{\mathrm{N}}_{1212}$ extends into a quadratic normalisation $(\mathcal{Q}, \mathrm{N})$ satisfying Condition ( ). For larger breadths, a map on length-two words normalising length-three words needs not normalise words of greater length.

The second main result of [22] involves termination. Every quadratic normalisation $(\mathcal{Q}, \mathrm{N})$ gives rise to a quadratic rewriting system, namely the one with rules $\mathbf{w} \longrightarrow \overline{\mathrm{N}}(\mathbf{w})$ for $\mathbf{w} \in \mathcal{Q}^2$. By Definition 6, such a rewriting system is confluent and normalising, meaning that, for every initial word, there exists a finite sequence of rewriting steps leading to a unique N-normal word, but its convergence, meaning that *any* sequence of rewriting steps is finite, is a quite different problem.

▶ **Theorem 8** ([22])**.** *If* $(\mathcal{Q}, \mathrm{N})$ *is a quadratic normalisation satisfying Condition* ( )*, then the associated rewriting system is convergent.*

More precisely, every rewriting sequence starting from a word of $\mathcal{Q}^p$ has length at most $\frac{p(p-1)}{2}$ (*resp.* $2^p - p - 1$) in the case of a breadth $(3,3)$ (*resp.* either $(3,4)$ or $(4,3)$). Theorem 8 is essentially optimal since there exist nonconvergent rewriting systems with breadth $(4,4)$.

The results of Section 5 rely on the special Condition ( ). As mentioned, this condition was already outlined by Dehornoy and Guiraud (see [22]). However, none of their results (in particular Theorem 8 given above for the sake of completeness) is either applied or needed to establish ours. The current work and its exposition are thus self-contained and our constructions never require any of their stronger hypotheses (neither cancellativity nor absence of nontrivial invertible elements). We want here to emphasise that Condition ( ) happens to appear as a common denominator from different approaches (see also [29]).

5 From an automatic structure to a self-similar structure

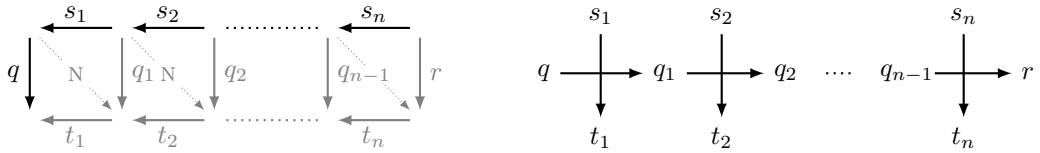
All the ingredients are now in place to effectively and naturally interpret as an automaton monoid any automatic monoid admitting a special language of normal forms – namely, a quadratic normalisation satisfying Condition ( ). The point is to construct a Mealy automaton encoding the behaviour of its language of normal forms under one-sided multiplication.

▶ **Definition 9.** *Assume that S is a semigroup admitting a quadratic normalisation $(\mathcal{Q}, \mathrm{N})$. We define the Mealy automaton $\mathcal{M}_{\mathcal{Q},\mathrm{N}} = (\mathcal{Q}, \mathcal{Q}, \tau, \sigma)$ such that, for every $(a,b) \in \mathcal{Q}^2$, $\sigma_b(a)$ is the rightmost element of $\mathcal{Q}$ in the normal form $\mathrm{N}(ab)$ of ab and $\tau_a(b)$ is the left one:*

$$\mathrm{N}(ab) = \tau_a(b)\sigma_b(a).$$

The latter correspondence can be simply interpreted via square-diagram *vs* cross-diagram:

For $\mathbf{s} = s_n \cdots s_1$, $\mathbf{t} = t_n \cdots t_1$, $\sigma_q(\mathbf{s}) = \mathbf{t}$, and $\tau_{\mathbf{s}}(q) = r$, we obtain diagrammatically:



We choose on purpose to always draw a normalisation square-diagram backward, such that it coincides with the associated cross-diagram. The function σ_q induced by the state q should map any word $\mathbf{s}$ (read backward) to some word $\mathbf{t}$ (read backward) with $\mathrm{N}(\mathbf{s}q) = \mathrm{N}(r\mathbf{t})$.

We now aim to strike reasonable (most often optimal) hypotheses for a quadratic normalisation $(\mathcal{Q}, \mathrm{N})$ associated with an original semigroup S to generate a semigroup $\langle\, \mathcal{M}_{\mathcal{Q},\mathrm{N}} \,\rangle_+$ that approximates S as sharply as possible. Since the generating sets coincide by Definition 9, we shall address the case when S is a quotient of $\langle\, \mathcal{M}_{\mathcal{Q},\mathrm{N}} \,\rangle_+$ (top-approximation, Lemma 10), and next, the case when $\langle\, \mathcal{M}_{\mathcal{Q},\mathrm{N}} \,\rangle_+$ is a quotient of S (bottom-approximation, Proposition 11).

Before establishing our top-approximation statement (Lemma 10), we could stress how semigroups might appear much more difficult to handle than monoids, especially when it comes to automaticity (see [32]) or self-similarity (see [11, 12]). For a semigroup S with a quadratic normalisation on $\mathcal{Q}$, two situations occur. First, if S is a monoid with unit $\mathtt{1}$, it admits a quadratic normalisation satisfying $\mathrm{N}(\mathtt{1}) = \mathtt{1}$ and

$$\mathrm{N}(\mathtt{1}q) = \mathrm{N}(q\mathtt{1}) = \mathtt{1}q \qquad (\boxed{1})$$

for each $q \in \mathcal{Q}$. Second, if S does not admit a unit, one can adjoin a unit $\mathtt{1}$ to obtain a monoid (if needed) with a quadratic normalisation satisfying Condition ($\boxed{1}$). The choice made for such a condition becomes natural whenever we think of the (adjoined or not) unit $\mathtt{1}$ as some *dummy* element that escapes from the normalisation and simply ensures its length-preserving property.

▶ **Lemma 10.** *If S is a monoid with a quadratic normalisation $(\mathcal{Q}, \mathrm{N})$ satisfying Condition ($\boxed{1}$), then the Mealy automaton $\mathcal{M}_{\mathcal{Q},\mathrm{N}}$ generates a monoid of which S is a quotient.*

Although specific to a monoidal framework and then requiring the innocuous Condition ($\boxed{1}$), the previous straightforward result relies only on the definition of a quadratic normalisation and on the well-fitted associated Mealy automaton (Definition 9). For the bottom-approximation statement, we consider an extra assumption, which happens to be necessary and sufficient.

▶ **Proposition 11.** *Assume that S is a semigroup with a quadratic normalisation $(\mathcal{Q}, \mathrm{N})$. If Condition ($\diamondsuit$) is satisfied, then the Mealy automaton $\mathcal{M}_{\mathcal{Q},\mathrm{N}}$ generates a semigroup quotient of S. The converse holds provided that Condition ($\boxed{1}$) is satisfied.*

Sketch proof. Let $S = \mathcal{Q}^+/\equiv_{\mathrm{N}}$ and $\mathcal{M}_{\mathcal{Q},\mathrm{N}} = (\mathcal{Q}, \mathcal{Q}, \tau, \sigma)$ as in Definition 9.

($\Leftarrow$) Assume that Condition ($\diamondsuit$) is satisfied and that there exists $(a, b, c, d) \in \mathcal{Q}^4$ with $ab \equiv_{\mathrm{N}} cd$. We have to prove $\sigma_{ab} = \sigma_{cd}$. Without loss of generality, the word ab can be supposed to be N-normal, that is, $\mathrm{N}(ab) = \mathrm{N}(cd) = ab$ holds.

Let $\mathbf{u} = q\mathbf{v} \in \mathcal{Q}^n$ for some $n > 0$ and $q \in \mathcal{Q}$. We shall prove both $\sigma_{ab}(\mathbf{u}) = \sigma_{cd}(\mathbf{u})$ (letterwise) and $\tau_{\mathbf{u}}(ab) \equiv_{\mathrm{N}} \tau_{\mathbf{u}}(cd)$ by induction on $n > 0$. For $n = 1$, we obtain the two square-diagrams on Figure 4 (left). With these notations, we have to prove $q_0'' = q_1''$ and $a'b' \equiv_{\mathrm{N}} c'd'$, the latter meaning $\mathrm{N}(a'b') = \mathrm{N}(c'd')$, that is, with the notations from Figure 4, the conjunction of $a'' = c''$ and $b'' = d''$. Now these three equalities hold whenever $(\mathcal{Q}, \mathrm{N})$ satisfies Condition ($\diamondsuit$), as shown on Figure 4 (right).

This allows to proceed the induction and to prove the implication ($\Leftarrow$).

($\Rightarrow$) Consider an arbitrary length 3 word over $\mathcal{Q}$, say qcd. Let a, b denote the elements in $\mathcal{Q}$ satisfying $\mathrm{N}(cd) = ab$. By definition, we deduce $ab \equiv_{\mathrm{N}} cd$. This implies $\sigma_{ab} = \sigma_{cd}$ by hypothesis. In particular, the images of any word $q\mathbf{v}$ under σ_{ab} and σ_{cd} coincide. Now $\sigma_{ab}(q\mathbf{v}) = \sigma_{cd}(q\mathbf{v})$ decomposes into

$$\sigma_{ab}(q) = q_0'' = q_1'' = \sigma_{cd}(q) \quad \text{and} \quad \sigma_{\tau_q(ab)}(\mathbf{v}) = \sigma_{a'b'}(\mathbf{v}) = \sigma_{c'd'}(\mathbf{v}) = \sigma_{\tau_q(cd)}(\mathbf{v})$$

(with notations of Figure 4). The last equality holds for any original word $\mathbf{v} \in \mathcal{Q}^*$ and implies $\sigma_{a'b'} = \sigma_{c'd'}$. Whenever, Condition ($\boxed{1}$) is satisfied, we deduce $\mathrm{N}(a'b') = \mathrm{N}(c'd')$ according to Lemma 10. For any such arbitrary word $qcd \in \mathcal{Q}^3$, we obtain $\overline{\mathrm{N}}_{121}(qcd) = \overline{\mathrm{N}}_{2121}(qcd)$. Therefore $(\mathcal{Q}, \mathrm{N})$ satisfies Condition ($\diamondsuit$). ◀

Gathering Lemma 10 and Proposition 11, we obtain the following main result.

▶ **Theorem 12.** *Assume that S is a monoid with a quadratic normalisation $(\mathcal{Q}, \mathrm{N})$ satisfying Conditions ($\boxed{1}$) and ($\diamondsuit$). The Mealy automaton $\mathcal{M}_{\mathcal{Q},\mathrm{N}}$ generates a monoid isomorphic to S.*

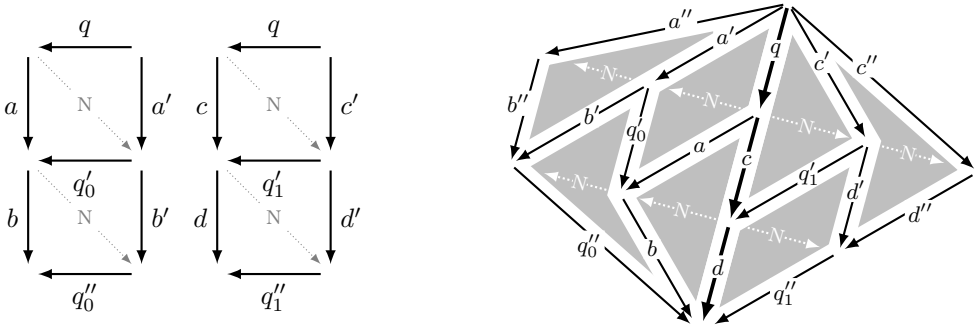


Figure 4 Proof of Proposition 11: initial data (left) can be pasted into Condition ( ) (right).

Proof. By construction, S and $\langle\, \mathcal{M}_{\mathcal{Q},\mathrm{N}} \,\rangle^1_+$ share a same generating subset $\mathcal{Q}$. Now, any defining relation for S maps to a defining relation for $\langle\, \mathcal{M}_{\mathcal{Q},\mathrm{N}} \,\rangle^1_+$ by Proposition 11, and conversely by Lemma 10. ◀

▶ **Corollary 13.** *Any monoid with a quadratic normalisation satisfying Conditions* ( ) *and* ( ) *is residually finite.*

To conclude this main section, we come back to that remark (following Definition 3) about the transducer approach by Thurston.

▶ **Definition 14.** *With any quadratic normalisation* $(\mathcal{Q}, \mathrm{N})$ *is associated its* Thurston transducer *defined as the Mealy automaton* $\mathcal{T}_{\mathcal{Q},\mathrm{N}}$ *with stateset* $\mathcal{Q}$*, alphabet* $\mathcal{Q}$*, and transitions as follows:*

▶ **Corollary 15.** *Assume that* S *is a monoid with a quadratic normalisation* $(\mathcal{Q}, \mathrm{N})$ *satisfying Conditions* ( ) *and* ( )*. The Thurston transducer* $\mathcal{T}_{\mathcal{Q},\mathrm{N}}$ *and the Mealy automaton* $\mathcal{M}_{\mathcal{Q},\mathrm{N}}$ *being dual automaton,* S *possesses both the explicitly dual properties of automaticity and self-similarity.*

These rather unexpected results provide the very first bridge between two fundamental areas that have always been widely seen as irreconcilable: automatic semigroups *vs* automaton semigroups. We choose to conclude by gathering several carefully selected examples, counterexamples, and open problems.

6 Examples and counterexamples

Our very first example is straightforward, but enlightening.

▶ **Example 16.** Every finite monoid $\mathcal{J}$ (in particular every finite group) is an *automaticon* monoid, that is, both an automatic and an automaton monoid. Consider its quadratic normalisation $(\mathcal{J}, \mathrm{N})$ with $\mathrm{N}(ab) = 1(ab)$ for every $(a, b) \in \mathcal{J}^2$. Figure 5 shows how to compute its breadth $(3, 2)$, witness of Condition ( ) for applying Theorem 12.

As mentioned in Section 1 and appearing on Figure 1, there exist automatic semigroups that cannot be automaton semigroups.

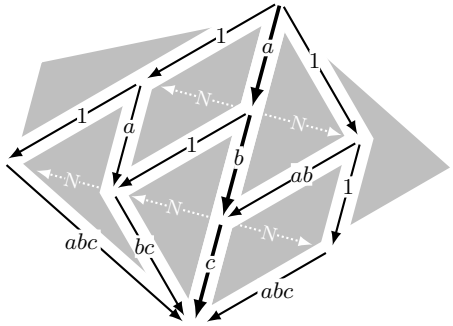


Figure 5 Computing the breadth $(3, 2)$ for any finite monoid $\mathcal{J}$ as in Example 16.

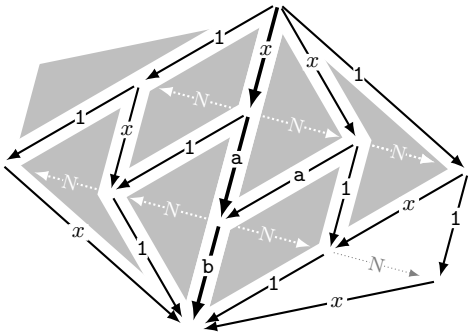


Figure 6 Computing the breadth $(3, 4)$ for the bicylic monoid $\mathbf{B}$ from Example 17.

▶ **Example 17.** The bicyclic monoid $\mathbf{B} = \langle\, \mathtt{a}, \mathtt{b} : \mathtt{ab} = \mathtt{1} \,\rangle^1_+$ is known to be automatic and not residually finite, hence cannot be an automaton monoid. Choose for $\mathbf{B}$ the quadratic normalisation $(\{\mathtt{a}, \mathtt{b}, \mathtt{1}\}, \mathrm{N})$ with $\mathrm{N}(\mathtt{ab}) = \mathtt{11}$, $\mathrm{N}(x\mathtt{1}) = \mathtt{1}x$ for $x \in \{\mathtt{a}, \mathtt{b}\}$, and $\mathrm{N}(xy) = xy$ otherwise. Figure 6 illustrates the computation (on the witness word $x\mathtt{ab}$ with $x \in \{\mathtt{a}, \mathtt{b}\}$) of its breadth $(3, 4)$. The Condition ( ) is hence not satisfied and Theorem 12 cannot apply. Precisely, according to the proof of Proposition 11 and Figure 4, we have $\sigma_{\mathtt{ab}}(x) = \mathtt{1} \neq x = \sigma_{\mathtt{11}}(x)$ for $x \in \{\mathtt{a}, \mathtt{b}\}$, hence $\sigma_{\mathtt{ab}} \neq \sigma_{\mathtt{1}} = \sigma_{\mathtt{11}}$.

By contrast, one of the simplest nontrivial examples could be the following.

▶ **Example 18.** The automatic monoid $\langle\, \mathtt{a}, \mathtt{b} : \mathtt{ab} = \mathtt{a} \,\rangle^1_+$ admits the quadratic normalisation $(\{\mathtt{a}, \mathtt{b}, \mathtt{1}\}, \mathrm{N})$ with $\mathrm{N}(\mathtt{ab}) = \mathtt{1a}$, $\mathrm{N}(x\mathtt{1}) = \mathtt{1}x$ for $x \in \{\mathtt{a}, \mathtt{b}\}$, and $\mathrm{N}(xy) = xy$ otherwise. Condition ( ) is satisfied, and the breadth is (3,3) according to the graph on Figure 7. By Theorem 12, it is therefore an automaton monoid, generated by the Mealy automaton displayed on Figure 7.

The latter happens to be the common smallest nontrivial member of the family of Baumslag–Solitar monoids (see [32] for instance), namely $\mathbf{BS}^1_+(1, 0)$, and of a wide family of right-cancellative semigroups, that we readily call *Artin–Krammer* monoids and that have been introduced and studied in [39] (see also [30, 31, 48]), namely $\mathbf{AK}^1_+(\Gamma)$ associated with the Coxeter-like matrix $\Gamma = \left[\begin{smallmatrix} 1 & 1 \\ 2 & 1 \end{smallmatrix}\right]$.

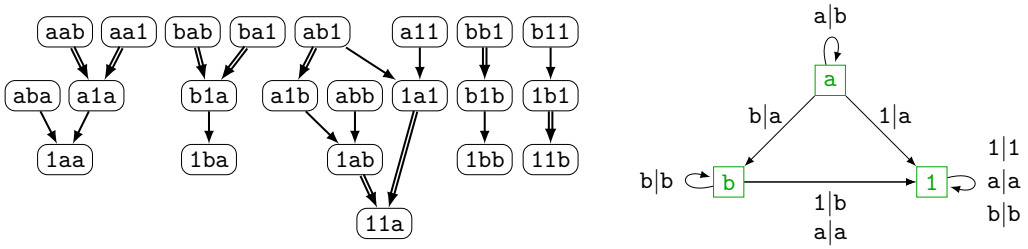


Figure 7 The $\overline{\mathrm{N}}$-graph for the quadratic normalisation associated with $\langle\, \mathtt{a}, \mathtt{b} : \mathtt{ab} = \mathtt{a} \,\rangle^1_+$ from Example 18: simple arrows correspond to $\overline{\mathrm{N}}_1$ and double arrows to $\overline{\mathrm{N}}_2$, while loops are simply omitted for better readability. The breadth is $(3, 3)$ as well. On the right is the associated Mealy automaton.

Examples 19 and 20 describe important members from both these families.

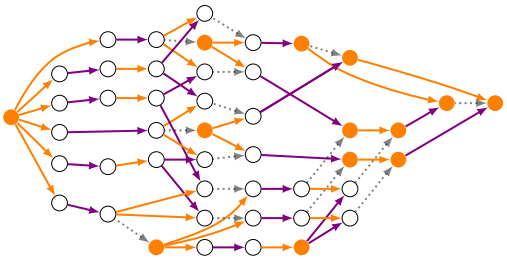


Figure 8 The minimal Garside family of the monoid $\mathbf{AK}^1_+\left(\begin{bmatrix}1&3&2\\4&1&3\\2&4&1\end{bmatrix}\right)$ from Example 19.

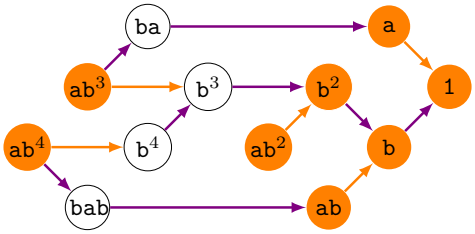


Figure 9 The minimal Garside family of the monoid $\mathbf{BS}^1_+(3,2)$ from Example 20.

▶ **Example 19.** The following Artin–Krammer monoid is emblematic:

$$\mathbf{AK}^1_+\left(\begin{bmatrix}1&3&2\\4&1&3\\2&4&1\end{bmatrix}\right) = \left\langle \mathtt{a},\mathtt{b},\mathtt{c} : \begin{matrix}\mathtt{abab}=\mathtt{aba}\\ \mathtt{ac}=\mathtt{ca}\\ \mathtt{bcbc}=\mathtt{bcb}\end{matrix} \right\rangle^1_+.$$

As displayed on Figure 8, its minimal so-called Garside family forms like a flint which encodes its whole combinatorics and, according to Theorem 12, makes it an automaticon monoid.

▶ **Example 20.** Consider the Baumslag–Solitar monoid $\mathbf{BS}^1_+(3,2) = \langle\, \mathtt{a},\mathtt{b} : \mathtt{ab}^3 = \mathtt{b}^2\mathtt{a} \,\rangle^1_+$. Displayed on Figure 9, its minimal Garside family contains eight elements (orange vertices) and makes it an automaticon monoid. This is an example of a group-embeddable automaton monoid whose enveloping group is not an automaton group. Indeed, the Baumslag–Solitar group $\mathbf{BS}(3,2)$ is precisely known as an example of non-residually finite group, hence cannot be an automaton group. The question remains open for those automaton semigroups whose enveloping group is a group of fractions, see Problem **C** below.

Concerning again group-embeddability, the following gives now an example of a cancellative automaton semigroup which is not group-embeddable.

▶ **Example 21.** The monoid $\mathbf{T} = \langle\, \mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{a}',\mathtt{b}',\mathtt{c}',\mathtt{d}' : \mathtt{ab}=\mathtt{cd}, \mathtt{a}'\mathtt{b}'=\mathtt{c}'\mathtt{d}', \mathtt{a}'\mathtt{d}=\mathtt{c}'\mathtt{b} \,\rangle^1_+$ is known (by Malcev work [41, 42, 43]) to be cancellative but not group-embeddable: from these three relations, we cannot deduce the relation $\mathtt{ad}' = \mathtt{cb}'$ that holds in the enveloping group. The quadratic normalisation $(\{\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{a}',\mathtt{b}',\mathtt{c}',\mathtt{d}'\}, \mathrm{N})$ defined by $\mathrm{N}(\mathtt{ab}) = \mathtt{cd}$, $\mathrm{N}(\mathtt{a}'\mathtt{b}') = \mathtt{c}'\mathtt{d}'$, and $\mathrm{N}(\mathtt{a}'\mathtt{d}) = \mathtt{c}'\mathtt{b}$ for instance has breadth $(3,3)$, hence satisfies Condition ( ) and Theorem 12 applies. This answers in particular a question by Cain [15].

Some classes of neither left- nor right-cancellative monoids have been studied and shown to admit nice normal forms yielding biautomatic structures (see [16]):

▶ **Example 22.** According to Schützenberger [54], plactic monoids are among the most fundamental monoids. The rank 2 plactic monoid is $\mathbf{P}_2 = \langle\, \mathtt{a},\mathtt{b} : \mathtt{aba} = \mathtt{baa}, \mathtt{bab} = \mathtt{bba} \,\rangle^1_+$. As noted in [20, 22], $\mathbf{P}_2$ admits the quadratic normalisation $(\mathcal{Q}, \mathrm{N})$ with $\mathcal{Q} = \{1, \mathtt{a}, \mathtt{b}, \mathtt{ba}\}$, $\mathrm{N}(\mathtt{ba}) = 1(\mathtt{ba})$, $\mathrm{N}((\mathtt{ba})\mathtt{a}) = \mathtt{a}(\mathtt{ba})$, $\mathrm{N}((\mathtt{ba})\mathtt{b}) = \mathtt{b}(\mathtt{ba})$, $\mathrm{N}(1x) = x1$ for $x \in \mathcal{Q}$, and $\mathrm{N}(xy) = xy$ otherwise. The latter has a breadth $(3,3)$, hence satisfies Condition ( ) and Theorem 12 ensures that $\mathbf{P}_2$ is an automaton monoid. Note that, for a higher rank plactic monoid $\mathbf{P}_X$, it suffices to take again for $\mathcal{Q}$ the set of *columns*, that is, the strictly decreasing products of elements of X. Chinese monoids admit quadratic normalisations with breadth $(4,3)$, hence satisfy Condition ( ), and Theorem 12 ensures that they are automaticon monoids [16, 51].

To conclude, we would like to illustrate the duality between "being an automatic semigroup" and "being an automaton semigroup" by highlighting a paradigmatic example.

▶ **Example 23.** The braid monoids were used by Thurston [24, Chapter 9] to describe his idea to build a single transducer that computes the so-called Adjan–Garside–Thurston normal form via iterated runs. The n-strand braid monoid is

$$\mathbf{B}^1_{n+} = \left\langle \sigma_1, \ldots, \sigma_{n-1} : \begin{array}{ll} \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j & \text{for } |i-j| \leq 1 \\ \sigma_i\sigma_j = \sigma_j\sigma_i & \text{for } |i-j| > 1 \end{array} \right\rangle^1_+ .$$

Garside theory allows to build a suitable generating set $\mathcal{Q}$ of size $n!$ and a corresponding quadratic normalisation with breadth $(3, 3)$. According to Corollary 15, its Thurston transducer and its Mealy automaton make therefore $\mathbf{B}^1_{n+}$ both an automatic and an automaton monoid. Such an approach may hopefully shed some light on the question of whether or not the braid groups are self-similar (Problem **B**). In particular, a positive answer to our following Problem **C** would imply a positive answer to Problem **B**.

Problem C. Is the group of fractions of an automaton monoid an automaton group?

References

1 Ali Akhavi, Ines Klimann, Sylvain Lombardy, Jean Mairesse, and Matthieu Picantin. On the finiteness problem for automaton (semi)groups. *Internat. J. Algebra Comput.*, 22(6):1–26, 2012.

2 Stanislas V. Alëšin. Finite automata and the Burnside problem for periodic groups. *Mat. Zametki*, 11:319–328, 1972.

3 Stanislav V. Alëšin. A free group of finite automata. *Vestnik Moskov. Univ. Ser. I Mat. Mekh.*, 4:12–14, 1983.

4 Algirdas Avižienis. Signed-digit number representations for fast parallel arithmetic. *IRE Trans. Electronic Computers*, 10(3):389–400, 1961.

5 Laurent Bartholdi. *FR – GAP package "Computations with functionally recursive groups", Version 2.4.5*, 2018. URL: `http://www.gap-system.org/Packages/fr.html`.

6 Laurent Bartholdi, Thibault Godin, Ines Klimann, and Matthieu Picantin. A New Hierarchy for Automaton Semigroups. In *23rd International Conference on Implementation and Applications of Automata (CIAA 2018)*, volume 10977 of *LNCS*, pages 71–83, 2018.

7 Laurent Bartholdi and Pedro V. Silva. Groups defined by automata. In J.-É. Pin, editor, *AutoMathA Handbook*. Europ. Math. Soc., 2010. (arXiv version: `arXiv:1012.1531`).

8 Laurent Bartholdi and Pedro V. Silva. Rational subsets of groups. In J.-É. Pin, editor, *AutoMathA Handbook*. Europ. Math. Soc., 2010. (arXiv version: `arXiv:1012.1532`).

9 Laurent Bartholdi and Zoran Šunić. Some solvable automaton groups. In *Topological and asymptotic aspects of group theory*, volume 394 of *Contemp. Math.*, pages 11–29. Amer. Math. Soc., Providence, RI, 2006.

10 Ievgen V. Bondarenko, Natalia V. Bondarenko, Saïd N. Sidki, and Flavia R. Zapata. On the conjugacy problem for finite-state automorphisms of regular rooted trees. *Groups Geom. Dyn.*, 7(2):323–355, 2013. With an appendix by Raphaël M. Jungers.

11 Tara Brough and Alan J. Cain. Automaton semigroup constructions. *Semigroup Forum*, 90(3):763–774, 2015.

12 Tara Brough and Alan J. Cain. Automaton semigroups: new constructions results and examples of non-automaton semigroups. *Theoret. Comput. Sci.*, 674:1–15, 2017.

13 Kai-Uwe Bux et al. Selfsimilar groups and conformal dynamics - Problem List. AIM workshop 2006. URL: `http://www.aimath.org/WWN/selfsimgroups/selfsimgroups.pdf`.

14 Alan J. Cain. Automaton semigroups. *Theoret. Comput. Sci.*, 410(47-49):5022–5038, 2009.

15 Alan J. Cain. Personal communication, 2016.

16 Alan J. Cain, Robert D. Gray, and António Malheiro. Rewriting systems and biautomatic structures for Chinese, hypoplactic, and Sylvester monoids. *Internat. J. Algebra Comput.*, 25(1-2):51–80, 2015.

17 Colin M. Campbell, Edmund F. Robertson, Nikola Ruškuc, and Richard M. Thomas. Automatic semigroups. *Theoret. Comput. Sci.*, 250(1-2):365–391, 2001.

18 Augustin-Louis Cauchy. *Sur les moyens d'éviter les erreurs dans les calculs numériques*, volume 5 of *Cambridge Library Collection - Mathematics*, pages 431–442. Cambridge University Press, 2009.

19 Daniele D'Angeli, Thibault Godin, Ines Klimann, Matthieu Picantin, and Emanuele Rodaro. Boundary action of automaton groups without singular points and Wang tilings. Submitted, 2016. `arXiv:1604.07736`.

20 Patrick Dehornoy. Garside and quadratic normalisation: a survey. In *19th International Conference on Developments in Language Theory (DLT 2015)*, volume 9168 of *LNCS*, pages 14–45, 2015.

21 Patrick Dehornoy et al. *Foundations of Garside theory.* Europ. Math. Soc. Tracts in Mathematics, volume 22, 2015. URL: `http://www.math.unicaen.fr/~garside/Garside.pdf`.

22 Patrick Dehornoy and Yves Guiraud. Quadratic normalization in monoids. *Internat. J. Algebra Comput.*, 26(5):935–972, 2016.

23 Murray Elder. *Automaticity, almost convexity and falsification by fellow traveler properties of some finitely presented groups.* PhD thesis, Univ Melbourne, 2000.

24 David B. A. Epstein, James W. Cannon, Derek F. Holt, Silvio V. F. Levy, Michael S. Paterson, and William P. Thurston. *Word processing in groups.* Jones and Bartlett Publishers, Boston, MA, 1992.

25 Pierre Gillibert. The finiteness problem for automaton semigroups is undecidable. *Internat. J. Algebra Comput.*, 24(1):1–9, 2014.

26 Thibault Godin, Ines Klimann, and Matthieu Picantin. On torsion-free semigroups generated by invertible reversible Mealy automata. In *9th International Conference on Language and Automata Theory and Applications (LATA 2015)*, pages 328–339, 2015.

27 Rostislav I. Grigorchuk. On Burnside's problem on periodic groups. *Funktsional. Anal. i Prilozhen.*, 14(1):53–54, 1980.

28 Rostislav I. Grigorchuk. Degrees of growth of finitely generated groups and the theory of invariant means. *Izv. Akad. Nauk SSSR Ser. Mat.*, 48(5):939–985, 1984.

29 Yves Guiraud and Matthieu Picantin. Resolutions by differential graded polygraphs. In preparation, 2019.

30 Alexander Hess. *Factorable monoids: resolutions and homology via discrete Morse theory.* PhD thesis, Univ Bonn, 2012. URL: `http://hss.ulb.uni-bonn.de/2012/2932/2932.pdf`.

31 Alexander Hess and Viktoriya Ozornova. Factorability, string rewriting and discrete Morse theory. Submitted. `arXiv:1412.3025`.

32 Michael Hoffmann. *Automatic Semigroups.* PhD thesis, Univ Leicester, 2001.

33 Michael Hoffmann and Richard M. Thomas. Biautomatic semigroups. In *15th International Symposium on Fundamentals of Computation Theory (FCT 2005)*, volume 3623 of *LNCS*, pages 56–67, 2005.

34 Ines Klimann. The finiteness of a group generated by a 2-letter invertible-reversible Mealy automaton is decidable. In *30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013)*, volume 20 of *LIPIcs*, pages 502–513, 2013.

35 Ines Klimann, Jean Mairesse, and Matthieu Picantin. Implementing Computations in Automaton (Semi)groups. In *17th International Conference on Implementation and Applications of Automata (CIAA 2012)*, volume 7381 of *LNCS*, pages 240–252, 2012.

36 Ines Klimann and Matthieu Picantin. Automaton (semi)groups: Wang tilings and Schreier tries. In Valérie Berthé and Michel Rigo, editors, *Sequences, Groups, and Number Theory.* Trends in Mathematics, 2018.

37 Ines Klimann, Matthieu Picantin, and Dmytro Savchuk. A Connected 3-State Reversible Mealy Automaton Cannot Generate an Infinite Burnside Group. In *19th International Conference on Developments in Language Theory (DLT 2015)*, volume 9168 of *LNCS*, pages 313–325, 2015.

38 Ines Klimann, Matthieu Picantin, and Dmytro Savchuk. Orbit automata as a new tool to attack the order problem in automaton groups. *J. Algebra*, 445:433–457, 2016.

39 Daan Krammer. An asymmetric generalisation of Artin monoids. *Groups Complex. Cryptol.*, 5:141–168, 2013.

40 Yaroslav Lavrenyuk, Volodymyr Mazorchuk, Andriy Oliynyk, and Vitaliy Sushchansky. Faithful group actions on rooted trees induced by actions of quotients. *Comm. Algebra*, 35(11):3759–3775, 2007.

41 Anatoly I. Malcev. On the immersion of an algebraic ring into a field. *Math. Ann.*, 113(1):686–691, 1937.

42 Anatoly I. Malcev. Über die Einbettung von assoziativen Systemen in Gruppen. *Rec. Math. [Mat. Sbornik] N.S.*, 6 (48):331–336, 1939.

43 Anatoly I. Malcev. Über die Einbettung von assoziativen Systemen in Gruppen. II. *Rec. Math. [Mat. Sbornik] N.S.*, 8 (50):251–264, 1940.

44 Victor D. Mazurov and Evgeny I. Khukhro. Unsolved problems in group theory. The Kourovka Notebook. No 19. URL: `https://kourovka-notebook.org/`.

45 David McCune. *Groups and Semigroups Generated by Automata*. PhD thesis, Univ Nebraska-Lincoln, 2011.

46 Yevgen Muntyan and Dmytro Savchuk. *AutomGrp – GAP package for computations in self-similar groups and semigroups, Version 1.3*, 2016. URL: `http://www.gap-system.org/Packages/automgrp.html`.

47 Volodymyr V. Nekrashevych. *Self-similar groups*, volume 117 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, RI, 2005.

48 Viktoriya Ozornova. *Factorability, discrete Morse theory, and a reformularion of $K(\pi, 1)$-conjecture*. PhD thesis, Univ Bonn, 2013. URL: `http://hss.ulb.uni-bonn.de/2013/3117/3117.pdf`.

49 Matthieu Picantin. Finite transducers for divisibility monoids. *Theoret. Comput. Sci.*, 362(1-3):207–221, 2006.

50 Matthieu Picantin. Tree products of cyclic groups and HNN extensions. Preprint, 2015. `arXiv:1306.5724v4`.

51 Matthieu Picantin. *Automates, (semi)groupes, dualités*. Habilitation à diriger des recherches, Univ Paris Diderot, 2017. URL: `https://www.irif.fr/~picantin/papers/hdr_memoire.pdf` and `https://www.irif.fr/~picantin/papers/hdr_soutenance.pdf`.

52 Matthieu Picantin. Automatic semigroups *vs* automaton semigroups. Full version of the current paper, 2019. `arXiv:1609.09364v5`.

53 Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, New York, NY, USA, 2009.

54 Marcel-Paul Schützenberger. Pour le monoïde plaxique. *Math. Inform. Sci. Humaines*, 140:5–10, 1997.

55 Pedro V. Silva. Groups and Automata: A Perfect Match. In *14th International Workshop on Descriptional Complexity of Formal Systems (DCFS 2012)*, volume 7386 of *LNCS*, pages 50–63, 2012.

56 Pedro V. Silva and Benjamin Steinberg. On a class of automata groups generalizing lamplighter groups. *Internat. J. Algebra Comput.*, 15(5-6):1213–1234, 2005.

57 Bartosz Tarnawski. Automatic groups as groups defined by transducers. Master's thesis, Univ Warsaw, Faculty of Mathematics, Informatics and Mechanics, Poland, 2017.

58 Daniel T. Wise. A non-Hopfian automatic group. *J. Algebra*, 180(3):845–847, 1996.

A Mahler's Theorem for Word Functions

Jean-Éric Pin[1]
IRIF, Université Paris Denis Diderot, CNRS - Case 7014 - F-75205 Paris Cedex 13, France
Jean-Eric.Pin@irif.fr

Christophe Reutenauer
Mathématiques, Université du Québec à Montréal, CP 8888, succ. Centre Ville, Canada H3C 3P8
reutenauer.christophe@uqam.ca

Abstract

Let p be a prime number and let $\mathcal{G}_p$ be the variety of all languages recognised by a finite p-group. We give a construction process of all $\mathcal{G}_p$-preserving functions from a free monoid to a free group. Our result follows from a new noncommutative generalization of Mahler's theorem on interpolation series, a celebrated result of p-adic analysis.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory; Theory of computation → Algebraic language theory

Keywords and phrases group languages, interpolation series, pro-p metric, regularity preserving

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.125

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Funding *Jean-Éric Pin*: DeLTA project (ANR-16-CE40-0007)
Christophe Reutenauer: Nserc Canada

1 Introduction

Throughout this paper, p denotes a prime number. A finite *p-group* is a group whose order is a power of p. Let $\mathcal{G}_p$ denote the variety of all languages recognised by a finite p-group. This variety, first studied over fourty years ago [2, p. 238] is generated by the p-binomial languages, as explained in Section 4.

A function f from A^* to B^* is *regularity-preserving* if, for each regular language L of B^*, the language $f^{-1}(L)$ is also regular. In a series of papers [8, 9, 10], Silva and the first author considered a more general situation: given a variety $\mathcal{V}$ of regular languages, a function f from A^* to B^* is *$\mathcal{V}$-preserving* if $L \in \mathcal{V}$ implies $f^{-1}(L) \in \mathcal{V}$. These functions admit a simple topological characterization. Indeed, one can attach to each variety $\mathcal{V}$ a metric[2] $d_{\mathcal{V}}$, called the *pro-$\mathcal{V}$ metric*, for which the following property holds: a function is $\mathcal{V}$-preserving if and only if it is uniformly continuous with respect to $d_{\mathcal{V}}$ [10, Theorem 4.1]. However, this characterization does not solve the following more difficult question:

Synthesis problem for $\mathcal{V}$. *Provide a construction process of all $\mathcal{V}$-preserving functions.*

For instance, although several families of regularity-preserving functions have been identified, the synthesis problem for these functions is still a major open problem.

The aim of this paper is to solve the synthesis problem for the variety $\mathcal{G}_p$. We actually solve this problem for all functions from A^* to the free group $F(B)$, a slightly more general setting, since the free monoid B^* embeds in $F(B)$. In a free group, the class $\mathcal{G}_p$ is defined in the same way: a subset of $F(B)$ is in $\mathcal{G}_p$ if it is recognized by a finite p-group.

[1] Corresponding author.
[2] Actually, it is only a pseudometric in the general case, but a metric in the case considered in this paper.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 125; pp. 125:1–125:13


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

One-letter case. If A and B are one-letter alphabets, then A^* is isomorphic to $\mathbb{N}$, $F(B)$ is isomorphic to $\mathbb{Z}$ and d_p is the *p-adic metric*. The *p-adic distance* between two distinct integers r and s is the real p^{-n}, where n is the exponent of p in the prime factorization of $|r-s|$. It turns out that *Mahler's theorem on interpolation series*, a celebrated result in p-adic analysis [4, 5] stated below, leads to a construction process of the $\mathcal{G}_p$-preserving functions from $\mathbb{N}$ to $\mathbb{Z}$.

Mahler's theorem is based on another result of independent interest. *Newton's forward difference formula* states that for each function $f\colon \mathbb{N} \to \mathbb{Z}$, there is a unique sequence of integers $\delta_k f$ such that, for all $n \in \mathbb{N}$, $f(n) = \sum_{k=0}^{\infty} \binom{n}{k} \delta_k f$. The value of these coefficients $\delta_k f$ is given by the formula $\delta_k f = (\Delta^k f)(0)$, where Δ^k is the k-th iteration of the *difference operator* Δ, defined by $(\Delta f)(n) = f(n+1) - f(n)$. A remarkable consequence of Newton's forward difference formula is that the map $f \to (\delta_k f)_{k \geqslant 0}$ defines a *bijection* between functions from $\mathbb{N}$ to $\mathbb{Z}$ and integer sequences. We call this bijection *Newton's bijection*.

Mahler's theorem states that the integer sequences that give rise to $\mathcal{G}_p$-preserving functions are precisely those converging to 0 in the p-adic metric. More precisely:

▶ **Theorem 1.1** (Mahler). *The following conditions are equivalent:*

(1) *$f\colon \mathbb{N} \to \mathbb{Z}$ is a $\mathcal{G}_p$-preserving function,*

(2) *f is uniformly continuous for the p-adic metric,*

(3) *the functions $\Delta^n f$ tend uniformly to the constant function $\mathbf{0}$ when n tends to ∞,*

(4) *the p-adic norm of $\delta_n f$ tends to 0 when n tends to ∞,*

(5) *f is the uniform limit of the polynomial functions $f_r(n) = \sum_{k=0}^{r} \binom{n}{k} \delta_k f$.*

This leads to a simple construction process of all $\mathcal{G}_p$-preserving functions from $\mathbb{N}$ to $\mathbb{Z}$: take a sequence $(\delta_k)_{k \geqslant 0}$ of integers converging to 0 and set $f(n) = \sum_{k=0}^{\infty} \binom{n}{k} \delta_k$.

Functions from A^* to $F(B)$. When B is a one-letter alphabet, a construction process of all $\mathcal{G}_p$-preserving functions was obtained by Silva and the first author in [10, 11]. We rely on this result to treat the general case where B is any finite alphabet.

We first equip both A^* and $F(B)$ with the pro-p metric, a natural extension of the p-adic metric, fully defined in Section 4.1. We follow [7] to extend the difference operators. Let f be a function from A^* to a group G. For each letter a, the *difference operator* Δ^a associates to f the function $\Delta^a f\colon A^* \to G$ defined by $\Delta^a f(u) = f(u)^{-1} f(ua)$. Next we attach a difference operator Δ^w to each word $w = a_1 \cdots a_n$ of A^* by setting $\Delta^w f = \Delta^{a_1}(\Delta^{a_2}(\cdots \Delta^{a_n} f) \cdots)$. Finally, we set $\delta_w f = \Delta^w f(1)$, where 1 is the empty word.

A noncommutative version of Newton's forward difference formula and of Newton's bijection was given by the first author in [7]. We give a simpler proof of these results in Section 2.5. In this noncommutative setting, Newton's bijection is now the map $f \to (\delta_w f)_{w \in A^*}$. If we just keep the elements $\delta_w f$ such that $|w| \leqslant n$ and replace every other $\delta_w f$ by the identity of the free group, the inverse of Newton's bijection gives back a function f_n, called the n-th *Newton polynomial function associated to f*.

Our main result now offers a noticeable analogy with Mahler's theorem:

▶ **Theorem 1.2.** *Let $f\colon A^* \to F(B)$ be a function. The following conditions are equivalent:*

(1) *f is a $\mathcal{G}_p$-preserving function,*

(2) *f is uniformly continuous for the pro-p metric,*

(3) *the functions $\Delta^w f$ tend uniformly to the constant function $\mathbf{1}$ when $|w|$ tends to ∞,*

(4) *the elements $\delta_w f$, where $w \in A^*$, tend to 1 when $|w|$ tends to ∞,*

(5) *f is the uniform limit of its Newton polynomial functions.*

Sequential products. A new operation on functions plays a key role in our proof of Theorem 1.2. Given an element g of a group G and a family $(f_a)_{a\in A}$ of functions from A^* to G, the *sequential product* of g and $(f_a)_{a\in A}$ is the function $f\colon A^* \to G$, defined, for each word $a_1\cdots a_n \in A^*$, by $f(a_1\cdots a_n) = g\prod_{1\leqslant i\leqslant n} f_{a_i}(a_1\cdots a_{i-1})$.

A function f from A^* to a group G is a *Newton polynomial function* if $\delta_w f = 1$ for almost all words w. We prove that the set of Newton polynomial functions from A^* to G is the smallest set of functions containing the constant functions and closed under sequential product. Moreover, if G is a finite p-group equipped with the discrete metric, then the Newton polynomial functions are exactly the uniformly continuous functions from A^* to G.

Two solutions of the synthesis problem. Theorem 1.2 now leads to two construction processes to obtain all $\mathcal{G}_p$-preserving functions from A^* to $F(B)$. The first one consists in taking any family $(\delta_w)_{w\in A^*}$ of elements of the free group converging to 1 when $|w|$ tends to ∞ and to use the inverse of Newton's bijection to get in return a $\mathcal{G}_p$-preserving function from A^* to $F(B)$. The second method is to start with the constant functions, use the sequential product to generate all Newton polynomial functions and finally take the uniform closure.

Related work. Another characterization of $\mathcal{G}_p$-preserving functions using profinite equations was obtained in [1, Lemma 3.3], but it only holds for *regular-preserving* functions. In the case of *sequential* and *rational* functions, $\mathcal{V}$-preserving functions were first investigated by Schützenberger and the second author [12]. For instance, they proved that a sequential function is $\mathcal{G}_p$-preserving if and only if the syntactic semigroup of its minimal sequential transducer is a finite p-group. Our results are of a different nature, since they concern all $\mathcal{G}_p$-preserving functions.

Organization. Difference operators and Newton's forward difference formula are introduced in Section 2. Section 3 is devoted to Newton polynomial functions and Section 4 to topological issues. The proof of our main result is presented in Section 5. Due to space constraints, missing proofs are given in the Appendix.

2 Difference operators and Newton's forward difference formula

Newton's forward difference formula gives an expression of a function from $\mathbb{N}$ to $\mathbb{Z}$ in terms of the initial value of the function and the powers of the forward difference operator. A noncommutative extension of this formula for functions from A^* to $F(B)$ was given in [7]. In this section, we give a new proof of these results. We first need to introduce a noncommutative version of the Magnus transformation.

2.1 Noncommutative Magnus transformation

Let A^{**} denote the free monoid freely generated by A^*. An element of A^{**} is a finite sequence $(w_1,\ldots,w_n)$ of elements of A^*. However, to avoid any confusion between the product in A^* and the product in A^{**}, we adopt an additive notation for A^{**}. This means that we replace the notation $(w_1,\ldots,w_n)$ by $w_1+\cdots+w_n$. The addition of two elements $(u_1+\cdots+u_m)$ and $(v_1+\cdots+v_n)$ of A^{**} is also denoted additively, which is coherent, since

$$(u_1+\cdots+u_m)+(v_1+\cdots+v_n) = u_1+\cdots+u_m+v_1+\cdots+v_n.$$

Accordingly, the neutral element of the monoid A^{**} is denoted 0. Note however that the addition is in general noncommutative. For each $w \in A^*$ and $x = x_1 + \cdots + x_n \in A^{**}$, let $x \cdot w = x_1 w + \cdots + x_n w$. This defines a *monoid right action* of A^* on A^{**}, which means that the following formulas hold for all $w, w_1, w_2 \in A^*$, and for all $x, x_1, x_2 \in A^{**}$,

$$0 \cdot w = 0 \qquad (x_1 + x_2) \cdot w = x_1 \cdot w + x_2 \cdot w \qquad x \cdot (w_1 w_2) = (x \cdot w_1) \cdot w_2.$$

The *noncommutative Magnus transformation* is the mapping μ from A^* into A^{**} defined recursively by setting $\mu(1) = 1$ and, for any $w \in A^*$ and $a \in A$,

$$\mu(wa) = \mu(w) + \mu(w) \cdot a. \tag{2.1}$$

For instance, $\mu(a) = 1 + a$, $\mu(ab) = 1 + a + b + ab$, $\mu(abc) = 1 + a + b + ab + c + ac + bc + abc$ and $\mu(abcd) = 1 + a + b + ab + c + ac + bc + abc + d + ad + bd + abd + cd + acd + bcd + abcd$.

2.2 Difference operators

Let G be a group and let $f\colon A^* \to G$ be a function. Following [7], we define the difference operators as follows. For each letter $a \in A$, $\Delta^a f$ is the function $A^* \to G$ defined by $\Delta^a f(w) = f(w)^{-1} f(wa)$ for any word w in A^*. We obtain in this way a function $a \mapsto \Delta^a$ from A into the set $\mathcal{M}$ of all mappings from G^{A^*} into itself. We view $\mathcal{M}$ as a monoid under the composition of mappings. Since A^* is the free monoid on A, this function from A to $\mathcal{M}$ extends uniquely to a monoid morphism from A^* into $\mathcal{M}$. Denoting $w \mapsto \Delta^w$ this extension, we get $\Delta^1 f = f$ and, for all words u, v in A^*,

$$\Delta^{uv} f = \Delta^u \Delta^v f. \tag{2.2}$$

For instance, one gets, for any $a, b, c \in A$ and $u \in A^*$,

$$(\Delta^1 f)(u) = f(u) \quad (\Delta^a f)(u) = f(u)^{-1} f(ua) \quad (\Delta^{ab} f)(u) = f(ub)^{-1} f(u) f(ua)^{-1} f(uab)$$
$$(\Delta^{abc} f)(u) = f(ubc)^{-1} f(ub) f(u)^{-1} f(uc) f(uac)^{-1} f(ua) f(uab)^{-1} f(uabc)$$

Here are two examples of differential operators. First, let us take $A^* = \mathbb{N}$ and $G = \mathbb{Z}$. Switching to additive notation, we find that $\Delta^1 f(n) = -f(n) + f(n+1)$, the usual difference operator, and more generally $\Delta^k f(n) = f(n+k) - \binom{n}{1} f(n+k-1) + \binom{n}{2} f(n+k-2) - \cdots + (-1)^k \binom{n}{k} f(n)$.

The next example requires an auxiliary definition. The *iterated commutator* $[x_1, x_2, \ldots, x_n]$ of n elements $x_1, x_2, \ldots, x_n$ of a group is defined by induction by setting $[x_1] = x_1$ and for $n \geqslant 2$, $[x_1, x_2, \ldots, x_n] = x_1 [x_2, x_3, \ldots, x_n] x_1^{-1} [x_2, x_3, \ldots, x_n]^{-1}$. In particular, since $[x_1, x_2] = x_1 x_2 x_1^{-1} x_2^{-1}$, one gets $[x_1, x_2, \ldots, x_n] = [x_1, [x_2, x_3, \ldots, x_n]]$.

▶ **Proposition 2.1.** *Let $f\colon A^* \to F(A)$ be the function defined by $f(x) = x^{-1}$. Then for every $n > 0$ and for all $a_1, \ldots, a_n \in A$, $\Delta^{a_1 a_2 \cdots a_n} f(x) = x [a_1, a_2, \ldots, a_n]^{-1} x^{-1}$.*

Difference operators commute with group morphisms:

▶ **Proposition 2.2.** *Let $f\colon A^* \to G$ be a function, let $\varphi\colon G \to H$ be a group morphism and let w be a word. Then $\Delta^w(\varphi \circ f) = \varphi \circ (\Delta^w f)$.*

2.3 The integration problem

Let G be a group and let $f\colon A^* \to G$ be a function. Then f and the functions $\Delta^a f$, for $a \in A$, are related by a functional equation.

▶ **Proposition 2.3.** *Let $a_1 \cdots a_n$ be a word of A^*. Then the following formula holds:*

$$f(a_1 \cdots a_n) = f(1) \prod_{1 \leqslant i \leqslant n} \Delta^{a_i} f(a_1 \cdots a_{i-1}). \tag{2.3}$$

The functional equation (2.3) gives an expression of f in terms of $f(1)$ and of the family $(\Delta^a f)_{a \in A}$. We now address the opposite question, which is somewhat similar to the problem of integrating a function from its derivative.

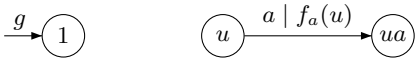
Integration problem. *Given an element g of G and a family $(f_a)_{a \in A}$ of functions from A^* to G, is there a function f such that $f(1) = g$ and $f_a = \Delta^a f$ for all $a \in A$?*

To solve the integration problem, it is convenient to introduce a new definition. Given an element g of G and a family $(f_a)_{a \in A}$ of functions from A^* to G, the *sequential product* $\mathrm{Seq}(g, (f_a)_{a \in A})$ is the function $f \colon A^* \to G$, defined, for each word $a_1 \cdots a_n \in A^*$, by

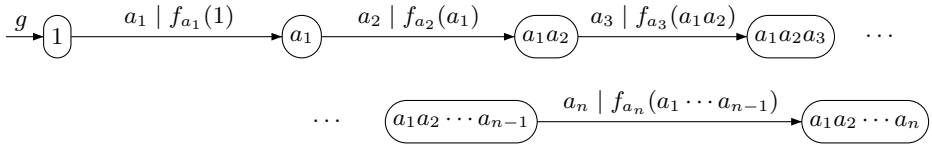
$$f(a_1 \cdots a_n) = g \prod_{1 \leqslant i \leqslant n} f_{a_i}(a_1 \cdots a_{i-1}). \tag{2.4}$$

By abuse of language, a function $f \colon A^* \to G$ is called *a sequential product of a family $(f_a)_{a \in A}$ of functions from A^* to G* if, for some $g \in G$, $f = \mathrm{Seq}(g, (f_a)_{a \in A})$.

This terminology stems from the fact that f can be realized by a sequential transducer with infinitely many states. Indeed, consider the sequential transducer $\mathcal{A} = (A^*, A, G, 1, \cdot, *, g)$, where A^* is the set of states, A the input alphabet, G the output group, 1 the initial state, g the initial prefix. The transition and the output functions are respectively defined by $u \cdot a = ua$ and $u * a = f_a(u)$.



A typical computation in $\mathcal{A}$ looks like this



and hence $\mathcal{A}$ computes the sequential product f defined by (2.4).

We are now ready to solve the integration problem.

▶ **Proposition 2.4.** *Let $g \in G$ and let $(f_a)_{a \in A}$ be a family of functions from A^* to G. Then the sequential product $\mathrm{Seq}(g, (f_a)_{a \in A})$ is the unique function f such that $f(1) = g$ and $\Delta^a f = f_a$ for all $a \in A$.*

Proof. Let $f = \mathrm{Seq}(g, (f_a)_{a \in A})$. Then $f(1) = g$ by definition. Let $u = a_1 \dots a_n$ be a word and a be a letter. Since $\Delta^a f(u) = f(u)^{-1} f(ua)$, one gets by (2.4)

$$\Delta^a f(u) = \Big(g \prod_{1 \leqslant i \leqslant n} f_{a_i}(a_1 \cdots a_{i-1})\Big)^{-1} g\Big(\prod_{1 \leqslant i \leqslant n} f_{a_i}(a_1 \cdots a_{i-1})\Big) f_a(a_1 \cdots a_n) = f_a(a_1 \cdots a_n)$$

whence $\Delta^a f = f_a$.

To prove uniqueness, consider a function f such that $f(1) = g$ and $\Delta^a f = f_a$ for all $a \in A$. Then for each word $a_1 \cdots a_n \in A^*$, one gets by (2.3),

$$f(a_1 \cdots a_n) = f(1) \prod_{1 \leqslant i \leqslant n} \Delta^{a_i} f(a_1 \cdots a_{i-1}) = g \prod_{1 \leqslant i \leqslant n} f_{a_i}(a_1 \cdots a_{i-1}).$$

and thus $f = \mathrm{Seq}(g, (f_a)_{a \in A})$. ◀

2.4 Newton's forward difference formula

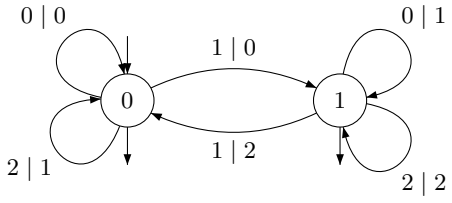
For each $w \in A^*$ and $f\colon A^* \to G$, let us set $\delta_w f = \Delta^w f(1)$ and let $\delta_f\colon A^* \to G$ be the map defined by $\delta_f(w) = \delta_w f$. This map extends to a monoid morphism $\delta_f^*\colon A^{**} \to G$. Thus $\delta_f^*(w) = \delta_w f$ and if $w_1 + \cdots + w_n$ is an element of A^{**}, then $\delta_f^*(w_1 + \cdots + w_n) = \delta_{w_1} f \cdots \delta_{w_n} f$.

▶ **Theorem 2.5.** *The equality $f = \delta_f^* \circ \mu$ holds for each function $f\colon A^* \to G$.*

The equality $f = \delta_f^* \circ \mu$ yields a noncommutative version of Newton's forward difference formula. Indeed, it extends the formula given in [11, Theorem 2.2] for functions from A^* to $\mathbb{Z}$, which itself extends Newton's forward difference formula for functions from $\mathbb{N}$ to $\mathbb{Z}$. To make this formula a little more concrete, let us compute a few values of $f(w)$. Let a, b, c, d be letters of A. Then, using the values of μ computed on page 4, one gets

$$\begin{aligned} f(1) &= \delta_1 f \qquad f(a) = (\delta_1 f)(\delta_a f) \qquad f(ab) = (\delta_1 f)(\delta_a f)(\delta_b f)(\delta_{ab} f) \\ f(abc) &= (\delta_1 f)(\delta_a f)(\delta_b f)(\delta_{ab} f)(\delta_c f)(\delta_{ac} f)(\delta_{bc} f)(\delta_{abc} f) \\ f(abcd) &= (\delta_1 f)(\delta_a f)(\delta_b f)(\delta_{ab} f)(\delta_c f)(\delta_{ac} f)(\delta_{bc} f)(\delta_{abc} f) \\ &\qquad (\delta_d f)(\delta_{ad} f)(\delta_{bd} f)(\delta_{abd} f)(\delta_{cd} f)(\delta_{acd} f)(\delta_{bcd} f)(\delta_{abcd} f). \end{aligned}$$

Here is a more complete example. Let $f\colon \{0,1,2\}^* \to \{0,1,2\}^*$ be the Euclidean division by 2 in base 3, that is, the function which associates to a word $u \in \{0,1,2\}^*$ representing an integer $\overline{u}$ in base 3, the unique word v of the same length as u representing the quotient of the division of $\overline{u}$ by 2. Since 1 is a letter of the alphabet, we let ϵ denote the empty word. The function f can be realized by the sequential transducer represented below.



For instance, $f(1212) = 0221$ since $\overline{1212} = 50$ and $\overline{0221} = 25 = 50/2$. Let us compute the functions $\Delta^x f$. First, we have

$$\Delta^0 f(w) = \begin{cases} 0 & \text{if } \overline{w} \text{ is even} \\ 1 & \text{if } \overline{w} \text{ is odd} \end{cases} \quad \Delta^1 f(w) = \begin{cases} 0 & \text{if } \overline{w} \text{ is even} \\ 2 & \text{if } \overline{w} \text{ is odd} \end{cases} \quad \Delta^2 f(w) = \begin{cases} 1 & \text{if } \overline{w} \text{ is even} \\ 2 & \text{if } \overline{w} \text{ is odd} \end{cases}$$

The other values of $\Delta^x f(w)$ can be obtained through the following result:

▶ **Proposition 2.6.** *Let $u, v \in A^*$ and let $g\colon \{0,1,2\}^* \to A^*$ be the function defined by*

$$g(w) = \begin{cases} u & \text{if } \overline{w} \text{ is even} \\ v & \text{if } \overline{w} \text{ is odd} \end{cases}$$

Then $\Delta^x g(w) = \varepsilon$ if $x \notin 1^*$ and $\Delta^{1^n} g(w) = (u^{-1}v)^{(-1)^{n-1+\overline{w}} 2^{n-1}}$ for $n \geqslant 1$.

It is now easy to compute the elements $\delta_w = (\Delta^w f)(\epsilon)$. One gets $\delta_0 = 0$, $\delta_1 = 0$, $\delta_2 = 1$, $\delta_{1^n 0} = (0^{-1}1)^{(-1)^{n-1}2^{n-1}}$, $\delta_{1^n 1} = (0^{-1}2)^{(-1)^{n-1}2^{n-1}}$, $\delta_{1^n 2} = (1^{-1}2)^{(-1)^{n-1}2^{n-1}}$ and $\delta_w = \epsilon$ in all other cases. We get for instance

$$\begin{aligned} f(1212) &= \delta_\epsilon \delta_1 \delta_2 \delta_{12} \delta_1 \delta_{11} \delta_{21} \delta_{121} \delta_2 \delta_{12} \delta_{22} \delta_{122} \delta_{12} \delta_{112} \delta_{212} \delta_{1212} \\ &= \delta_1 \delta_2 \delta_{12} \delta_1 \delta_{11} \delta_2 \delta_{12} \delta_{12} \delta_{112} = 01(1^{-1}2)0(0^{-1}2)1(1^{-1}2)(1^{-1}2)(1^{-1}2)^{-2} = 0221. \end{aligned}$$

2.5 Newton's bijection

For each $n \in \mathbb{N}$, let C_n be the set of words of A^* of length at most n. Let ρ_n be the monoid endomorphism on A^{**} which maps every element of C_n to itself, and maps any other element of A^* to 0. In other words, if $x = \sum_{1 \leqslant i \leqslant r} u_i$ is an element of A^{**}, where each $u_i \in A^*$, then

$$\rho_n(x) = \sum_{i \in E_n(x)} u_i, \text{ where } E_n(x) = \{i \in \{1, \dots r\} \mid |u_i| \leqslant n\}.$$

For each $n \geqslant 0$, C_n is a finite subset of A^{**} and C_n^* is a free submonoid of A^{**}. The function $\mu_n = \rho_n \circ \mu$ from A^* to the free monoid C_n^* is called the *truncated noncommutative Magnus transformation*. For instance, $\mu_2(abcd) = 1 + a + b + ab + c + ac + bc + d + ad + bd + cd$, a result obtained by only keeping the words of length $\leqslant 2$ in $\mu(abcd)$.

Recall that to each function $f\colon A^* \to G$ is associated the map $\delta_f\colon A^* \to G$ defined by $\delta_f(w) = \delta_w(f)$. The *Newton map* is the map $\delta\colon f \to \delta_f$. Let $f^*\colon A^{**} \to G$ denote the unique monoid morphism extending f and let γ be the map defined by $\gamma(f) = f^* \circ \mu$.

▶ **Theorem 2.7.** *The Newton map δ is a permutation on the set of functions from A^* to G and its inverse permutation is γ.*

Proof. Since $f = \delta_f^* \circ \mu$ by Theorem 2.5, $\gamma \circ \delta$ is the identity function. Therefore γ is surjective, δ is injective and it suffices to prove that γ is injective. Let $g, h\colon A^* \to G$ be such that $g^* \circ \mu = h^* \circ \mu$. Let us show by induction on $|w|$ that $g(w) = h(w)$. If $|w| = 0$, then w is the empty word 1 , $\mu(1) = 1$, $g^*(1) = g(1)$, $h^*(1) = h(1)$ and thus $g(1) = h(1)$. Suppose now that $|w| = n + 1$. Then $\mu(w) = \mu_n(w) + w$ and since g^* and h^* are monoid morphisms, one gets $g^* \circ \mu(w) = g^*(\mu_n(w) + w) = g^*(\mu_n(w))g(w)$ and similarly $h^* \circ \mu(w) = h^*(\mu_n(w))h(w)$. Since $\mu_n(w)$ is a sum of words of length $\leqslant n$, the induction hypothesis gives $g^*(\mu_n(w)) = h^*(\mu_n(w))$. Now since $g^* \circ \mu(w) = h^* \circ \mu(w)$, one gets $g(w) = h(w)$, which concludes the induction step. ◀

Theorem 2.7 solves the following interpolation problem.

▶ **Corollary 2.8.** *For each function $g : A^* \to G$, there exists a unique function $f\colon A^* \to G$ such that, for all $u \in A^*$, $\delta_u f = g(u)$.*

3 Newton polynomial functions

Let G be a group. A function $f\colon A^* \to G$ is called a *Newton polynomial function* if $\delta_w f = 1$ for almost all words $w \in A^*$. Note that by Proposition 2.6, the Euclidean division by 2 in base 3 is not a Newton polynomial function.

Let $\mathbf{1}$ denote the constant function from A^* to G that maps every word to 1. The *degree* of a Newton polynomial function is -1 if $f = \mathbf{1}$; otherwise, it is the smallest d such that $\delta_w f = 1$ for any word of length $d + 1$.

Here is another convenient characterization of Newton polynomial functions.

▶ **Proposition 3.1.** *A function $f\colon A^* \to G$ is a Newton polynomial function of degree d if and only if d is the smallest integer such that $\Delta^w f = \mathbf{1}$ for all words w of length $d + 1$.*

The following result gives a construction process of the set of Newton polynomial functions.

▶ **Theorem 3.2.** *The set of Newton polynomial functions from A^* to G is the smallest set of functions from A^* to G containing the constant functions and closed under sequential product.*

Theorem 3.2 is an immediate consequence of the following proposition.

▶ **Proposition 3.3.** *Let G be a group and let $f\colon A^* \to G$ be a function. The following conditions are equivalent:*

(1) *f is a Newton polynomial function of degree $\leqslant d$,*

(2) *there exists a family $(f_a)_{a\in A}$ of Newton polynomial functions of degree $\leqslant d-1$ such that $f = \mathrm{Seq}(f(1), (f_a)_{a\in A})$.*

In this case, one has $f_a = \Delta^a f$ for every $a \in A$.

Proof. (1) implies (2). Suppose that f is a Newton polynomial function of degree $\leqslant d$. Then for any letter a, $\Delta^a f$ is a Newton polynomial function of degree at most $d-1$. Moreover, Proposition 2.3 shows that $f(a_1\cdots a_n) = f(1)\prod_{1\leqslant i\leqslant n}\Delta^{a_i} f(a_1\cdots a_{i-1})$, which proves (2).

(2) implies (1). Suppose that (2) holds. Proposition 2.4 shows that, for each letter a, $\Delta^a f = f_a$ and hence $\Delta^a f$ is a Newton polynomial function of degree $\leqslant d-1$. It follows that f is a Newton polynomial function of degree $\leqslant d$. ◀

A Newton polynomial function of degree 0 is a constant map different from $\mathbf{1}$. A Newton polynomial function of degree 1 is an *affine morphism*, that is, a function f of the form $f(w) = f(1)g(w)$ for some monoid morphism $g\colon A^* \to G$. Equivalently, conjugating by $f(1)$, one gets $f(w) = h(w)f(1)$ for some monoid morphism $h : A^* \to G$.

The function $f\colon A^* \to F(A)$ defined by $f(a_1\cdots a_n) = a_1(a_1a_2)(a_1a_2a_3)\ \cdots\ (a_1\cdots a_n)$ is a Newton polynomial function of degree 2. Indeed, it is equal to the sequential product $\mathrm{Seq}(1, (f_a)_{a\in A})$ where each f_a is the affine morphism defined by $f_a(u) = ua$.

Recall that δ_f^* is a map from A^{**} to G, but we keep the same notation for its restriction to C_n^*. Let $f\colon A^* \to G$ be a function. For each $n \geqslant 0$, the *n-th Newton polynomial function* associated to f is the function f_n from A^* to G defined by $f_n = \delta_f^* \circ \mu_n$. This terminology is justified by Proposition 3.4 below.

It is not difficult to see that f_0 is the constant function equal to $f(1)$. Indeed, since $\Delta^1 f = f$, one gets $f_0(u) = \delta_f^* \circ \mu_0(u) = \delta_f^*(1) = \delta_1 f = \Delta^1 f(1) = f(1)$.

▶ **Proposition 3.4.** *For each $n \geqslant 0$, f_n is a Newton polynomial function of degree at most n.*

We need an auxiliary lemma.

▶ **Lemma 3.5.** *The following formula holds for all $n > 0$ and $a \in A$.*

$$\Delta^a(f_n) = \delta^*_{\Delta^a f} \circ \mu_{n-1} = (\Delta^a f)_{n-1}$$

Proof of Proposition 3.4. We prove the result by induction on n. For $n = 0$, we have already seen that f_0 is a constant function, and thus a Newton polynomial function of degree $\leqslant 0$. Applying Proposition 2.3 to f_n, one gets, for every word $a_1\cdots a_k \in A^*$, $f_n(a_1\cdots a_k) = f_n(1)\prod_{1\leqslant i\leqslant k}\Delta^{a_i} f_n(a_1\ldots a_{i-1})$. Now, $f_n(1) = \delta_f^* \circ \mu_n(1) = \delta_f^*(1) = f(1)$ and $\Delta^a f_n = (\Delta^a f)_{n-1}$ by Lemma 3.5. It follows that $f_n(a_1\ldots a_k) = f(1)\prod_{1\leqslant i\leqslant k}(\Delta^{a_i} f)_{n-1}(a_1\ldots a_{i-1})$. By the induction hypothesis applied to $\Delta^a f$, $(\Delta^a f)_{n-1}$ is a Newton polynomial function of degree at most $n-1$. Hence by Proposition 3.3, f_n is a Newton polynomial function of degree at most n. ◀

A function $f\colon A^* \to G$ is called a *G-polynomial* if $f(w) = 1$ for almost all words $w \in A^*$. The *degree* of a G-polynomial is -1 if $f = \mathbf{1}$; otherwise, it is the smallest d such that $f(w) = 1$ for any word of length $d+1$. One can now enrich Theorem 2.7 as follows.

▶ **Theorem 3.6.** *For each degree d, the maps δ and γ define mutually inverse bijections between the set of Newton polynomial functions of degree d and the set of G-polynomials of degree d.*

Proof. It suffices to prove that δ and γ define mutually inverse bijections between the set of Newton polynomial functions of degree $\leqslant d$ and the set of G-polynomials of degree $\leqslant d$. Let f be a Newton polynomial function of degree $\leqslant d$. Then by definition, $\delta(f)$ is a G-polynomial of degree $\leqslant d$. Let now f be a G-polynomial of degree $\leqslant d$. Theorem 2.7 shows that $f = \delta \circ \gamma(f) = \delta_{\gamma(f)}$. It follows that for every word w of length $> d$, $1 = f(w) = \delta_{\gamma(f)}(w)$. Thus $\gamma(f)$ is a Newton polynomial of degree $\leqslant d$. ◀

4 Topology

4.1 Pro-p metrics

If $\bar{B}$ is a copy of B, the free group $F(B)$ is the quotient of $(B \cup \bar{B})^*$ under the congruence generated by the relations $b\bar{b} = 1 = \bar{b}b$ for all $b \in B$.

Recall that a group G is called *residually p-finite* if for any $g \neq 1$ in G, there is some finite p-group H and some morphism $G \to H$ whose kernel does not contain g. It is a well-known fact that free groups are residually p-finite.

Let G be a residually p-finite group and let $g \in G$. The *pro-p valuation* of g, denoted $v_p(g)$, is the largest n such that g belongs to the kernel of any morphism from G to a p-group of order p^n. The pro-p valuation is always finite, except for $g = 1$, in which case it is infinite. The *pro-p norm* of g is $|g|_p = p^{-v_p(g)}$, with the usual convention $p^{-\infty} = 0$. Finally G becomes a metric space for the *pro-p metric* $d_p \colon G \times G \to \mathbb{R}$ defined by $d_p(x, y) = |x^{-1}y|_p$.

The condition $d_p(x, y) \leqslant p^{-k}$ means that $x^{-1}y$ is in the kernel of each group morphism from G into a p-group of cardinality at most p^k. We leave to the reader to verify that if $G = \mathbb{Z}$, one recovers the usual p-adic valuation, norm and metric.

Another useful example occurs when G is a finite p-group. Recall that the *discrete metric* on G is the metric d defined by $d(x, y) = 1$ if $x \neq y$ and $d(x, y) = 0$ if $x = y$. In this case, the double inequality $d_p(x, y) \leqslant d(x, y) \leqslant |G|\, d_p(x, y)$ shows that the pro-p metric is uniformly equivalent to the discrete metric.

There are two equivalent ways to define the pro-p metric on a free monoid A^*. The first solution is to view A^* as a subspace of the free group $F(A)$ and to consider the restriction to A^* of the pro-p metric on $F(A)$.

The second solution is to directly define the pro-p metric as follows. Let us say that a finite p-group G *separates* two words u and v of A^* if there exists a monoid morphism $\varphi \colon A^* \to G$ such that $\varphi(u) \neq \varphi(v)$. Then $d_p(u, v) = 0$ if $u = v$ and $d_p(u, v) = p^{-n}$, where p^n is the minimal size of a p-group separating u and v, if $u \neq v$.

▶ **Proposition 4.1.** *Every monoid morphism from A^* to a p-group is uniformly continuous.*

Proof. Let π be a monoid morphism from A^* to a p-group G and let $u, v \in A^*$. If $d_p(u, v) \leqslant |G|^{-1}$, then $\pi(u) = \pi(v)$ and thus $d_p(\pi(u), \pi(v)) = 0$. Thus π is uniformly continuous. ◀

Let us now review the connections with combinatorics on words and regular languages. A word $u = a_1 a_2 \cdots a_n$ (where $a_1, \ldots, a_n$ are letters) is a *subword* of a word v if v can be written as $v = v_0 a_1 v_1 \cdots a_n v_n$. For instance, ab is a subword of $cacbc$.

Following Eilenberg [2] and Lothaire [3, Chapter 6], let $\binom{v}{u}$ denote the number of distinct ways to write a word u as a subword of v. More formally, if $u = a_1 a_2 \cdots a_n$, then

$$\binom{v}{u} = \mathrm{Card}\{(v_0, v_1, \ldots, v_n) \mid v_0 a_1 v_1 \cdots a_n v_n = v\}$$

A language of A^* is *p-binomial* if for some word v and some integer r it is equal to

$$L(v,r) = \{w \in A^* \mid \binom{w}{v} \equiv r \bmod p\}.$$

It follows from [2, p. 238] that a language belongs to $\mathcal{G}_p$ if and only if it is a Boolean combination of p-binomial languages. We will also use the following consequence of [11, Proposition 1.3 and Theorem 1.4].

▶ **Proposition 4.2.** *Let $f\colon A^* \to B^*$ be a function. The following conditions are equivalent:*
(1) *f is uniformly continuous for the pro-p metric,*
(2) *f is $\mathcal{G}_p$-preserving,*
(3) *for each p-binomial language L of B^*, $f^{-1}(L)$ is a Boolean combination of p-binomial languages in A^*.*

4.2 Uniform continuity and Newton polynomial functions

The aim of this section is to describe the uniformly continuous functions from A^* to a finite p-group. We first give a purely algebraic characterization of these functions (Proposition 4.3). Then we show that these functions are closed under applying differential operators (Proposition 4.4) and under taking sequential products (Proposition 4.5).

▶ **Proposition 4.3.** *Let G be a finite p-group and let $f\colon A^* \to G$ be a function. Then f is uniformly continuous for the pro-p metric if and only if there exist a finite p-group K and a monoid morphism $\zeta\colon A^* \to K$ such that f factors through ζ, that is, there is a map $\lambda\colon K \to G$ such that $f = \lambda \circ \zeta$.*

▶ **Proposition 4.4.** *Let G be a finite p-group. If $f\colon A^* \to G$ is uniformly continuous for the pro-p metric, then so is $\Delta^w f$ for any word $w \in A^*$.*

Proof. By induction and by Equation (2.2), it is enough to prove the result for $w = a$ for any letter $a \in A$. In this case, $\Delta^a f\colon A^* \to G$ is the composition of the following functions:

$$\begin{array}{lll} A^* \to A^* \times A^* & A^* \times A^* \to A^* \times A^* & A^* \times A^* \to G \times G \\ \quad u \mapsto (u,u) & \quad (u,v) \mapsto (u,va) & \quad (u,v) \mapsto (f(u),f(v)) \end{array}$$

$$\begin{array}{ll} G \times G \to G \times G & G \times G \to G \\ \quad (g,h) \mapsto (g^{-1},h) & \quad (g,h) \mapsto gh \end{array}$$

as shown by the sequence

$$u \mapsto (u,u) \mapsto (u,ua) \mapsto (f(u),f(ua)) \mapsto ((f(u))^{-1},f(ua)) \mapsto (f(u))^{-1}f(ua) = \Delta^a f(u).$$

Since the pro-p metric is compatible with the monoid structure (see [6, Section 2] or [11, Section 1.4]), each of these functions is uniformly continuous and so is their composition. ◀

▶ **Proposition 4.5.** *Let G be a residually p-finite group. Any sequential product of uniformly continuous functions from A^* to G is uniformly continuous for the pro-p metric.*

Here is an important consequence of these results.

▶ **Proposition 4.6.** *Let G be a finite p-group. Every Newton polynomial function $f\colon A^* \to G$ is uniformly continuous.*

Proof. We prove the result by induction on the degree d of f. If $d \leqslant 0$, then f is a constant function and hence f is uniformly continuous. Otherwise, Proposition 3.3 shows that f is a sequential product of a family $(f_a)_{a\in A}$ of Newton polynomial functions of degree $\leqslant d-1$. By the induction hypothesis, each f_a is uniformly continuous and hence f is uniformly continuous by Proposition 4.5. ◀

We now establish the converse of Proposition 4.6.

▶ **Proposition 4.7.** *Let G be a finite p-group. If a function $f\colon A^* \to G$ is uniformly continuous for the pro-p metric, then f is a Newton polynomial function.*

Several auxiliary definitions are needed to prove this proposition.

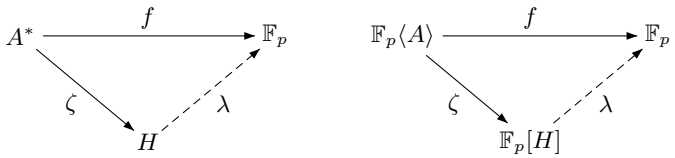
Let $\mathbb{F}_p$ be the field with p elements and let $\mathbb{F}_p[G]$ be the *group algebra* of G over $\mathbb{F}_p$. Each group morphism $G_1 \to G_2$ extends uniquely, by linearity, to an $\mathbb{F}_p$-algebra morphism from $\mathbb{F}_p[G_1]$ to $\mathbb{F}_p[G_2]$. Similarly, each function from G to $\mathbb{F}_p$ extends uniquely, by linearity, to a linear form on $\mathbb{F}_p[G]$.

The vector space of linear forms on a $\mathbb{F}_p$-algebra R (that is, the *dual* of R) is a left R-module: the action is defined, for any elements x, y in R and any linear form f on R by $(x \cdot f)(y) = f(yx)$.

Sketch of the proof of Proposition 4.7. Let p^r be the order of G. We prove the result by induction on r.

For $r = 1$, $G = \mathbb{Z}/p\mathbb{Z}$ and we switch to additive notation. Thus we have to show that $\Delta^w f = \mathbf{0}$ for almost all w. Since $\mathbb{Z}/p\mathbb{Z}$ is the additive group of the field $\mathbb{F}_p$, we may consider f as a function $A^* \to \mathbb{F}_p$. Since f is uniformly continuous, there exist by Proposition 4.3 a finite p-group H, a monoid morphism $\zeta\colon A^* \to H$ and a function $\lambda\colon H \to \mathbb{F}_p$ such that $f = \lambda \circ \zeta$, see the diagram on the left hand side of the figure below.

We extend by linearity all these functions, as explained previously, and denote these extensions by the same letters. We obtain the diagram on the right hand side of the figure below. Now ζ is a morphism of $\mathbb{F}_p$-algebra and f, as well as λ, are $\mathbb{F}_p$-linear forms.



With these notations, one can first show that

$$\Delta^w f = \Delta^{a_1\cdots a_n} f = \big((a_1 - 1)\cdots(a_n - 1)\big)\cdot f|_{A^*}. \tag{4.1}$$

Since $f = \lambda \circ \zeta$ and $\zeta\big((a_1 - 1)\cdots(a_n - 1)\big) = \big(\zeta(a_1) - 1\big)\cdots\big(\zeta(a_n) - 1\big)$, a little bit of work shows that

$$\big((a_1 - 1)\cdots(a_n - 1)\big)\cdot f = \Big(\big((\zeta(a_1) - 1)\cdots(\zeta(a_n) - 1)\big)\cdot\lambda\Big)\circ\zeta \tag{4.2}$$

Let $I_H = \Big\{ \sum_{g\in H} a_g g \mid \sum_{g\in H} a_g = 0 \Big\}$ be the *augmentation ideal* of $\mathbb{F}_p[H]$. It follows from [2, Proposition VIII.10.4] that if $n \geqslant |H|$, then $I_H^n = 0$. Since every element $\zeta(a_i) - 1$ belongs to I_H, one gets $\big(\zeta(a_1) - 1\big)\cdots\big(\zeta(a_n) - 1\big) \in I_H^n$ and hence $\big((\zeta(a_1) - 1)\cdots(\zeta(a_n) - 1)\big) = 0$. Formulas (4.1) and (4.2) now show that $\Delta^w f = \mathbf{0}$, which settles the case $r = 1$.

Suppose now that $r > 1$ and let $f\colon A^* \to G$ be a uniformly continuous function for the pro-p metric. By a standard result of group theory [13, Theorem 6.5, p. 116], G has a normal

subgroup C of order p. Now the quotient map $q\colon G \to G/C$ is uniformly continuous and so is $q \circ f\colon A^* \to G/C$. Since $|G/C| = p^{r-1}$, the induction hypothesis can be applied: there exists n such that for any word v in A^* of length $\geqslant n$, one has $\Delta^v(q \circ f) = \mathbf{1}$.

Since $\Delta^v(q \circ f) = q \circ (\Delta^v f)$ by Proposition 2.2, one has, for $|v| \geqslant n$, $q \circ (\Delta^v f) = \mathbf{1}$ and hence $\Delta^v f$ maps A^* into C. Note that $\Delta^v f$ is uniformly continuous by Proposition 4.4. Applying the first part of the proof to C, we get the following conclusion: for each v of length $\geqslant n$, there exists n_v such that for each word u of length at least n_v, one has $\Delta^u\Delta^v f = \mathbf{1}$. Let N be the maximum of all n_v taken over the finitely many v of length n. Then for each word w of length at least $N + n$, we may write $w = uv$, with $|v| = n$ and $|u| \geqslant N \geqslant n_v$. Then $\Delta^w f = \Delta^u\Delta^v f = \mathbf{1}$ and thus f is a Newton polynomial function. ◀

Putting Propositions 4.6 and 4.7 together, we get the main result of this section.

▶ **Theorem 4.8.** *Let G be a finite p-group. A function $f\colon A^* \to G$ is uniformly continuous for the pro-p metric if and only if it is a Newton polynomial function.*

5 Proof of the main result

We need two results on families of functions uniformly converging for the pro-p metric.

▶ **Proposition 5.1.** *Let $f\colon A^* \to F(B)$ be a function. If the elements $\delta_u f$, $u \in A^*$, tend to 1 when $|u|$ tends to ∞, then the sequence f_n tends uniformly to f.*

▶ **Proposition 5.2.** *A family of functions $(g_u\colon A^* \to F(B))_{u\in A^*}$ converges uniformly to the function $g\colon A^* \to F(B)$ when $|u|$ tends to infinity if and only if, for any finite p-group H and any morphism $\varphi\colon G \to H$, there exists N such that, for all $u \in A^*$ such that $|u| \geqslant N$, one has $\varphi \circ g_u = \varphi \circ g$.*

Proof of Theorem 1.2. The equivalence of (1) and (2) follows from Proposition 4.2. Let us prove that (2) implies (3). Let $f\colon A^* \to F(B)$ be uniformly continuous. Let H be any finite p-group and φ be any group morphism $F(B) \to H$. Since φ is uniformly continuous, so is $\varphi \circ f$. By Proposition 4.7, $\varphi \circ f$ is a Newton polynomial function and hence, for almost all $w \in A^*$, $\Delta^w(\varphi \circ f) = \mathbf{1}$. Thus, by Proposition 2.2, $\varphi \circ (\Delta^w f) = \mathbf{1}$. Thus by Proposition 5.2, (3) holds.

The implication (4) ⇒ (5) follows from Propositions 3.4 and 5.1. Note that (3) ⇒ (4) is clear, and (5) ⇒ (2) follows from general theorems of topology, since, by Proposition 4.6, Newton polynomial functions are uniformly continuous. ◀

6 Conclusion and perspectives

By combining topology, algebra, automata and combinatorics on words, we solved the synthesis problem for $\mathcal{G}_p$ in two different ways. Our results are based on a noncommutative extension of Mahler's theorem, a difficult mathematical result. In addition, we introduced two new concepts that would merit further study: the sequential product and Newton polynomial functions. We used the sequential product to solve the integration problem for a function f from A^* to a group G, knowing its initial value $f(1)$ and the functions $\Delta^a f$ for every letter a. We also proved that Newton's bijection induces a degree-preserving bijection between Newton polynomial functions and functions from A^* to G mapping almost every word to 1, a surprising combinatorial result.

Although these results offer exciting new perspectives, there is still a long way to go before one can solve the synthesis problem for regularity preserving functions. Solving the synthesis problem for other varieties of group languages is the next challenge.

References

1 Michaël Cadilhac, Olivier Carton, and Charles Paperman. Continuity and rational functions. In *44th International Colloquium on Automata, Languages, and Programming*, volume 80 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 115, 14. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2017.

2 Samuel Eilenberg. *Automata, Languages and Machines*, volume B. Academic Press, New York, 1976.

3 M. Lothaire. *Combinatorics on words*. Cambridge Mathematical Library. Cambridge University Press, Cambridge, 1997.

4 Kurt Mahler. An interpolation series for continuous functions of a p-adic variable. *J. Reine Angew. Math.*, 199:23–34, 1958. Correction **208** (1961), 70–72.

5 Kurt Mahler. A correction to the paper "An interpolation series for continuous functions of a p-adic variable.". *J. Reine Angew. Math.*, 208:70–72, 1961.

6 Jean-Éric Pin. Topologie p-adique sur les mots. *Journal de théorie des nombres de Bordeaux*, 5:263–281, 1993.

7 Jean-Éric Pin. Newton's forward difference equation for functions from words to words. In Arnold Beckmann, Victor Mitrana, and Mariya Soskova, editors, *Evolving Computability*, volume 9136 of *Lect. Notes Comp. Sci.*, pages 71–82. Springer International Publishing, 2015.

8 Jean-Éric Pin and Pedro V. Silva. A topological approach to transductions. *Theoret. Comput. Sci.*, 340:443–456, 2005.

9 Jean-Éric Pin and Pedro V. Silva. A Mahler's theorem for functions from words to integers. In Susanne Albers and Pascal Weil, editors, *25th International Symposium on Theoretical Aspects of Computer Science (STACS 2008)*, pages 585–596, Dagstuhl, Germany, 2008. Internationales Begegnungs- Und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

10 Jean-Éric Pin and Pedro V. Silva. On profinite uniform structures defined by varieties of finite monoids. *Internat. J. Algebra Comput.*, 21:295–314, 2011.

11 Jean-Éric Pin and Pedro V. Silva. A noncommutative extension of Mahler's theorem on interpolation series. *European J. Combin.*, 36:564–578, 2014.

12 Christophe Reutenauer and Marcel-Paul Schützenberger. Variétés et fonctions rationnelles. *Theoret. Comput. Sci.*, 145(1-2):229–240, 1995.

13 Harvey E. Rose. *A course on finite groups*. Universitext. Springer-Verlag London, Ltd., London, 2009.

On All Things Star-Free

Thomas Place
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400, Talence and IUF, France

Marc Zeitoun
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400, Talence, France

Abstract

We investigate the star-free closure, which associates to a class of languages its closure under Boolean operations and marked concatenation. We prove that the star-free closure of any finite class and of any class of groups languages with decidable separation (plus mild additional properties) has decidable separation. We actually show decidability of a stronger property, called covering. This generalizes many results on the subject in a unified framework. A key ingredient is that star-free closure coincides with another closure operator where Kleene stars are also allowed in restricted contexts.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Regular languages, separation problem, star-free closure, group languages

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.126

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version `https://arxiv.org/abs/1904.11863`

Funding DeLTA project (ANR-16-CE40-0007)

1 Introduction

This paper investigates a remarkable operation on classes of languages: the *star-free closure*. It builds a new class $SF(\mathcal{C})$ from an input class $\mathcal{C}$ by closing it under union, complement and concatenation. This generalizes an important specific class: the one of *star-free languages*, *i.e.*, the star-free closure of the class consisting of all finite languages. Star-free languages are those that can be defined in first order logic [12]. The correspondence was lifted to the quantifier alternation hierarchy of first order logic by Thomas [30], which corresponds to a classification of star-free languages: the dot-depth hierarchy [4]. These results extend to the star-free closure [22]. For each input class $\mathcal{C}$, $SF(\mathcal{C})$ corresponds to a variant of first-order logic (specified by the set of predicates that are allowed). Moreover, its quantifier alternation hierarchy corresponds to a classification of $SF(\mathcal{C})$: the concatenation hierarchy of basis $\mathcal{C}$.

Schützenberger proved that one may decide whether a regular language is star-free [27]. This result established a framework for investigating and understanding classes of languages, based on the *membership problem*: is it decidable to test whether an input regular language belongs to the class under investigation? Similar results were obtained for other prominent classes. Yet, this fruitful line of research also includes some of the most famous open problems in automata theory. For example, only the first levels of the dot-depth hierarchy are known to have decidable membership (see [14] for a survey).

Recently, these results were unified and generalized. First, the problem itself was strengthened: membership was replaced by separation as a means to investigate classes. The separation problem asks whether two input languages can be separated by one from the class under study. While more general and difficult than membership, separation is also more flexible. This was exploited to show that separation is decidable for several levels in the dot-depth hierarchy [19, 17]. In fact, this is a particular instance of a *generic* result applying to every hierarchy whose basis $\mathcal{C}$ is *finite* and satisfies some mild properties [18, 21]. Moreover,

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 126; pp. 126:1–126:14


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

the same result was obtained when the basis $\mathcal{C}$ is a class of group languages (*i.e.*, recognized by a finite group) with decidable separation [26]. Altogether, these results generalize most of the known results regarding the decidability of levels in concatenation hierarchies.

Contributions. This paper is a continuation of these research efforts. Instead of looking at levels within hierarchies, we investigate the star-free closure as a whole. First, we show that the star-free closure of a finite class has decidable separation. We then use this result to establish our main theorem: the star-free closure of a class of group languages with *decidable separation* has also decidable separation. In both cases, we actually prove the decidability of a stronger property called covering. Let us mention some important features of this work.

A first point is that the case of a finite class is important by itself. Foremost, it is a crucial step for the main result on the star-free closure of classes of group languages. Second, it yields a new proof that covering is decidable for the star-free languages (this is shown in [20] or can be derived from [9, 1]). This new proof is simpler and generic. While the original underlying technique goes back to Wilke [31], the proof has been simplified at several levels. The main simplification is obtained thanks to an abstract framework, introduced in [24]. It is based on the central notion of rating map, which is meant to measure the quality of a separator. For the framework to be relevant, we actually need to generalize separation to multiple input languages, which leads to the covering problem. Another key difference is that previously existing proofs (specific to the star-free languages) involve abstracting words by new letters at some point, which requires a relabeling procedure and a change of alphabet. Here, we cannot use this approach as the classes we build with star-free closure are less robust in general. We work with a fixed alphabet, which also makes the proof simpler.

A crucial ingredient in the proof is the notion of prefix code with bounded synchronization delay. Generalizing a definition of Schützenberger [28] which was also considered by Diekert and Walter [6, 7], we define a new closure operator that permits Kleene stars on such languages (this is a semantic property). This yields an operator that happens to coincide with the star-free closure when applied to the classes that we investigate. It serves as a key intermediary: in our proofs, we heavily rely on Kleene stars to construct languages. We therefore present this important step in the body of the paper (Theorem 7). Moreover, its proof provides yet another characterization of $SF(\mathcal{C})$, which is effective when the class $\mathcal{C}$ is finite (thus generalizing Schützenberger's membership result). At last regarding membership, it is worth pointing out that not only do we cover more cases, but also that it is straightforward to reprove the known algebraic characterizations from our results (see *e.g.*, [3]).

Finally, let us present important applications of our main result applying to input classes made of group languages. First, one may look at the input class containing all group languages. Straubing [29] described an algebraic counterpart of the star-free closure of this class, which was then shown to be recursive by Rhodes and Karnofsky [10]. Altogether, this implies that membership is decidable for the star-free closure of group languages, as noted by Margolis and Pin [11]. Here, we are able to generalize this result to separation and covering as separation is known to be decidable for the group languages [2].

Another important application is the class of languages definable by first-order logic with modular predicates $\mathrm{FO}(<, \mathrm{MOD})$. This class is known to have decidable membership [3]. Moreover, it is the star-free closure of the class consisting of the languages counting the length of words modulo some number. Since this input class is easily shown to have decidable separation (see [26] for example), our main theorem applies.

The third application applies to first-order logic endowed with predicates counting the number of occurrences of a letter before a position modulo some integer. The languages definable in this logic form the star-free closure of the languages recognized by Abelian groups:

this follows from a generic correspondence between star-free closure and variants of first-order logic [22, 13], together with the description of languages recognized by Abelian groups [8]. Our main theorem applies, since the class of Abelian groups has decidable separation [5, 1].

Organization. In Section 2, we recall some useful background. Section 3 presents a generic characterization of star-free closure. Then, Sections 4 and 5 are devoted to our two main theorems applying respectively to finite input classes and those made of group languages. Due to space limitations, several proofs are postponed to the full version of the paper [25].

2 Preliminaries

We fix a finite alphabet A for the whole paper. As usual, A^* denotes the set of all words over A, including the empty word ε. For $u, v \in A^*$, we denote by uv the word obtained by concatenating u and v. A *language* is a subset of A^*. We lift concatenation to languages: for $K, L \subseteq A^*$, we let $KL = \{uv \mid u \in K \text{ and } v \in L\}$. Finally, we use Kleene star: if $K \subseteq A^*$, K^+ denotes the union of all languages K^n for $n \geq 1$ and $K^* = K^+ \cup \{\varepsilon\}$.

A *class of languages* is a set of languages. A class $\mathcal{C}$ is a *Boolean algebra* when it is closed under union, intersection and complement. Moreover, $\mathcal{C}$ is *quotient-closed* if for every $L \in \mathcal{C}$ and $w \in A^*$, the languages $w^{-1}L \stackrel{\text{def}}{=} \{u \in A^* \mid wu \in L\}$ and $Lw^{-1} \stackrel{\text{def}}{=} \{u \in A^* \mid uw \in L\}$ belong to $\mathcal{C}$. All classes considered in the paper are quotient-closed Boolean algebras containing only *regular languages* (this will be implicit in our statements). These are the languages that can be equivalently defined by monadic second-order logic, finite automata or finite monoids. We briefly recall the monoid-based definition below.

We shall often consider *finite* quotient-closed Boolean algebras. If $\mathcal{C}$ is such a class, one may associate a canonical equivalence $\sim_{\mathcal{C}}$ over A^*. For $w, w' \in A^*$, $w \sim_{\mathcal{C}} w'$ if and only if $w \in L \Leftrightarrow w' \in L$ for every $L \in \mathcal{C}$. Moreover, we write $[w]_{\mathcal{C}} \in A^*/\sim_{\mathcal{C}}$ for the $\sim_{\mathcal{C}}$-class of w. One may then verify that the languages in $\mathcal{C}$ are exactly the unions of $\sim_{\mathcal{C}}$-classes. Moreover, since $\mathcal{C}$ is quotient-closed, $\sim_{\mathcal{C}}$ is a congruence for word concatenation (see [22] for proofs).

Regular languages. A *monoid* is a set M endowed with an associative multiplication $(s, t) \mapsto s \cdot t$ (also denoted by st) having a neutral element 1_M. An *idempotent* of a monoid M is an element $e \in M$ such that $ee = e$. It is folklore that for any *finite* monoid M, there exists a natural number $\omega(M)$ (denoted by ω when M is understood) such that s^ω is an idempotent for every $s \in M$. Observe that A^* is a monoid whose multiplication is concatenation (the neutral element is ε). Thus, we may consider monoid morphisms $\alpha : A^* \to M$ where M is an arbitrary monoid. Given such a morphism and $L \subseteq A^*$, we say that L is *recognized* by α when there exists a set $F \subseteq M$ such that $L = \alpha^{-1}(F)$. A language L is *regular* if and only if it is recognized by a morphism into a *finite* monoid. Moreover, it is known that there exists a canonical recognizer of L, which can be computed from any representation of L (such as a finite automaton): the syntactic morphism of L. We refer the reader to [15] for details.

Group languages. A group is a monoid G in which every element $g \in G$ has an inverse $g^{-1} \in G$, *i.e.*, $gg^{-1} = g^{-1}g = 1_G$. A "*group language*" is a language L recognized by a morphism into a *finite group*. All classes of group languages investigated here are quotient-closed Boolean algebras. Typically, publications on the topic consider *varieties* of group languages which is more restrictive: they involve an additional closure property called "inverse morphic image" (see [13]). For example, the class MOD described below is *not* a variety.

▶ **Example 1.** A simple example of quotient-closed Boolean algebra of group languages is the class of *all* group languages: GR. Another one is MOD, which contains the Boolean combinations of languages $\{w \in A^* \mid |w| = k \mod m\}$ with $k, m \in \mathbb{N}$ such that $k < m$.

Decision problems. We rely on three decision problems to investigate classes of languages. Each one depends on a parameter class $\mathcal{C}$, which we fix for the definition. The first problem, *$\mathcal{C}$-membership*, takes a single regular language L as input and asks whether $L \in \mathcal{C}$.

The second one, *$\mathcal{C}$-separation*, takes two regular languages L_1 and L_2 as input and asks whether L_1 is $\mathcal{C}$-separable from L_2 (is there a third language $K \in \mathcal{C}$ such that $L_1 \subseteq K$ and $L_2 \cap K = \emptyset$). This generalizes membership: $L \in \mathcal{C}$ if and only if L is $\mathcal{C}$-separable from $A^* \setminus L$.

The third problem, *$\mathcal{C}$-covering* was introduced in [24]. Given a language L, a *cover of* L is a finite set of languages $\mathbf{K}$ such that $L \subseteq \bigcup_{K \in \mathbf{K}} K$. Moreover, a $\mathcal{C}$-cover of L is a cover $\mathbf{K}$ of L such that all $K \in \mathbf{K}$ belong to $\mathcal{C}$. Consider a pair $(L_1, \mathbf{L}_2)$ where L_1 is a language and $\mathbf{L}_2$ is a *finite set of languages*. We say that $(L_1, \mathbf{L}_2)$ is *$\mathcal{C}$-coverable* when there exists a $\mathcal{C}$-cover $\mathbf{K}$ of L_1 such that for every $K \in \mathbf{K}$, there exists $L \in \mathbf{L}_2$ satisfying $K \cap L = \emptyset$. The *$\mathcal{C}$-covering problem* takes as input a single regular language L_1 and a finite set of regular languages $\mathbf{L}_2$. It asks whether $(L_1, \mathbf{L}_2)$ $\mathcal{C}$-coverable. Covering generalizes separation if $\mathcal{C}$ is closed under union: L_1 is $\mathcal{C}$-separable from L_2, if and only if $(L_1, \{L_2\})$ is $\mathcal{C}$-coverable (see [24]).

Star-free closure and main results. We investigate an operation defined on classes: *star-free closure*. Consider a class $\mathcal{C}$. The *star-free closure of* $\mathcal{C}$, denoted by $SF(\mathcal{C})$, is the least class containing $\mathcal{C}$ and the singletons $\{a\}$ for every $a \in A$, and closed under Boolean operations and concatenation. It is standard and simple to verify that when $\mathcal{C}$ is a quotient-closed Boolean algebra (which will always be the case here), this is also the case for $SF(\mathcal{C})$.

Our main theorems state conditions on the input class $\mathcal{C}$ guaranteeing decidability of our decision problems for $SF(\mathcal{C})$. First, we may handle *finite* classes.

▶ **Theorem 2.** *Let $\mathcal{C}$ be a finite quotient-closed Boolean algebra. Then, membership, separation and covering are decidable for $SF(\mathcal{C})$.*

The second theorem applies to input classes made of group languages.

▶ **Theorem 3.** *Let $\mathcal{C}$ be a quotient-closed Boolean algebra of group languages with decidable separation. Then, membership, separation and covering are decidable for $SF(\mathcal{C})$.*

The remainder of the paper is devoted to proving these theorems. We first focus on *$SF(\mathcal{C})$-membership* in Section 3. Naturally, this is weaker than directly handling $SF(\mathcal{C})$-covering. Yet, detailing membership independently allows to introduce many proof ideas and techniques that are needed to prove the "full" theorems. We detail these theorems in Sections 4 and 5. We only present the algorithms: proofs are deferred to the full paper [25].

3 Bounded synchronization delay and algebraic characterization

This section is devoted to $SF(\mathcal{C})$-membership. We handle it with a generic algebraic characterization of $SF(\mathcal{C})$ (effective under the hypotheses of Theorems 2 and 3), generalizing earlier work by Pin, Straubing and Thérien [29, 16]. We rely on an alternate definition of star-free closure involving a semantic restriction of the Kleene star, which we first present.

3.1 Bounded synchronization delay

We define a second operation on classes of languages $\mathcal{C} \mapsto SD(\mathcal{C})$. We shall later prove that it coincides with star-free closure (provided that $\mathcal{C}$ satisfies mild hypotheses). It is based on the work of Schützenberger [28] who defined a single class SD corresponding to the star-free languages (*i.e.*, $SF(\{\emptyset, A^*\})$). Here, we generalize it as an operation. The definition involves a semantic restriction of the Kleene star operation on languages: it may only be applied to "*prefix codes with bounded synchronization delay*". Introducing this notion requires basic definitions from coding theory that we first recall.

A language $K \subseteq A^*$ is a *prefix code* when $\varepsilon \notin K$ and $K \cap KA^+ = \emptyset$ (no word in K has a strict prefix in K). Note that this implies the following weaker property that we shall use implicitly: every $w \in K^*$ admits a *unique* decomposition $w = w_1 \cdots w_n$ with $w_1, \ldots, w_n \in K$ (this property actually defines *codes* which are more general).

Given $d \geq 1$, a prefix code $K \subseteq A^+$ has *synchronization delay* d if for every $u, v, w \in A^*$ such that $uvw \in K^+$ and $v \in K^d$, we have $uv \in K^+$. Finally, a prefix code $K \subseteq A^+$ has *bounded synchronization delay* when it has synchronization delay d for some $d \geq 1$.

▶ **Example 4.** Let $A = \{a, b\}$. Clearly, $\{ab\}$ is a prefix code with synchronization delay 1: if $uvw \in (ab)^+$ and $v = ab$, we have $uv \in (ab)^+$. Similarly, one may verify that $(aab)^*ab$ is a prefix code with synchronization delay 2 (but not 1). On the other hand, $\{aa\}$ does not have bounded synchronization delay. If $d \geq 1$, $a(aa)^d a \in (aa)^*$ but $a(aa)^d \notin (aa)^*$.

We present the operation $\mathcal{C} \mapsto SD(\mathcal{C})$. The definition involves *unambiguous concatenation*. Given $K, L \subseteq A^*$, their concatenation KL is *unambiguous* when every word $w \in KL$ admits a *unique* decomposition $w = uv$ with $u \in K$ and $v \in L$. Given a class $\mathcal{C}$, $SD(\mathcal{C})$ is the least class containing $\emptyset$ and $\{a\}$ for every $a \in A$, and closed under the following properties:

- **Intersection with $\mathcal{C}$:** if $K \in SD(\mathcal{C})$ and $L \in \mathcal{C}$, then $K \cap L \in SD(\mathcal{C})$.
- **Disjoint union:** if $K, L \in SD(\mathcal{C})$ are disjoint, then $K \uplus L \in SD(\mathcal{C})$.
- **Unambiguous product:** if $K, L \in SD(\mathcal{C})$ and KL is unambiguous, then $KL \in SD(\mathcal{C})$.
- **Kleene star for prefix codes with bounded synchronization delay:** if $K \in SD(\mathcal{C})$ is a prefix code with bounded synchronization delay, then $K^* \in SD(\mathcal{C})$.

▶ Remark 5. Schützenberger proved in [28] that $SD(\{\emptyset, A^*\}) = SF(\{\emptyset, A^*\})$. His definition of $SD(\{\emptyset, A^*\})$ was slightly less restrictive than ours: it does not require that the unions are disjoint and the concatenations unambiguous. It will be immediate from the correspondence with star-free closure that the two definitions are equivalent.

▶ Remark 6. This closure operation is different from standard ones. Instead of requiring that $\mathcal{C} \subseteq SD(\mathcal{C})$, we impose a stronger requirement: intersection with languages in $\mathcal{C}$ is allowed. If we only asked that $\mathcal{C} \subseteq SD(\mathcal{C})$, we would get a weaker operation which does not correspond to star-free closure in general. For example, let $A = \{a, b\}$ and consider the class MOD of Example 1. Observe that $(aa)^* \in SD(\text{MOD})$. Indeed, $\{a\} \in SD(\text{MOD})$ has bounded synchronization delay, $(AA)^* \in \text{MOD}$ and $(aa)^* = a^* \cap (AA)^*$. Yet, one may verify that $(aa)^*$ cannot be built from the languages of MOD with union, concatenation and Kleene star applied to prefix codes with bounded synchronization delay.

3.2 Algebraic characterization of star-free closure

We now reduce deciding membership for $SF(\mathcal{C})$ to computing $\mathcal{C}$-*stutters*. Let us first define this new notion. Let $\mathcal{C}$ be a quotient-closed Boolean algebra and $\alpha : A^* \to M$ be a morphism. A $\mathcal{C}$-*stutter* for α is an element $s \in M$ such that for every $\mathcal{C}$-cover $\mathbf{K}$ of $\alpha^{-1}(s)$, there exists

$K \in \mathbf{K}$ satisfying $K \cap KK \neq \emptyset$. When α is understood, we simply speak of a $\mathcal{C}$-stutter. Finally, we say that α is $\mathcal{C}$-*aperiodic* when for every $\mathcal{C}$-stutter $s \in M$, we have $s^\omega = s^{\omega+1}$. The reduction is stated in the following theorem.

▶ **Theorem 7.** *Let $\mathcal{C}$ be a quotient-closed Boolean algebra and consider a regular language $L \subseteq A^*$. The following properties are equivalent:*
1. $L \in SF(\mathcal{C})$.
2. $L \in SD(\mathcal{C})$.
3. *The syntactic morphism of L is $\mathcal{C}$-aperiodic.*

Naturally, the characterization need not be effective: this depends on $\mathcal{C}$. Deciding whether a morphism is $\mathcal{C}$-aperiodic boils down to computing $\mathcal{C}$-stutters. Yet, this is possible under the hypotheses of Theorems 2 and 3. First, if $\mathcal{C}$ is a finite quotient-closed Boolean algebra, deciding whether an element is a $\mathcal{C}$-stutter is simple: there are finitely many $\mathcal{C}$-covers and we may check them all. If $\mathcal{C}$ is a quotient-closed Boolean algebra of group languages, the question boils down to $\mathcal{C}$-separation as stated in the next lemma (proved in [25]).

▶ **Lemma 8.** *Let $\mathcal{C}$ be a quotient-closed Boolean algebra of group languages and $\alpha : A^* \to M$ be a morphism. For all $s \in M$, s is a $\mathcal{C}$-stutter if and only if $\{\varepsilon\}$ is **not** $\mathcal{C}$-separable from $\alpha^{-1}(s)$.*

Altogether, we obtain the membership part in Theorems 20 and 25. We conclude the section with an extended proof sketch for the most interesting direction in Theorem 7: 3) ⇒ 2) (a detailed proof for the two other directions is provided in the full version of this paper [25]).

Proof of 3) ⇒ 2) in Theorem 7. Let $\mathcal{C}$ be a quotient-closed Boolean algebra and $\alpha : A^* \to M$ be a $\mathcal{C}$-aperiodic morphism. We show that all languages recognized by α belong to $SD(\mathcal{C})$.

Given $K \subseteq A^*$ and $s \in M$, we say that K is s-*safe* when $s\alpha(u) = s\alpha(v)$ for every $u, v \in K$. We extend this notion to sets of languages: such a set $\mathbf{K}$ is s-safe when every $K \in \mathbf{K}$ is s-safe. We shall use s as an induction parameter. Finally, given a language $P \subseteq A^*$, an $SD(\mathcal{C})$-partition of P is a finite partition of P into languages of $SD(\mathcal{C})$.

▶ **Proposition 9.** *Let $P \subseteq A^+$ be a prefix code with bounded synchronization delay. Assume that there exists a 1_M-safe $SD(\mathcal{C})$-partition of P. Then, for every $s \in M$, there exists an s-safe $SD(\mathcal{C})$-partition of P^*.*

We first apply Proposition 9 to conclude the main argument. We show that every language recognized by α belongs to $SD(\mathcal{C})$. By definition, $SD(\mathcal{C})$ is closed under disjoint union. Hence, it suffices to show that $\alpha^{-1}(t) \in SD(\mathcal{C})$ for every $t \in M$. We fix $t \in M$ for the proof.

Clearly, $A \subseteq A^+$ is a prefix code with bounded synchronization delay and $\{\{a\} \mid a \in A\}$ is a 1_M-safe $SD(\mathcal{C})$-partition of A. Hence, Proposition 9 (applied in the case $s = 1_M$) yields a 1_M-safe $SD(\mathcal{C})$-partition $\mathbf{K}$ of A^*. One may verify that $\alpha^{-1}(t)$ is the disjoint union of all $K \in \mathbf{K}$ intersecting $\alpha^{-1}(t)$. Hence, $\alpha^{-1}(t) \in SD(\mathcal{C})$ which concludes the main argument.

It remains to prove Proposition 9. We let $P \subseteq A^*$ be a prefix code with bounded synchronization delay, $\mathbf{H}$ a 1_M-safe $SD(\mathcal{C})$-partition of P and $s \in M$. We need to build an $SD(\mathcal{C})$-partition $\mathbf{K}$ of P^* such that every $K \in \mathbf{K}$ is s-safe. We proceed by induction on the three following parameters listed by order of importance: (1) the size of $\alpha(P^+) \subseteq M$, (2) the size of $\mathbf{H}$ and (3) the size of $s \cdot \alpha(P^*) \subseteq M$. We distinguish two cases depending on the following property of s and $\mathbf{H}$. We say that s *is* $\mathbf{H}$-*stable* when the following holds:

$$\text{for every } H \in \mathbf{H}, \quad s \cdot \alpha(P^*) = s \cdot \alpha(P^*H). \tag{1}$$

The base case happens when s is $\mathbf{H}$-stable. Otherwise, we use induction on our parameters.

Base case: s is H-stable. Since α is $\mathcal{C}$-aperiodic, we have the following simple fact.

▶ **Fact 10.** *There is a* finite *quotient-closed Boolean algebra $\mathcal{D} \subseteq \mathcal{C}$ such that α is $\mathcal{D}$-aperiodic.*

Since $\mathcal{D}$ is finite, we may consider the associated canonical equivalence $\sim_{\mathcal{D}}$ over A^*. We let $\mathbf{K} = \{P^* \cap D \mid D \in A^*/\sim_{\mathcal{D}}\}$. Clearly, $\mathbf{K}$ is a partition of P^*. Let us verify that it only contains languages in $SD(\mathcal{C})$. We have $P \in SD(\mathcal{C})$: it is the disjoint union of all languages in the $SD(\mathcal{C})$-partition $\mathbf{H}$ of P. Moreover, $P^* \in SD(\mathcal{C})$ since P is a prefix code with bounded synchronization delay. Hence, $P^* \cap D \in SD(\mathcal{C})$ for every $D \in A^*/\sim_{\mathcal{D}}$ since $D \in \mathcal{D} \subseteq \mathcal{C}$. Therefore, it remains to show that every language $K \in \mathbf{K}$ is s-safe. This is a consequence of the following lemma which is proved using the hypothesis (1) that s is $\mathbf{H}$-stable.

▶ **Lemma 11.** *For every $u, v \in P^*$ such that $u \sim_{\mathcal{D}} v$, we have $s\alpha(u) = s\alpha(v)$.*

Inductive step: s is not H-stable. By hypothesis, we know that (1) does not hold. Therefore, we get some $H \in \mathbf{H}$ such that the following **strict** inclusion holds,

$$s \cdot \alpha(P^*H) \subsetneq s \cdot \alpha(P^*). \tag{2}$$

We fix this language $H \in \mathbf{H}$ for the remainder of the proof. The following lemma is proved by induction on our second parameter (the size of $\mathbf{H}$).

▶ **Lemma 12.** *There exists a 1_M-safe $SD(\mathcal{C})$-partition $\mathbf{U}$ of $(P \setminus H)^*$.*

We fix the partition $\mathbf{U}$ of $(P \setminus H)^*$ given by Lemma 12 and distinguish two independent subcases. Since $H \subseteq P$ (as H is an element of the partition $\mathbf{H}$ of P), we have $\alpha(P^*H) \subseteq \alpha(P^+)$. We use a different argument depending on whether this inclusion is strict or not.

Sub-case 1: $\alpha(P^*H) = \alpha(P^+)$. Since H is 1_M-safe by hypothesis, there exists $t \in M$ such that $\alpha(H) = \{t\}$. Similarly, since every $U \in \mathbf{U}$ is 1_M-safe, there exists $r_U \in M$ such that $\alpha(U) = \{r_U\}$. The construction of $\mathbf{K}$ is based on the next lemma which is proved using (2), the hypothesis of Sub-case 1 and induction on our third parameter (the size of $s \cdot \alpha(P^*) \subseteq M$).

▶ **Lemma 13.** *For every $U \in \mathbf{U}$, there exists an sr_Ut-safe $SD(\mathcal{C})$-partition $\mathbf{W}_U$ of P^*.*

We are ready to define the partition $\mathbf{K}$ of P^*. Using Lemma 13, we define,

$$\mathbf{K} = \mathbf{U} \cup \bigcup_{U \in \mathbf{U}} \{UHW \mid W \in \mathbf{W}_U\}$$

It remains to show that $\mathbf{K}$ is an s-safe $SD(\mathcal{C})$-partition of P^*. First, $\mathbf{K}$ is a partition of P^* since P is a prefix code and $H \subseteq P$. Indeed, every word $w \in P^*$ admits a *unique* decomposition $w = w_1 \cdots w_n$ with $w_1, \ldots, w_n \in P$. If no factor w_i belongs to H, then $w \in (P \setminus H)^*$ and w belongs to some unique $U \in \mathbf{U}$. Otherwise, let w_i be the leftmost factor such that $w_i \in H$. Thus, $w_1 \cdots w_{i-1} \in (P \setminus H)^*$, which also yields a unique $U \in \mathbf{U}$ such that $w_1 \cdots w_{i-1} \in U$ and $w_{i+1} \cdots w_n \in P^*$ which yields a unique $W \in \mathbf{W}_U$ such that $w_{i+1} \cdots w_n \in W$. Thus, $w \in UHW$ which is an element of $\mathbf{K}$ (the only one containing w).

Moreover, every $K \in \mathbf{K}$ belongs to $SD(\mathcal{C})$. If $K \in \mathbf{U}$, this is immediate by definition of $\mathbf{U}$ in Lemma 12. Otherwise, $K = UHW$ with $U \in \mathbf{U}$ and $W \in \mathbf{W}_U$. We know that $U, H, W \in SD(\mathcal{C})$ by definition. Moreover, one may verify that the concatenation UHW is *unambiguous* since P is a prefix code, $U \subseteq (P \setminus H)^*$ and $W \subseteq H^*$. Hence, $K \in SD(\mathcal{C})$.

Finally, we verify that $\mathbf{K}$ is s-safe. Consider $K \in \mathbf{K}$ and $w, w' \in K$, we show that $s\alpha(w) = s\alpha(w')$. If $K \in \mathbf{U}$, this is immediate: $\mathbf{U}$ is 1_M-safe by definition. Otherwise, $K = UHW$ with $U \in \mathbf{U}$ and $W \in \mathbf{W}_U$. By definition, $\alpha(H) = \{t\}$ and $\alpha(U) = \{r_U\}$ which implies that $s\alpha(w) = sr_U t\alpha(x)$ and $s\alpha(w') = sr_U t\alpha(x')$ for $x, x' \in W$. Moreover, $W \in \mathbf{W}_U$ is $sr_U t$-safe by definition. Hence, $s\alpha(w) = s\alpha(w')$, which concludes the proof of this sub-case.

Sub-case 2: $\boldsymbol{\alpha(P^*H) \subsetneq \alpha(P^+)}$. Consider $w \in P^*$. Since P is a prefix code, w admits a unique decomposition $w = w_1 \cdots w_n$ with $w_1, \ldots, w_n \in P$. We may look at the rightmost factor $w_i \in H \subseteq P$ to uniquely decompose w in two parts (each of them possibly empty): the prefix $w_1 \cdots w_i \in ((P \setminus H)^* H)^*$ and the suffix in $w_{i+1} \cdots w_n \in (P \setminus H)^*$. Using induction, we construct $SD(\mathcal{C})$-partitions of the possible languages of prefixes and suffixes. Then, we combine them to construct a partition of the whole set P^*. We already handled the suffixes: $\mathbf{U}$ is an $SD(\mathcal{C})$-partition of $(P \setminus H)^*$. The prefixes are handled using the hypothesis of Sub-case 2 and induction on our first parameter (the size of $\alpha(P^+)$).

▶ **Lemma 14.** *There exists a 1_M-safe $SD(\mathcal{C})$-partition $\mathbf{V}$ of $((P \setminus H)^* H)^*$.*

Using Lemma 14, we define $\mathbf{K} = \{VU \mid V \in \mathbf{V} \text{ and } U \in \mathbf{U}\}$. It follows from the above discussion that $\mathbf{K}$ is a partition of P^* since $\mathbf{V}$ and $\mathbf{U}$ are partitions of $((P \setminus H)^* H)^*$ and $(P \setminus H)^*$, respectively. Moreover, every $K \in \mathbf{K}$ belongs to $SD(\mathcal{C})$: $K = VU$ with $V \in \mathbf{V}$ and $U \in \mathbf{U}$, and one may verify that this is an *unambiguous* concatenation. It remains to show that $\mathbf{K}$ is s-safe. Let $K \in \mathbf{K}$ and $w, w' \in K$. We show that $s\alpha(w) = s\alpha(w')$. By definition, we have $K = VU$ with $V \in \mathbf{V}$ and $U \in \mathbf{U}$. Therefore, $w = vu$ and $w' = v'u'$ with $u, u' \in U$ and $v, v' \in V$. Since U and V are both 1_M-safe by definition, we have $\alpha(u) = \alpha(u')$ and $\alpha(v) = \alpha(v')$. It follows that $s\alpha(w) = s\alpha(w')$, which concludes the proof of Proposition 9. ◀

4 Covering when the input class is finite

This section is devoted to Theorem 2. We show that when $\mathcal{C}$ is a finite quotient-closed Boolean algebra, $SF(\mathcal{C})$-covering is decidable by presenting a generic algorithm. It is formulated within a framework designed to handle covering questions, which was originally introduced in [24]. We start by briefly recalling it (we refer the reader to [24] for details).

4.1 Rating maps and optimal imprints

The framework is based on an algebraic object called "rating map". These are morphisms of commutative and idempotent monoids. We write such monoids $(R, +)$: the binary operation "$+$" is called *addition* and the neutral element is denoted by 0_R. Being idempotent means that $r + r = r$ for every $r \in R$. For every commutative and idempotent monoid $(R, +)$, one may define a canonical ordering $\leq$ over R: for $r, s \in R$, we have $r \leq s$ when $r + s = s$. One may verify that $\leq$ is a partial order which is compatible with addition.

▶ **Example 15.** For every set E, $(2^E, \cup)$ is an idempotent and commutative monoid. The neutral element is $\emptyset$ and the canonical ordering is inclusion.

A rating map is a morphism $\rho : (2^{A^*}, \cup) \to (R, +)$ where $(R, +)$ is a *finite* idempotent and commutative monoid, called the *rating set of* ρ. That is, ρ is a map from 2^{A^*} to R such that $\rho(\emptyset) = 0_R$ and $\rho(K_1 \cup K_2) = \rho(K_1) + \rho(K_2)$ for every $K_1, K_2 \subseteq A^*$.

For the sake of improved readability, when applying a rating map ρ to a singleton set $\{w\}$, we write $\rho(w)$ for $\rho(\{w\})$. Moreover, we write $\rho_* : A^* \to R$ for the restriction of ρ to A^*: for every $w \in A^*$, we have $\rho_*(w) = \rho(w)$ (this notation is useful when referring to the language $\rho_*^{-1}(r) \subseteq A^*$, which consists of all words $w \in A^*$ such that $\rho(w) = r$).

Most of the theory makes sense for arbitrary rating maps. However, we shall often have to work with special rating maps satisfying additional properties. We define two kinds.

Nice rating maps. A rating map $\rho : 2^{A^*} \to R$ is *nice* when, for every nonempty language $K \subseteq A^*$, there exist finitely many words $w_1, \ldots, w_n \in K$ such that $\rho(K) = \rho(w_1) + \cdots + \rho(w_k)$.

When a rating map $\rho : 2^{A^*} \to R$ is nice, it is characterized by the canonical map $\rho_* : A^* \to R$. Indeed, for $K \subseteq A^*$, we may consider the sum of all elements $\rho(w)$ for $w \in K$: while it may be infinite, this sum boils down to a finite one since R is commutative and idempotent. The hypothesis that ρ is nice implies that $\rho(K)$ is equal to this sum.

Multiplicative rating maps. A rating map $\rho : 2^{A^*} \to R$ is multiplicative when its rating set R has more structure: it needs to be an *idempotent semiring*. A *semiring* is a tuple $(R, +, \cdot)$ where R is a set and "$+$" and "$\cdot$" are two binary operations called addition and multiplication. Moreover, $(R, +)$ is a commutative monoid, $(R, \cdot)$ is a monoid (the neutral element is denoted by 1_R), the multiplication distributes over addition and the neutral element "0_R" of $(R, +)$ is a zero for $(R, \cdot)$ ($0_R \cdot r = r \cdot 0_R = 0_R$ for every $r \in R$). A semiring R is *idempotent* when $r + r = r$ for every $r \in R$, *i.e.*, when the additive monoid $(R, +)$ is idempotent (there is no additional constraint on the multiplicative monoid $(R, \cdot)$).

▶ **Example 16.** A key example of an infinite idempotent semiring is the set 2^{A^*}. Union is the addition and language concatenation is the multiplication (with $\{\varepsilon\}$ as neutral element).

Let $\rho : 2^{A^*} \to R$ be a rating map: $(R, +)$ is an idempotent commutative monoid and ρ is a morphism from $(2^{A^*}, \cup)$ to $(R, +)$. We say that ρ is *multiplicative* when the rating set R is equipped with a multiplication "$\cdot$" such that $(R, +, \cdot)$ is an idempotent semiring and ρ is also a monoid morphism from $(2^{A^*}, \cdot)$ to $(R, \cdot)$. That is, the two following additional axioms have to be satisfied: $\rho(\varepsilon) = 1_R$ and $\rho(K_1 K_2) = \rho(K_1) \cdot \rho(K_2)$ for every $K_1, K_2 \subseteq A^*$.

▶ Remark 17. Rating maps which are both nice and multiplicative are finitely representable. As we explained, if $\rho : 2^{A^*} \to R$ is nice, it is characterized by the canonical map $\rho_* : A^* \to R$. When ρ is also multiplicative, ρ_* is finitely representable: it is a morphism into a finite monoid. Hence, we may speak of algorithms whose input is a nice multiplicative rating map.

Rating maps which are not nice and multiplicative cannot be finitely represented in general. Yet, they are crucial: while our main statements consider nice multiplicative rating maps, many proofs involve auxiliary rating maps which are neither nice nor multiplicative.

Optimal imprints. Now that we have rating maps, we turn to imprints. Consider a rating map $\rho : 2^{A^*} \to R$. Given any finite set of languages $\mathbf{K}$, we define the ρ-imprint of $\mathbf{K}$. Intuitively, when $\mathbf{K}$ is a cover of some language L, this object measures the "quality" of $\mathbf{K}$. The ρ-imprint *of* $\mathbf{K}$ is the following subset of R:

$$\mathcal{I}[\rho](\mathbf{K}) = \{r \mid r \leq \rho(K) \text{ for some } K \in \mathbf{K}\}.$$

We may now define optimality. Consider an arbitrary rating map $\rho : 2^{A^*} \to R$ and a Boolean algebra $\mathcal{C}$. Given a language L, an optimal $\mathcal{C}$-cover of L for ρ is a $\mathcal{C}$-cover $\mathbf{K}$ of L which satisfies the following property:

$$\mathcal{I}[\rho](\mathbf{K}) \subseteq \mathcal{I}[\rho](\mathbf{K}') \quad \text{for every } \mathcal{C}\text{-cover } \mathbf{K}' \text{ of } L.$$

In general, there can be infinitely many optimal $\mathcal{C}$-covers for a given rating map ρ. It is shown in [24] that there always exists at least one (using closure under intersection for $\mathcal{C}$).

Clearly, for a Boolean algebra $\mathcal{C}$, a language L and a rating map ρ, all optimal $\mathcal{C}$-covers of L for ρ have the same ρ-imprint. Hence, this unique ρ-imprint is a *canonical* object for $\mathcal{C}$, L and ρ. We call it the *$\mathcal{C}$-optimal ρ-imprint on L* and we write it $\mathcal{I}_{\mathcal{C}}[L, \rho]$:

$$\mathcal{I}_{\mathcal{C}}[L, \rho] = \mathcal{I}[\rho](\mathbf{K}) \quad \text{for any optimal } \mathcal{C}\text{-cover } \mathbf{K} \text{ of } L \text{ for } \rho.$$

We complete the definition with a simple useful fact (a proof is available in [23]).

▶ **Fact 18.** *Let $\mathcal{C}$ be a Boolean algebra, $\rho : 2^{A^*} \to R$ a rating map and $L_1, L_2 \subseteq A^*$. Then, $\mathcal{I}_{\mathcal{C}}[L_1, \rho] \cup \mathcal{I}_{\mathcal{C}}[L_2, \rho] = \mathcal{I}_{\mathcal{C}}[L_1 \cup L_2, \rho]$.*

Connection with covering. Consider the special case when the language L that needs to be covered is A^*. In that case, we write $\mathcal{I}_{\mathcal{C}}[\rho]$ for $\mathcal{I}_{\mathcal{C}}[A^*, \rho]$. It is shown in [24] that for every *Boolean algebra $\mathcal{C}$*, deciding $\mathcal{C}$-covering formally reduces to computing $\mathcal{C}$-optimal imprints from input nice multiplicative rating maps.

▶ **Proposition 19.** *Let $\mathcal{C}$ be a Boolean algebra. Assume that there exists an algorithm which computes $\mathcal{I}_{\mathcal{C}}[\rho]$ from an input nice multiplicative rating map ρ. Then, $\mathcal{C}$-covering is decidable.*

4.2 Algorithm

We may now present our algorithm for $SF(\mathcal{C})$-covering when $\mathcal{C}$ is a finite quotient-closed Boolean algebra. We fix $\mathcal{C}$ for the presentation. In view of Proposition 19, we need to prove that one may compute $\mathcal{I}_{SF(\mathcal{C})}[\rho]$ from an input nice multiplicative rating map ρ.

Our algorithm actually computes slightly more information. Since $\mathcal{C}$ is a finite quotient-closed Boolean algebra, we may consider the equivalence $\sim_{\mathcal{C}}$ over A^*. In particular, the set $A^*/\!\sim_{\mathcal{C}}$ of $\sim_{\mathcal{C}}$-classes is a finite monoid (we write "$\bullet$" for its multiplication) and the map $w \mapsto [w]_{\mathcal{C}}$ is a morphism. Given a rating map $\rho : 2^{A^*} \to R$ we define:

$$\mathcal{P}^{\mathcal{C}}_{SF(\mathcal{C})}[\rho] = \{(C, r) \in (A^*/\!\sim_{\mathcal{C}}) \times R \mid r \in \mathcal{I}_{SF(\mathcal{C})}[C, \rho]\}$$

Observe that $\mathcal{P}^{\mathcal{C}}_{SF(\mathcal{C})}[\rho]$ captures more information than $\mathcal{I}_{SF(\mathcal{C})}[\rho]$. Indeed, it encodes all sets $\mathcal{I}_{SF(\mathcal{C})}[C, \rho]$ for $C \in A^*/\!\sim_{\mathcal{C}}$ and by Fact 18, $\mathcal{I}_{SF(\mathcal{C})}[\rho]$ is the union of all these sets.

Our main result is a least fixpoint procedure for computing $\mathcal{P}^{\mathcal{C}}_{SF(\mathcal{C})}[\rho]$ from a nice multiplicative rating map ρ. It is based on a generic characterization theorem which we first present. Given an arbitrary nice multiplicative rating map $\rho : 2^{A^*} \to R$ and a set $S \subseteq (A^*/\!\sim_{\mathcal{C}}) \times R$, we say that S is *$SF(\mathcal{C})$-saturated for ρ* when the following properties are satisfied:

1. **Trivial elements.** For every $w \in A^*$, we have $([w]_{\mathcal{C}}, \rho(w)) \in S$.
2. **Downset.** For every $(C, r) \in S$ and $q \in R$, if $q \leq r$, then $(C, q) \in S$.
3. **Multiplication.** For every $(C, q), (D, r) \in S$, we have $(C \bullet D, qr) \in S$.
4. ***SF*($\mathcal{C}$)-closure.** For all $(E, r) \in S$, if $E \in A^*/\!\sim_{\mathcal{C}}$ is idempotent, then $(E, r^{\omega} + r^{\omega+1}) \in S$.

▶ **Theorem 20** ($SF(\mathcal{C})$-optimal imprints ($\mathcal{C}$ finite))**.** *Let $\rho : 2^{A^*} \to R$ be a* nice *multiplicative rating map. Then, $\mathcal{P}^{\mathcal{C}}_{SF(\mathcal{C})}[\rho]$ is the least $SF(\mathcal{C})$-saturated subset of $(A^*/\!\sim_{\mathcal{C}}) \times R$ for ρ.*

Given a nice multiplicative rating map $\rho : 2^{A^*} \to R$ as input, it is clear that one may compute the least $SF(\mathcal{C})$-saturated subset of $(A^*/\!\sim_{\mathcal{C}}) \times R$ with a least fixpoint procedure. Hence, Theorem 20 provides an algorithm for computing $\mathcal{P}^{\mathcal{C}}_{SF(\mathcal{C})}[\rho]$. As we explained above, we may then compute $\mathcal{I}_{SF(\mathcal{C})}[\rho]$ from this set. Together with Proposition 19, this yields Theorem 2 as a corollary: $SF(\mathcal{C})$-covering is decidable when $\mathcal{C}$ is a finite quotient-closed Boolean algebra. Theorem 20 is proved in the full version of this paper [25].

5 Covering when the input class is made of group languages

This section is devoted to Theorem 3. We show that when $\mathcal{C}$ is a quotient-closed Boolean algebra of group languages with decidable separation, $SF(\mathcal{C})$-covering is decidable.

As in Section 4, we rely on Proposition 19: we present an algorithm computing $\mathcal{I}_{SF(\mathcal{C})}[\rho]$ from an input nice multiplicative rating map ρ. We do not work with $\mathcal{I}_{SF(\mathcal{C})}[\rho]$ itself but with another set carrying more information. Its definition requires introducing a few additional concepts. We first present them and then turn to the algorithm. For more details, see [26].

5.1 Preliminary definitions

Optimal ε-approximations. In this case, handling $SF(\mathcal{C})$ involves considering $\mathcal{C}$-optimal covers of $\{\varepsilon\}$. Since $\{\varepsilon\}$ is a singleton, there always exists such a cover consisting of a single language, which leads to the following definition.

Let $\mathcal{C}$ be a Boolean algebra (we shall use the case when $\mathcal{C}$ contains only group languages but this is not required for the definitions) and $\tau : 2^{A^*} \to Q$ a rating map. A *$\mathcal{C}$-optimal ε-approximation for τ* is a language $L \in \mathcal{C}$ such that $\varepsilon \in L$ and $\tau(L) \leq \tau(L')$ for every $L' \in \mathcal{C}$ satisfying $\varepsilon \in L'$. As expected, there always exists a $\mathcal{C}$-optimal ε-approximation for any rating map τ (see the full version of this paper [25] for a proof).

By definition, all $\mathcal{C}$-optimal ε-approximations for τ have the same image under τ. We write it $\mathbb{i}_{\mathcal{C}}[\tau] \in Q$: $\mathbb{i}_{\mathcal{C}}[\tau] = \tau(L)$ for every $\mathcal{C}$-optimal ε-approximation L for τ. It turns out that when τ is nice and multiplicative, computing $\mathbb{i}_{\mathcal{C}}[\tau]$ from τ boils down to $\mathcal{C}$-separation. This is important: this is exactly how our algorithm for $SF(\mathcal{C})$-covering depends on $\mathcal{C}$-separation.

▶ **Lemma 21.** *Let $\tau : 2^{A^*} \to Q$ be a nice rating map and $\mathcal{C}$ a Boolean algebra. Then, $\mathbb{i}_{\mathcal{C}}[\tau]$ is the sum of all $q \in Q$ such that $\{\varepsilon\}$ is not $\mathcal{C}$-separable from $\tau_*^{-1}(q)$.*

Nested rating maps. We want an algorithm which computes $\mathcal{I}_{SF(\mathcal{C})}[\rho]$ from an input nice multiplicative rating map ρ for a fixed quotient-closed Boolean algebra of group languages $\mathcal{C}$. Yet, we shall not use optimal ε-approximations with this input rating map ρ. Instead, we consider an auxiliary rating map built from ρ (the definition is taken from [23]).

Consider a Boolean algebra $\mathcal{D}$ (we shall use the case $\mathcal{D} = SF(\mathcal{C})$) and a rating map $\rho : 2^{A^*} \to R$. We build a new map $\xi_{\mathcal{D}}[\rho] : 2^{A^*} \to 2^R$ whose rating set is $(2^R, \cup)$. For every $K \subseteq A^*$, we define $\xi_{\mathcal{D}}[\rho](K) = \mathcal{I}_{\mathcal{D}}[K, \rho] \in 2^R$. It follows from Fact 18 that this is indeed a rating map (on the other hand $\xi_{\mathcal{D}}[\rho]$ need not be nice nor multiplicative, see [23] for details).

We may now explain which set is computed by our algorithm instead of $\mathcal{I}_{SF(\mathcal{C})}[\rho]$. Consider a nice multiplicative rating map $\rho : 2^{A^*} \to R$. Since $\xi_{SF(\mathcal{C})}[\rho] : 2^{A^*} \to 2^R$ is a rating map, we may consider the element $\mathbb{i}_{\mathcal{C}}[\xi_{SF(\mathcal{C})}[\rho]] \in 2^R$. By definition, $\mathbb{i}_{\mathcal{C}}[\xi_{SF(\mathcal{C})}[\rho]] = \xi_{SF(\mathcal{C})}[\rho](L)$ where L is a $\mathcal{C}$-optimal ε-approximation for $\xi_{SF(\mathcal{C})}[\rho]$. Therefore, $\mathbb{i}_{\mathcal{C}}[\xi_{SF(\mathcal{C})}[\rho]]$ is a subset of $\xi_{SF(\mathcal{C})}[\rho](A^*) = \mathcal{I}_{SF(\mathcal{C})}[A^*, \rho] = \mathcal{I}_{SF(\mathcal{C})}[\rho]$. When $\mathcal{C}$ is a quotient-closed Boolean algebra of group languages, one may compute the whole set $\mathcal{I}_{SF(\mathcal{C})}[\rho]$ from this subset.

▶ **Proposition 22.** *Let $\mathcal{C}$ be a quotient-closed Boolean algebra of group languages and $\rho : 2^{A^*} \to R$ a nice multiplicative rating map. Then, $\mathcal{I}_{SF(\mathcal{C})}[\rho]$ is the least subset of R containing $\mathbb{i}_{\mathcal{C}}[\xi_{SF(\mathcal{C})}[\rho]]$ and satisfying the three following properties:*

- ***Trivial elements.*** *For every $w \in A$, $\rho(w) \in \mathcal{I}_{SF(\mathcal{C})}[\rho]$.*
- ***Downset.*** *For every $r \in \mathcal{I}_{SF(\mathcal{C})}[\rho]$ and $q \leq r$, we have $q \in \mathcal{I}_{SF(\mathcal{C})}[\rho]$.*
- ***Multiplication.*** *For every $q, r \in \mathcal{I}_{SF(\mathcal{C})}[\rho]$, we have $qr \in \mathcal{I}_{SF(\mathcal{C})}[\rho]$.*

▶ Remark 23. Intuitively, we use $\mathbb{i}_{\mathcal{C}}[\xi_{SF(\mathcal{C})}[\rho]]$ to "nest" two optimizations: one for $\mathcal{C}$ and the other for $SF(\mathcal{C})$. Indeed, $\mathbb{i}_{\mathcal{C}}[\xi_{SF(\mathcal{C})}[\rho]] = \xi_{SF(\mathcal{C})}[\rho](L) = \mathcal{I}_{SF(\mathcal{C})}[L, \rho]$ where L is a $\mathcal{C}$-optimal ε-approximation for $\xi_{SF(\mathcal{C})}[\rho]$. Hence, $\mathbb{i}_{\mathcal{C}}[\xi_{SF(\mathcal{C})}[\rho]]$ is least set $\mathcal{I}[\rho](\mathbf{K}) \subseteq R$ (with respect to inclusion), over all $SF(\mathcal{C})$-covers $\mathbf{K}$ of some language $L \in \mathcal{C}$ containing ε.

5.2 Algorithm

We may now present our algorithm for computing $\mathcal{I}_{SF(\mathcal{C})}[\rho]$. We fix a quotient-closed Boolean algebra of group languages $\mathcal{C}$ for the presentation. As expected, the main procedure computes $\mathbb{i}_{\mathcal{C}}[\xi_{SF(\mathcal{C})}[\rho]]$ (see Proposition 22). In this case as well, this procedure is obtained from a characterization theorem.

Consider a nice multiplicative rating map $\rho : 2^{A^*} \to R$. We define the $SF(\mathcal{C})$-complete subsets of R for ρ. The definition depends on auxiliary nice multiplicative rating maps. We first present them. Clearly, 2^R is an idempotent semiring (addition is union and the multiplication is lifted from the one of R). For every $S \subseteq R$, we use it as the rating set of a nice multiplicative rating map $\eta_{\rho,S} : 2^{A^*} \to 2^R$. Since we are defining a *nice* multiplicative rating map, it suffices to specify the evaluation of letters. For $a \in A$, we let $\eta_{\rho,S}(a) = S \cdot \{\rho(a)\} \cdot S \in 2^R$. Observe that by definition, we have $\mathbb{i}_{\mathcal{C}}[\eta_{\rho,S}] \subseteq R$.

We are ready to define the $SF(\mathcal{C})$-complete subsets of R. Consider $S \subseteq R$. We say that S is *$SF(\mathcal{C})$-complete for ρ* when the following conditions are satisfied:

1. **Downset.** For every $r \in S$ and $q \leq r$, we have $q \in S$.
2. **Multiplication.** For every $q, r \in S$, we have $qr \in S$.
3. **$\mathcal{C}$-operation.** We have $\mathbb{i}_{\mathcal{C}}[\eta_{\rho,S}] \subseteq S$.
4. **$SF(\mathcal{C})$-closure.** For every $r \in S$, we have $r^\omega + r^{\omega+1} \in S$.

▶ Remark 24. The definition of $SF(\mathcal{C})$-complete subsets does not explicitly require that they contain some trivial elements. Yet, this is implied by $\mathcal{C}$-operation. Indeed, if $S \subseteq R$ is $SF(\mathcal{C})$-complete, then $\eta_{\rho,S}(\varepsilon) = \{1_R\}$ (this is the multiplicative neutral element of 2^R). This implies that $1_R \in \mathbb{i}_{\mathcal{C}}[\eta_{\rho,S}]$ and we obtain from $\mathcal{C}$-operation that $1_R \in S$.

▶ **Theorem 25** ($SF(\mathcal{C})$-optimal imprints ($\mathcal{C}$ made of group languages))**.** *Let $\rho : 2^{A^*} \to R$ be a nice multiplicative rating map. Then, $\mathbb{i}_{\mathcal{C}}[\xi_{SF(\mathcal{C})}[\rho]]$ is the least $SF(\mathcal{C})$-complete subset of R.*

When $\mathcal{C}$-separation is decidable, Theorem 25 yields a least fixpoint procedure for computing $\mathbb{i}_{\mathcal{C}}[\xi_{SF(\mathcal{C})}[\rho]]$ from a nice multiplicative rating map $\rho : 2^{A^*} \to R$. The computation starts from the empty set and saturates it with the four operations in the definition of $SF(\mathcal{C})$-complete subsets. It is clear that we may implement downset, multiplication and $SF(\mathcal{C})$-closure. Moreover, we may implement $\mathcal{C}$-operation as this boils down to $\mathcal{C}$-separation by Lemma 21. Eventually, the computation reaches a fixpoint and it is straightforward to verify that this set is the least $SF(\mathcal{C})$-complete subset of R, *i.e.*, $\mathbb{i}_{\mathcal{C}}[\xi_{SF(\mathcal{C})}[\rho]]$ by Theorem 25.

By Proposition 22, we may compute $\mathcal{I}_{SF(\mathcal{C})}[\rho]$ from $\mathbb{i}_{\mathcal{C}}[\xi_{SF(\mathcal{C})}[\rho]]$. Altogether, this yields the decidability of $SF(\mathcal{C})$-covering by Proposition 19. Hence, Theorem 3 is proved.

6 Conclusion

We proved that for any quotient-closed Boolean algebra $\mathcal{C}$, $SF(\mathcal{C})$-covering is decidable whenever $\mathcal{C}$ is either finite or made of group languages and with decidable separation. Moreover, we presented an algebraic characterization of $SF(\mathcal{C})$ which holds for every quotient-closed Boolean algebra $\mathcal{C}$, generalizing earlier results [29, 16]. A key proof ingredient is an

alternative definition of star-free closure: the operation $\mathcal{C} \mapsto SD(\mathcal{C})$ which we prove to be equivalent. This correspondence generalizes the work of Schützenberger [28] who introduced a single class SD (i.e. $SD(\{\emptyset, A^*\})$) corresponding to the star-free languages (i.e. $SF(\{\emptyset, A^*\})$).

Our results can be instantiated for several input classes $\mathcal{C}$. Theorem 2 applies when $\mathcal{C}$ is finite. In this case, the only prominent application is the class of star-free languages itself. It was already known that covering is decidable for this class [9, 20]. However, Theorem 2 is important for two reasons. First, its proof is actually simpler than the earlier ones specific to the star-free languages (this is achieved by relying on the operation $\mathcal{C} \mapsto SD(\mathcal{C})$). More importantly, Theorem 2 is used as a key ingredient for proving our second generic statement: Theorem 3, which applies to classes made of group languages with decidable separation. It is known that separation is decidable for the class GR of all group languages [2]. Hence, we obtain that $SF(\mathrm{GR})$-covering is decidable. Another application is the class MOD consisting of languages counting the length of words modulo some number (deciding MOD-separation is a simple exercise). We get the decidability of $SF(\mathrm{MOD})$-covering. This is important, as the languages in $SF(\mathrm{MOD})$ are those definable in first-order logic with modular predicates $(\mathrm{FO}(<, \mathrm{MOD}))$. A last example is given by the input class consisting of all languages counting the number of occurrences of letters modulo some number. These are exactly the languages recognized by finite commutative groups, for which separation is decidable [5].

References

1 Jorge Almeida. Some Algorithmic Problems for Pseudovarieties. *Publicationes Mathematicae Debrecen*, 54:531–552, 1999.

2 Christopher J. Ash. Inevitable Graphs: a Proof of the Type II Conjecture and some Related Decision Procedures. *IJAC*, 1(1):127–146, 1991.

3 David A. Mix Barrington, Kevin Compton, Howard Straubing, and Denis Thérien. Regular languages in NC1. *Journal of Computer and System Sciences*, 44(3):478–499, 1992.

4 Janusz A. Brzozowski and Rina S. Cohen. Dot-Depth of Star-Free Events. *J. Comp. Sys. Sci.*, 5(1):1–16, 1971.

5 Manuel Delgado. Abelian pointlikes of a monoid. *Semigroup Forum*, 56:339–361, 1998.

6 Volker Diekert and Tobias Walter. Characterizing classes of regular languages using prefix codes of bounded synchronization delay. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming*, ICALP'16, pages 129:1–129:14, 2016.

7 Volker Diekert and Tobias Walter. Characterizing classes of regular languages using prefix codes of bounded synchronization delay. *IJAC*, 27(6):561–590, 2017.

8 Samuel Eilenberg. *Automata, languages, and machines*, volume B. Academic Press, 1976.

9 Karsten Henckell. Pointlike sets: the finest aperiodic cover of a finite semigroup. *Journal of Pure Applied Algebra*, 55(1-2):85–126, 1988.

10 Joel Karnofsky and John Rhodes. Decidability of complexity one-half for finite semigroups. *Semigroup Forum*, 24(1):55–66, 1982. doi:10.1007/BF02572755.

11 Stuart W. Margolis and Jean-Éric Pin. Products of group languages. In *FCT'85*. Springer, 1985.

12 Robert McNaughton and Seymour A. Papert. *Counter-Free Automata*. MIT Press, 1971.

13 Jean-Éric Pin. Bridges for Concatenation Hierarchies. In *ICALP'98*, pages 431–442. Springer, 1998.

14 Jean-Éric Pin. The dot-depth hierarchy, 45 years later. In *The Role of Theory in Computer Science. Essays Dedicated to Janusz Brzozowski*. World Scientific Publishing, 2017.

15 Jean-Éric Pin. Mathematical Foundations of Automata Theory. Lecture notes, 2019. URL: https://www.irif.fr/~jep/PDF/MPRI/MPRI.pdf.

16 Jean-Éric Pin, Howard Straubing, and Denis Thérien. Locally trivial categories and unambiguous concatenation. *Journal of Pure and Applied Algebra*, 52(3):297–311, 1988.

17 Thomas Place. Separating Regular Languages with Two Quantifiers Alternations. In *LICS'15*, pages 202–213. IEEE Computer Society, 2015.

18 Thomas Place. Separating regular languages with two quantifier alternations. *Logical Methods in Computer Science*, 14(4), 2018.

19 Thomas Place and Marc Zeitoun. Going Higher in the First-Order Quantifier Alternation Hierarchy on Words. In *ICALP'14*, pages 342–353, 2014.

20 Thomas Place and Marc Zeitoun. Separating Regular Languages with First-Order Logic. *Logical Methods in Computer Science*, 12(1), 2016.

21 Thomas Place and Marc Zeitoun. Separation for Dot-Depth Two. In *32th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS'17, 2017. `arXiv:1901.03361`.

22 Thomas Place and Marc Zeitoun. Generic results for concatenation hierarchies. *Theory of Computing Systems (ToCS)*, 2018. Selected papers from CSR'17.

23 Thomas Place and Marc Zeitoun. Separation for dot-depth two. In preparation, preprint at `http://www.labri.fr/~zeitoun/research/pdf/boolpol-full.pdf`, 2018.

24 Thomas Place and Marc Zeitoun. The Covering Problem. *Logical Methods in Computer Science*, 14(3), 2018.

25 Thomas Place and Marc Zeitoun. On all things star-free. Full version of this paper, `arXiv:1904.11863`, 2019.

26 Thomas Place and Marc Zeitoun. Separation and covering for group based concatenation hierarchies. In *LICS'19*, 2019.

27 Marcel Paul Schützenberger. On Finite Monoids Having Only Trivial Subgroups. *Information and Control*, 8(2):190–194, 1965.

28 Marcel Paul Schützenberger. Sur certaines opérations de fermeture dans les langages rationnels. *Symposia Mathematica*, XV:245–253, 1975. Convegno di Informatica Teorica, INDAM, Roma, 1973.

29 Howard Straubing. Aperiodic homomorphisms and the concatenation product of recognizable sets. *Journal of Pure and Applied Algebra*, 15(3):319–327, 1979.

30 Wolfgang Thomas. Classifying Regular Events in Symbolic Logic. *J. Comp. Sys. Sci.*, 25(3):360–376, 1982.

31 Thomas Wilke. Classifying Discrete Temporal Properties. In *Proceedings of the 16th Annual Conference on Theoretical Aspects of Computer Science*, STACS'99, pages 32–46, 1999.

From Nondeterministic to Multi-Head Deterministic Finite-State Transducers

Martin Raszyk
Department of Computer Science, ETH Zürich, Universitätstrasse 6, 8092, Switzerland
martin.raszyk@inf.ethz.ch

David Basin
Department of Computer Science, ETH Zürich, Universitätstrasse 6, 8092, Switzerland
basin@inf.ethz.ch

Dmitriy Traytel
Department of Computer Science, ETH Zürich, Universitätstrasse 6, 8092, Switzerland
traytel@inf.ethz.ch

Abstract

Every nondeterministic finite-state automaton is equivalent to a deterministic finite-state automaton. This result does not extend to finite-state transducers – finite-state automata equipped with a one-way output tape. There is a strict hierarchy of functions accepted by one-way deterministic finite-state transducers (1DFTs), one-way nondeterministic finite-state transducers (1NFTs), and two-way nondeterministic finite-state transducers (2NFTs), whereas the two-way deterministic finite-state transducers (2DFTs) accept the same family of functions as their nondeterministic counterparts (2NFTs).

We define *multi-head* one-way deterministic finite-state transducers (MH-1DFTs) as a natural extension of 1DFTs. These transducers have multiple one-way reading heads that move asynchronously over the input word. Our main result is that MH-1DFTs can deterministically express any function defined by a one-way nondeterministic finite-state transducer. Of independent interest, we formulate the all-suffix regular matching problem, which is the problem of deciding for each suffix of an input word whether it belongs to a regular language. As part of our proof, we show that an MH-1DFT can solve all-suffix regular matching, which has applications, e.g., in runtime verification.

2012 ACM Subject Classification Theory of computation → Transducers

Keywords and phrases Formal languages, Nondeterminism, Multi-head automata, Finite transducers

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.127

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Funding This research is supported by the Swiss National Science Foundation grant "Big Data Monitoring" (167162).

1 Introduction

Finite-state automata (FAs) are a fundamental model of computation. In its simplest form, a finite-state automaton reads an input word once, from left to right, while updating its state deterministically. FAs accept the regular languages. It is well-known that neither allowing the reading head to move in both directions on the input word nor updating the state nondeterministically extends their expressiveness beyond the regular languages.

A generalization of the finite-state automata are *finite-state transducers* (FTs). FTs extend a finite-state automaton with an output tape and each transition also outputs a (possibly empty) sequence of symbols from an output alphabet. The language accepted by a finite-state transducer is a relation (transduction) between input and output words. A finite-state transducer is *functional* (f-FT) if the relation represents a function. Any deterministic finite-state transducer is functional (hence, we just write DFT instead of


$$\begin{array}{ll} \mathcal{L}(\text{1DFA}) = \mathcal{L}(\text{1NFA}) \subsetneq \mathcal{L}(\text{MH-1DFA}) & \mathcal{L}(\text{1DFT}) \subsetneq \mathcal{L}(f\text{-1NFT}) \subsetneq \mathcal{L}(\text{MH-1DFT}) \\ \quad\quad\quad\quad \| & \quad\quad\quad\quad\quad \not\supseteq \\ \mathcal{L}(\text{2DFA}) = \mathcal{L}(\text{2NFA}) & \mathcal{L}(\text{2DFT}) = \mathcal{L}(f\text{-2NFT}) \end{array}$$

Figure 1 The language hierarchy accepted by models of automata (left) and transducers (right).

f-DFT). For transducers, nondeterminism makes a difference: adding nondeterminism extends the expressiveness of a finite-state transducer to nonfunctional relations. One can also classify finite-state transducers as one-way or two-way (1FTs or 2FTs) depending on whether the reading head can only move forwards or in both directions on the input word.

Another generalization of finite-state automata adds multiple reading heads that move asynchronously over the input word (see, e.g., [6] for a survey). This results in an expressive computational model with problems like emptiness, finiteness, and equivalence not being semi-decidable already for two reading heads [6]. Multi-head finite-state automata induce a strict hierarchy of languages when increasing the number of reading heads [10]. The problem of simulating two-head one-way nondeterministic finite-state automata by multi-head two-way deterministic finite-state automata is equivalent to the $\mathsf{L} \stackrel{?}{=} \mathsf{NL}$ problem [9].

We combine the previous two generalizations to the notion of multi-head finite-state transducers (Section 2). We show that multi-head one-way deterministic finite-state transducers (MH-1DFTs) can simulate any functional one-way nondeterministic finite-state transducer (f-1NFT), and thereby establish inclusion between the classes of languages accepted by these models. Central to our proof is the ability of an MH-1DFT to decide for each suffix of an input word whether it belongs to a regular language (Section 3); we call this transduction *all-suffix regular matching*. Computing this transduction allows us to deterministically find an accepting computation of the nondeterministic transducer, whenever it exists (Section 4).

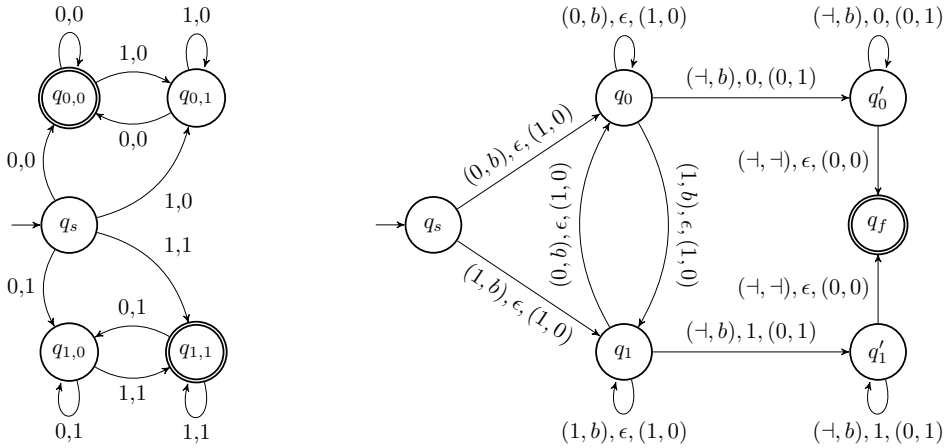
Figure 1 shows how our contributions, the transducer model and the proper inclusion of language classes, highlighted in gray, fit into the landscape of other well-studied language classes. We discuss the remaining inclusions in Section 5.

Preliminaries. Let $\mathbb{I}$ be the set of all finite intervals over the positive integers. We denote an interval $I \in \mathbb{I}$ by $[a..b] = \{x \mid a \leq x \leq b\}$. We define $[a..b) := \emptyset$, if $a \geq b$, and $[a..b) := [a..(b-1)]$, if $a < b$. Moreover, we write $[k]$ for $[1..k]$. Given a finite alphabet Σ, we denote the set of all finite words over Σ by Σ^*, the empty word by ϵ, and the length of a word $w \in \Sigma^*$ by $|w|$. Given a tuple of positions $ps \in [|w|]^{|ps|}$ of length $|ps|$, $w[ps]$ denotes $w[ps_1]w[ps_2]\dots w[ps_{|ps|}]$, i.e., the word consisting of symbols from w at the positions ps.

A *one-way deterministic finite-state automaton* (1DFA) is a tuple $A = (\Sigma, Q, q_s, Q_F, \delta)$, where Σ is the input alphabet, Q is a finite set of states, $q_s \in Q$ is the initial state, $Q_F \subseteq Q$ is the set of accepting states, and $\delta : Q \times \Sigma \to Q$ is the transition function. We extend the function δ to $\delta^* : Q \times \Sigma^* \to Q$ in the natural way. A *one-way nondeterministic finite-state automaton* (1NFA) is obtained by replacing the transition function in the definition of a 1DFA by a transition *relation* $\delta \subseteq Q \times \Sigma \times Q$.

A *one-way nondeterministic finite-state transducer* (1NFT) is derived from an underlying 1NFA by extending its transition relation with output words over an output alphabet Γ, i.e., the transition relation becomes $\delta \subseteq Q \times \Sigma \times Q \times \Gamma^*$. We extend the relation δ to $\delta^* \subseteq Q \times \Sigma^* \times Q \times \Gamma^*$ in the natural way. A 1NFT is functional if the transduction it defines is a function.

▶ **Example 1.** Figure 2a shows a f-1NFT computing the function f that maps a non-empty binary word w to $0^{|w|}$, if w ends with a zero, and $1^{|w|}$, otherwise. In the first step, the transducer guesses the last symbol and moves to a state $q_{i,j}$, where i is the guess and j is



(a) The f-1NFT from Example 1. A transition labeled by i, j represents a transition when reading the symbol i and producing the output j.

(b) The MH-1DFT from Example 3. A transition labeled by $(s_1, s_2), o, (m_1, m_2)$ represents a transition when reading the symbols (s_1, s_2), producing the output o, and advancing the reading heads by the offsets (m_1, m_2). Here, $b \in \{0, 1\}$.

Figure 2 The transducers from Examples 1 and 3.

the current (i.e., first) symbol. In the subsequent steps, the transducer updates the current state based on the current symbol. In each transition, the transducer outputs its original guess. Finally, the transducer accepts if the last symbol equals its guess.

2 Multi-Head One-Way Deterministic Finite-State Transducer

We define multi-head one-way deterministic finite-state transducers (MH-1DFTs) by adapting the definition of multi-head two-way finite-state automata [6] to multi-head one-way deterministic finite-state automata and extending the transition function with output symbols.

The following definition of an MH-1DFT formalizes a transition on a non-final state that reads a κ-tuple of symbols, enters a new state, produces some output, and advances some of the reading heads. We use a special symbol $\dashv$ to mark the end of the input tape, on the right-hand side. We further impose two conditions on the transition function. First, no reading head moves out of the input tape. Second, some reading head advances at each transition except when the new state is accepting. Without loss of generality, the transition relation δ forbids ϵ-transitions to non-final states. Hence, an MH-1DFT can only reject an input word by permitting no further transition. If δ is total, then no input word is rejected and the transduction is a total function.

▶ **Definition 2.** *A multi-head one-way deterministic finite-state transducer (MH-1DFT) is a tuple $A = (\Sigma, \dashv, \Gamma, \kappa, Q, q_s, Q_F, \delta)$, where Σ is an input alphabet, $\dashv \notin \Sigma$ is the right endmarker, Γ is an output alphabet, κ is the number of reading heads, Q is a finite set of states, $q_s \in Q$ is the initial state, $Q_F \subseteq Q$ is a set of accepting states, and $\delta : (Q \setminus Q_F) \times (\Sigma \cup \{\dashv\})^\kappa \to Q \times \Gamma^* \times \{0, 1\}^\kappa$ is the partial transition function such that:*

- *$\delta(q, es) = (q', \tilde{o}, ms)$ and $es_i = \dashv$, for any $i \in [\kappa]$, implies that $ms_i = 0$, and*
- *$\delta(q, es) = (q', \tilde{o}, \vec{0})$ implies that $q' \in Q_F$.*

A *configuration* of an MH-1DFT $A = (\Sigma, \dashv, \Gamma, \kappa, Q, q_s, Q_F, \delta)$ is a tuple $c = (w, o, q, ps) \in \mathcal{C}^A$, where $w \in \Sigma^*$ is the input word, $o \in \Gamma^*$ is the output word produced by A so far, $q \in Q$ is the current state, and $ps \in [|w| + 1]^\kappa$ are the positions of the κ reading heads on the input tape. A configuration is accepting if $q \in Q_F$. A *step* s^A is a relation on

$\mathcal{C}^A$ such that $((w, o, q, ps), (w', o', q', ps')) \in s^A$ if and only if $w = w'$, $o' = o \cdot \tilde{o}$, for some $\tilde{o} \in \Gamma^*$, and $\delta(q, w[ps]) = (q', \tilde{o}, ps' - ps)$. A *computation* of an MH-1DFT A on a word w is a maximal finite sequence of configurations $c_1, c_2, \ldots, c_l$ that starts with the initial configuration $c_1 = (w, \epsilon, q_s, \vec{1})$ and in which all pairs of consecutive configurations are contained in the step relation s^A. (The finiteness is given because $ps' - ps \neq \vec{0}$ except when $q' \in Q_F$ by Definition 2.) A computation is accepting if its last configuration c_l is accepting. Otherwise, it is rejecting.

We say that an MH-1DFT A accepts a language $L \subseteq \Sigma^* \times \Gamma^*$ if the computation on an input word w producing some output o satisfies $(w, o) \in L$ if and only if it is accepting. One can also view a language $L \subseteq \Sigma^* \times \Gamma^*$ accepted by an MH-1DFT A as a function $f : X \to \Gamma^*$, where X is the set of all input words w such that $(w, o) \in L$ for some (unique) output o and $f(w) = o$.

▶ **Example 3.** Figure 2b shows a two-head MH-1DFT computing the function f from Example 1. Initially, the first reading head reads the entire input word to determine the last symbol. Subsequently, the second reading head outputs the last symbol for each input symbol it reads. Once the second reading head has arrived at the right endmarker $\dashv$, the MH-1DFT accepts without advancing any reading head. The empty word ϵ is not mapped to any output as there is no transition from the initial state q_s when reading $(\dashv, \dashv)$.

3 All Suffix Regular Matching

All-suffix regular matching is the problem of deciding for each suffix of an input word whether it belongs to a regular language. More formally, for a regular language L, we define a function $t_L : \Sigma^* \to \{0, 1\}^*$ that maps a word w to a binary word $t_L(w)$ of length $|w| + 1$ such that, for any $1 \leq i \leq |w| + 1$, $t_L(w)[i] = 1$ if and only if the suffix $w[i..|w|]$ is in the language L. The last symbol in $t_L(w)$ denotes whether the empty word ϵ is in the language L.

We show that, for any regular language L, all-suffix regular matching can be solved by an MH-1DFT, i.e., there exists an MH-1DFT computing the function t_L. This construction is subsequently used in our simulation of any f-1NFT by an MH-1DFT (Section 4).

3.1 Informal Account

Let A be a 1DFA with $|Q|$ states. A naive approach to all-suffix regular matching is to run A on each suffix of the input word, i.e., starting from every position. The MH-1DFT solving this problem must output the decisions sequentially starting from the leftmost suffix (the entire word). Suppose we already know the decision for some *past* suffixes and want to compute the decision for a *current* suffix. To reuse the decisions for the past suffixes, we can first run the automaton A again on these suffixes until we reach the current position. Now we continue to run the automaton A on the past suffixes and also run A from its starting state on the current suffix. If the state of A in the run on the current suffix equals at some point the state of A in a run on a past suffix, then we know that the decision for the current suffix is the same as the decision for that past suffix and we can output it without the need to run A further.

Our MH-1DFT keeps a list of decisions and states for past suffixes such that running A on them until the current position yields different states. The length of this list is at most $|Q|$. To compute the decision for the current suffix, our MH-1DFT uses a reading head h positioned at the current position to run A from all the stored states for past suffixes as well as from the initial state for the current suffix. The reading head h stops once the state for the current suffix equals the state for a past suffix (or the end of the input word is reached).

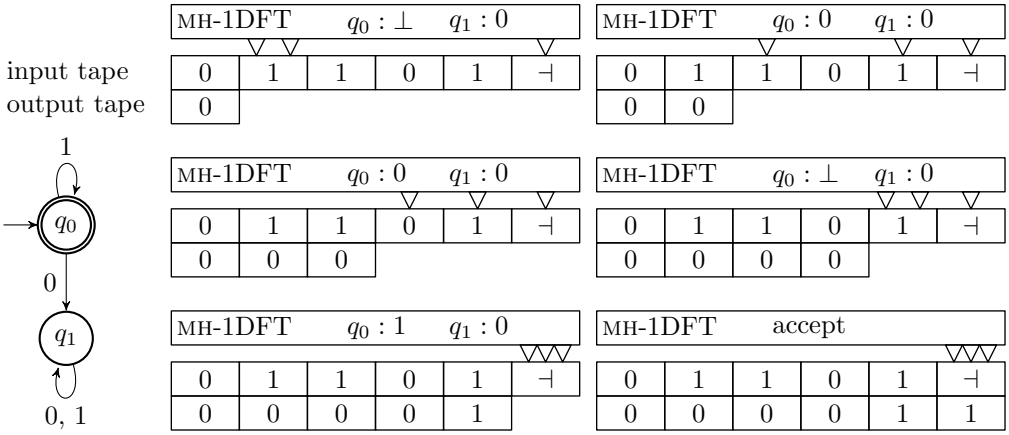


Figure 3 The automaton and configurations of the MH-1DFT from Example 4.

It remains to be shown that a finite number of reading heads suffice, independently of the input word's length. To this end, observe that whenever the reading head h moves on, the list of decisions and different states for past suffixes at that position expands (otherwise, the decision for the current suffix would have been known and h would have stopped). As the length of the list at any position is at most $|Q|$, it suffices to use $|Q| + 1$ reading heads in total. The extra reading head updates the stored states after a decision for the current suffix is computed.

▶ **Example 4.** Consider the regular language L consisting of all binary words without zero. A two-state deterministic automaton A accepting L is depicted in Figure 3. We describe a computation of our MH-1DFT with three reading heads on the input word $w = 01101$.

To compute the decision for the first suffix, one reading head reads the entire input. The remaining two reading heads advance and the decision 0 is stored (with the updated initial state $\delta(q_0, 0) = q_1$) and output (to the output tape, which is below the input tape in Figure 3).

To compute the decision for the second suffix, another reading head positioned at the second symbol advances to the fifth symbol until the two states (for the past and current suffix) become a single state q_1. The only remaining reading head advances and the decision 0 is stored (with the updated initial state $\delta(q_0, 1) = q_0$) and output.

Since the initial state q_0 is now stored with a decision, the decision for the third suffix can be immediately output, the last reading head advanced, and the stored states updated. The decision for the fourth suffix can again be immediately output, the last reading head advanced, and the stored states updated (to a single state q_1 since $\delta(q_0, 0) = \delta(q_1, 0) = q_1$). For the last suffix, the initial state q_0 is no longer stored with a decision, so a reading head reads the last symbol to compute and output the decision 1. Then the last reading head advances and the decision is stored (with an updated initial state $\delta(q_0, 1) = q_0$). Finally, as the last reading head has arrived at the right endmarker, our MH-1DFT outputs the Boolean decision 1 for the empty suffix (the initial state q_0 is accepting) and accepts.

3.2 The Multi-Head Transducer

We now formally define an MH-1DFT that solves the all-suffix regular matching problem for a regular language L. Let A be a 1DFA accepting L. Suppose A's set of states is $Q = \{q_1, q_2, \ldots, q_n\}$ and that the current suffix of an input word w starts at the position

1 $\tilde{\delta}((qbs, q), es)$:
2 $D := \{q_j \mid qbs_j \neq \bot\}; D' := \{q_{j'} \mid \exists j.\ qbs_j = (q_{j'}, \_)\}$
3 $k := |D| + 1; k' := |D'| + 1$
4 if $es_{k'} = \dashv$ then $b := q \in Q_F$
5 else if $\exists j, q_{j'}, \beta_j.\ qbs_j = (q_{j'}, \beta_j) \wedge q = q_{j'}$ then $b := \beta_j$
6 else $b := \bot$ fi
7 if $b \in \{0, 1\}$ then
8 if $es_{n+1} = \dashv$ then output b and return $\tilde{q}_f$ fi
9 $qbs' := \bot^n$
10 $\forall q_j \in D$. let $q_{j'} := \delta(q_j, es_{n+1}), (\_, \beta_j) := qbs_j$ in $qbs'_{j'} := (q_{j'}, \beta_j)$
11 let $q_{j'} := \delta(q_s, es_{n+1})$ in $qbs'_{j'} := (q_{j'}, b)$
12 output b
13 let $\tilde{k} := k + (q_s \notin D)$ in advance reading heads $\tilde{k}, \ldots, n + 1$
14 return (qbs', q_s)
15 else
16 $qbs' := qbs$
17 $\forall q_j \in D$. let $(q_{j'}, \beta_j) := qbs_j$ in $qbs'_j := (\delta(q_{j'}, es_{k'}), \beta_j)$
18 advance reading head k' fi
19 return $(qbs', \delta(q, es_{k'}))$

Figure 4 The transition function $\tilde{\delta}$ of the MH-1DFT $\tilde{A}_L$.

i. Let $\tilde{Q}_i = \{\delta^*(q_s, w[j..i)) \mid j \in [1..i)\}$ be the set of stored states for the past suffixes. For each $i' \in [i, |w| + 1]$, let $\tilde{Q}_{i,i'} = \{\delta^*(q_j, w[i..i')) \mid q_j \in \tilde{Q}_i\}$ be the states obtained by running the stored states for the past suffixes from the current position i up to the position i'. Let $\beta_{w,i}(q)$ equal one if and only if A accepts the word $w[i..|w|]$ when run from the state q.

We define our MH-1DFT $\tilde{A}_L = (\Sigma, \dashv, \Gamma, \kappa, \tilde{Q}, \tilde{q}_s, \tilde{Q}_F, \tilde{\delta})$ as follows. We set the output alphabet to $\Gamma = \{0, 1\}$ and the number of reading heads to $\kappa = |Q| + 1 = n + 1$. The set of states is $\tilde{Q} = (((Q \times \{0, 1\}) \cup \bot)^{|Q|} \times Q) \cup \{\tilde{q}_f\}$, where $\tilde{q}_f$ is a designated accepting state. With the last reading head at the position i, i.e., $ps_{n+1} = i$, and the reading head for the current suffix at the position i' (the current suffix starts at the position i), a state $(qbs, q) \in \tilde{Q}$ consists of an n-tuple qbs whose j-th component $qbs_j = (q_{j'}, \beta_{w,i}(q_j))$ stores the decision $\beta_{w,i}(q_j)$ for a past suffix corresponding to the stored state $q_j \in \tilde{Q}_i$ and the state $q_{j'}$ obtained by running A from the state q_j on $w[i..i')$. If the state q_j is not among the stored states for the current position i, then $qbs_j = \bot$. Furthermore, the state q from the state $(qbs, q) \in \tilde{Q}$ of the MH-1DFT $\tilde{A}_L$ is the state obtained by running A on the current suffix up to the position i', i.e., $q = \delta^*(q_s, w[i..i'))$. We point out that the positions i and i' themselves are not explicitly stored in the state (but $i = ps_{n+1}$ and $i' = ps_{k'}$, with k' as in Figure 4). The initial state is $\tilde{q}_s = (\bot^n, q_s)$. The set of accepting states is $\tilde{Q}_F = \{\tilde{q}_f\}$. The transition function $\tilde{\delta} : (\tilde{Q} \setminus \tilde{Q}_F) \times (\Sigma \cup \{\dashv\})^{n+1} \to \tilde{Q} \times \Gamma^* \times \{0, 1\}^{n+1}$ is defined using pseudocode in Figure 4. The transition function of the MH-1DFT from Example 4 is shown in Figure 5. We use the notation let $q_{j'} := \delta(q_s, es_{n+1})$ in $qbs'_{j'} := (q_{j'}, b)$ for *pattern-matching*, i.e., j' denotes the index of the state $\delta(q_s, es_{n+1})$ in the expression $qbs'_{j'} := (q_{j'}, b)$.

The last reading head is at the current position $i = ps_{n+1}$ and all reading heads j and j' with $j > j'$ satisfy $ps_j \leq ps_{j'}$ (the reading heads do not overtake each other). The transducer maintains the invariant that the number of reading heads beyond the position i' of the reading head for the current suffix equals $|\tilde{Q}_{i,i'}|$. The sets D and D' (Line 2) equal the set of

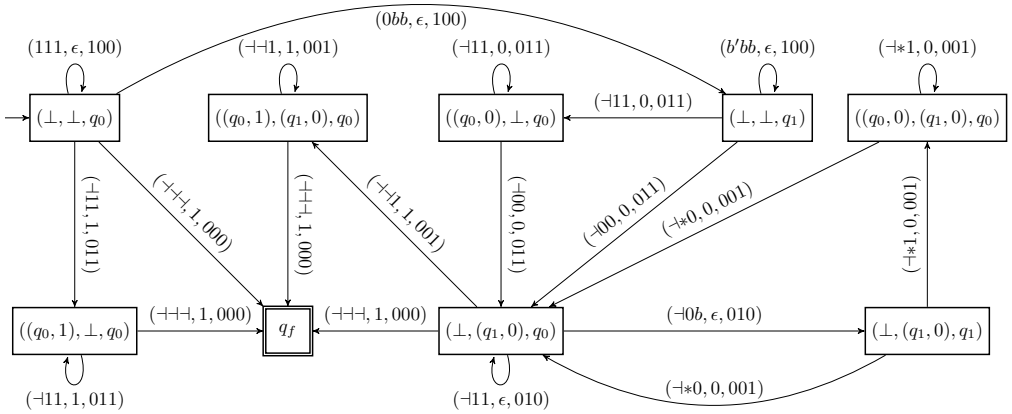


Figure 5 The transition function of the MH-1DFT from Example 4. A transition labeled by $(s_1 s_2 s_3, o, m_1 m_2 m_3)$ represents a transition when reading the symbols (s_1, s_2, s_3), producing the output o, and advancing the reading heads by the offsets (m_1, m_2, m_3). Here, $b, b' \in \{0, 1\}$ denote arbitrary symbols from the input alphabet, whereas $* \in \{0, 1, \dashv\}$ denotes an arbitrary input symbol. (Only states and transitions reachable in a computation on an input word are shown.)

stored states $\tilde{Q}_i$ and the set of states $\tilde{Q}_{i,i'}$, obtained by running A from the stored states $\tilde{Q}_i$ on $w[i..i')$. The invariant implies that $k' = |D'| + 1$ (Line 3) is the reading head for the current suffix.

The transducer $\tilde{A}_L$ first tries to determine the decision for the current suffix (Lines 4–6). If this can be determined, then $\tilde{A}_L$ updates the stored states (Lines 9–11), outputs the decision, advances all reading heads at the current position i, and sets the state for the current suffix to the initial state of A (Line 14). If A's initial state is not among the stored states $D = \tilde{Q}_i$, the reading head for the current suffix has already advanced, i.e., $\tilde{A}_L$ only needs to advance the remaining reading heads $k + 1 = \tilde{k}, \dots, n + 1$ at the current position.

If the decision for i has not been determined, then the states D' and q are updated (Lines 16–17 and 19) and the reading head for the current suffix advances (Line 18).

3.3 Proof of Correctness

To prove that the MH-1DFT $\tilde{A}_L$ computes t_L, we formulate an invariant on a configuration of $\tilde{A}_L$ that is satisfied by each configuration from a computation on an input. For the accepting state $\tilde{q}_f$, the invariant states that the output is correct: $\mathcal{I}(w, o, \tilde{q}_f, ps) \equiv (o = t_L(w))$ (C0).

For a state $(qbs, q) \in \tilde{Q}$, the invariant captures the properties of a state and the reading heads' positions mentioned previously. In addition, it states that correct output has been produced for all positions preceding the current position $i = ps_{n+1}$. In the following, we use the definitions from Lines 2–3 in Figure 4. We further define $i' = ps_{k'}$ to be the position of the reading head for the current suffix. To capture the expansion of the set of stored states by running A on the current suffix, we define $\tilde{Q}'_j = \tilde{Q}_{i,j} \cup \{\delta^*(q_s, w[i..j))\}$, for all $j \in [i..i')$, and $\tilde{Q}'_j = \tilde{Q}_{i,j}$, for all $j \in [i'..(|w| + 1)]$. Then the invariant is as follows:

$$\mathcal{I}(w, o, (qbs, q), ps) \equiv (|o| = ps_{n+1} - 1 \wedge \forall j \in [|o|].\ o_j = t_L(w)[j]) \wedge \quad \text{(C1)}$$

$$\forall j \in [n].\ \forall q_{j'}, \beta_j.\ (qbs_j = (q_{j'}, \beta_j) \implies q_{j'} = \delta^*(q_j, w[i..i')) \wedge \beta_j = \beta_{w,i}(q_j)) \wedge \quad \text{(C2)}$$

$$q = \delta^*(q_s, w[i..i')) \wedge \quad \text{(C3)}$$

$$D = \tilde{Q}_i \wedge \quad \text{(C4)}$$

$$D' = \tilde{Q}_{i,i'} \wedge \tag{C5}$$
$$\forall j \in [i..i'). \, \tilde{Q}_{i,j} \subsetneq \tilde{Q}'_j \wedge \tag{C6}$$
$$(ps_{n+1} \leq \cdots \leq ps_1) \wedge \tag{C7}$$
$$\forall j \in [i..|w|]. \, |\{h \mid ps_h > j\}| = |\tilde{Q}'_j|. \tag{C8}$$

▶ **Lemma 5.** *For any input word w, the initial configuration of $\tilde{A}_L$ satisfies the invariant, i.e., $\mathcal{I}(w, \epsilon, (\bot^n, q_s), 1^{n+1})$ holds.*

Proof. Observe that $i = i' = 1$ and $D = D' = \tilde{Q}_i = \tilde{Q}_{i,j} = \emptyset$, for all $j \in [i..|w|]$. ◀

▶ **Lemma 6.** *Let $c_1 = (w, o, (qbs, q), ps)$ and $c_2 = (w, o', q', ps')$ be two configurations of $\tilde{A}_L$ such that $\mathcal{I}(c_1)$ holds and $(c_1, c_2) \in s^{\tilde{A}_L}$. Then $\mathcal{I}(c_2)$ holds.*

Proof. We refer to Line l in Figure 4 as Ll. Furthermore, we refer to the i-th conjunct in $\mathcal{I}(c_1)$ and $\mathcal{I}(c_2)$ as Ci and Ci', respectively and to the i-th fact labeled in the proof as Fi. Let us denote by i_j, i'_j, D_j, D'_j, k_j, k'_j, ${}^j\tilde{Q}'_{j'}$, ${}^j ps$, $j \in \{1, 2\}$, the respective definitions for the configuration c_j. To derive that $\mathcal{I}(c_2)$ holds, we analyze the transition function $\tilde{\delta}$ in Figure 4.

First we show that $b \neq \bot \implies b = t_L(w)[i_1]$ (F1). If $es_{k'_1} = \dashv$, then $i'_1 = |w| + 1$, C3, and L4 imply that $b = t_L(w)[i_1]$. If L5's condition holds, then C2 and C3 imply that $b = t_L(w)[i_1]$.

Consider the case $b \in \{0, 1\}$ (L8–14). If $es_{n+1} = \dashv$, then $i_1 = i'_1 = |w| + 1$ and $b \neq \bot$ (due to $i'_1 = |w| + 1$ and L4) which with F1 implies $b = t_L(w)[i_1]$. C1 and $b = t_L(w)[i_1]$ imply C0, i.e., $\mathcal{I}(c_2)$ holds since c_2 is accepting and thus $\mathcal{I}(c_2) \equiv$ C0. Otherwise (if $es_{n+1} \neq \dashv$), we have $i_1 \leq |w|$. Because $\tilde{k} = |D_1| + 1 + (q_s \notin D_1) \leq n + 1$, the last reading head advances, i.e., $i_2 = i_1 + 1$ (F2). Now, F1, F2, and C1 imply C1'. Next we show that $|\{h \mid {}^1ps_h > i_1\}| = \tilde{k} - 1$ (F3). From C4, we have $D_1 = \tilde{Q}_{i_1}$. It follows that $|{}^1\tilde{Q}'_{i_1}| = |D_1| + (q_s \notin D_1)$. C8 then implies $|\{h \mid {}^1ps_h > i_1\}| = \tilde{k} - 1$, i.e., that precisely those readings heads at the position i_1 advanced (L13). In particular, this implies C7'. L9–11 and C4 imply $D_2 = D'_2 = \tilde{Q}_{i_1+1} \overset{\text{F2}}{=} \tilde{Q}_{i_2}$ (F4). This immediately implies C4'. Next we show that $i_2 = i'_2$ (F5). If $i_2 = |w| + 1$, then $i_2 = i'_2$ follows from C7'. Suppose that $i_2 \leq |w|$. It follows that $|\{h \mid {}^2ps_h > i_2\}| \overset{\text{F2–3,L13}}{=} |\{h \mid {}^1ps_h > i_1 + 1| \overset{\text{C8}}{=} |{}^1\tilde{Q}'_{i_1+1}|$. If $i'_1 > i_1 + 1$, we derive $|{}^1\tilde{Q}'_{i_1+1}| \overset{i'_1 > i_1+1}{=} |\tilde{Q}_{i_1+1}|$. If $i'_1 \leq i_1 + 1$, we derive $|{}^1\tilde{Q}'_{i_1+1}| \overset{\text{L5}}{=} |\tilde{Q}_{i_1+1}|$. Hence, $|\{h \mid {}^2ps_h > i_2\}| = |\tilde{Q}_{i_1+1}| \overset{\text{F2}}{=} |\tilde{Q}_{i_2}| \overset{\text{F4}}{=} D'_2$, which implies that $i_2 = i'_2$. F5 implies C6' and F5 together with L14 imply C3'. F4–5 imply C5'. L9–11, C2, F1, and F5 further imply C2'. F3 and L13 imply that those reading heads at i_1 advanced. Hence for C8', it suffices to show $|{}^2\tilde{Q}'_j| = |{}^1\tilde{Q}'_j|$, for all $j \in [(i_1 + 1)..|w|]$. We derive $|{}^2\tilde{Q}'_j| \overset{\text{F2,F5}}{=} |\tilde{Q}_{i_1+1,j}|$. If $i'_1 \leq j$, then $|\tilde{Q}_{i_1+1,j}| \overset{\text{L5},i'_1 \leq j}{=} |\tilde{Q}_{i_1,j}| \overset{i'_1 \leq j}{=} |{}^1\tilde{Q}'_j|$. If $i'_1 > j$, then $|\tilde{Q}_{i_1+1,j}| \overset{i'_1 > j}{=} |{}^1\tilde{Q}'_j|$. This completes the proof of C8'. Thus, $\mathcal{I}(c_2)$ holds in the case $b \in \{0, 1\}$.

We continue with the case $b = \bot$ (L17–19). Together with L5, $b = \bot$ implies $|D'_1| < n$. We obtain $k'_1 \leq n$ and that $i_2 = {}^2ps_{n+1} = {}^1ps_{n+1} = i_1$ (F6). Moreover, L17 implies $D_2 \overset{\text{L17}}{=} D_1 \overset{\text{C4}}{=} \tilde{Q}_{i_1} \overset{\text{F6}}{=} \tilde{Q}_{i_2}$ which yields C4'. Since no output is produced, F6 immediately yields C1'. Together with L4, $b = \bot$ implies $i'_1 \leq |w|$. Next we show that $i'_2 = i'_1 + 1$ (F7). We derive $|\{h \mid {}^1ps_h > i'_1\}| \overset{\text{C8},i'_1 \leq |w|}{=} |{}^1\tilde{Q}'_{i'_1}| \overset{\text{C5}}{=} |D'_1| \overset{\text{L3}}{=} k'_1 - 1$ (F8). The fact F8 implies that the single advancing reading head k'_1 is the first reading head at the position i'_1, which further implies C7'. If $i'_1 + 1 = |w| + 1$, then $i'_2 = i'_1 + 1$ follows from $|D'_2| \overset{\text{L17}}{\leq} |D'_1|$. If $i'_1 + 1 \leq |w|$, then $i'_2 = i'_1 + 1$ follows from $|D'_2| \overset{\text{L17}}{=} |{}^1\tilde{Q}'_{i'_1+1}|$ and $|\{h \mid {}^1ps_h > i'_1 + 1\}| \overset{\text{C8},i'_1+1 \leq |w|}{=} |{}^1\tilde{Q}'_{i'_1+1}|$. Then, L17, the facts F6–7 and C2, C4–5 imply C2' and C5'. Furthermore, L19, F6–7 and C3 imply C3'. Together with L5 and F6–7, $b = \bot$ implies that $\tilde{Q}_{i_1,i'_1} \subsetneq {}^2\tilde{Q}'_{i'_1}$ (F9).

F6–7 and F9 together with C6 yield C6'. It suffices to show C8' for $j = i'_1$; otherwise ${}^1\tilde{Q}'_j = {}^2\tilde{Q}'_j$ and the number of heads at positions $> j$ did not change. For $j = i'_1$, we derive $|\{h \mid {}^2ps_h > i'_1\}| \overset{\text{L18}}{=} |\{h \mid {}^1ps_h > i'_1\}| + 1 \overset{\text{C8}}{=} |{}^1\tilde{Q}'_{i'_1}| + 1 \overset{b=\bot,\text{L5}}{=} |{}^2\tilde{Q}'_{i'_1}|$. This completes the proof of C8'. Thus, $\mathcal{I}(c_2)$ holds in the case $b = \bot$. ◀

▶ **Theorem 7.** *For any input word w, the output produced by $\tilde{A}_L$ is $t_L(w)$.*

Proof. Let $c_1, c_2, \ldots, c_l$ be $\tilde{A}_L$'s computation on w. Lemmas 5 and 6 imply that $\mathcal{I}(c_i)$ holds for all configurations c_i, $i \in [l]$. Since $\tilde{A}_L$'s transition function $\tilde{\delta}$ is total (the transducer cannot get stuck), Definition 2 implies that c_l is an accepting configuration. The invariant $\mathcal{I}(w, o, \tilde{q}_f, ps)$ for the last configuration $c_l = (w, o, \tilde{q}_f, ps)$ then implies that $o = t_L(w)$. ◀

4 Simulation

Let A be a f-1NFT and let $L \subseteq \Sigma^* \times \Gamma^*$ be the language accepted by A. We show that there exists an MH-1DFT accepting the same language L. This establishes inclusion between the classes of languages accepted by these models. Because two-head finite-state automata strictly extend the expressiveness of one-head finite-state automata [10], the inclusion is proper.

4.1 Informal Account

To simulate the f-1NFT A with $|Q|$ states on an input w, we first check if w is accepted by A's underlying automaton. This can be done using a single head that reads the entire input. If the automaton rejects, then clearly no $(w, o) \in L$ and our MH-1DFT simulating A may also immediately reject. Otherwise there is exactly one $(w, o) \in L$, since A is functional. Hence, it suffices to follow an accepting computation of the underlying automaton and concatenate the outputs from A's transition relation to produce the output o. A problem with this straightforward approach is the nondeterminism of A that may have multiple transitions from a given state of A on the same symbol. Nonetheless, if we are able to determine a transition to a state from which A's underlying automaton accepts the rest of the input word, then we can follow it, since this transition is part of an accepting computation. So now our problem is reduced to checking from which states a 1NFA accepts a suffix of an input word. But since the language accepted by a 1NFT run from a particular state is regular, this is precisely an instance of all-suffix regular matching, for which we have already constructed an MH-1DFT in the previous section.

Our MH-1DFT simulating A follows the described approach. It tries to find an accepting computation of the underlying automaton of A starting from its initial state. To this end, our MH-1DFT runs $|Q|$ instances of the (distinct) MH-1DFTs for the regular languages accepted by the underlying finite-state automaton of A when run from one of its states. If at some point, no transition can be made from the current state to a new state from which the remainder of the input word is accepted, then our MH-1DFT rejects the input word. Otherwise, it follows one such transition (an arbitrary one if there are multiple transitions) and outputs the output word from the transition of the transducer A. Upon reaching the right endmarker of the input word, our MH-1DFT accepts if the current simulated state is accepting with respect to A.

▶ **Example 8.** We revisit the f-1NFT A from Example 1. In Example 3, we proposed an ad-hoc MH-1DFT that simulates A. Here, we show how to obtain an MH-1DFT that simulates A by following the general approach. First we construct a 1DFA for each regular language accepted by A's underlying automaton when run from one of its states. For instance, the

regular language L_s for the initial state of A contains all non-empty binary words (note that this is the set of all input words w such that $(w, o) \in L$ for some o) and can be accepted by a simple two-state 1DFA. For the states $q_{0,0}$ and $q_{0,1}$, the regular languages $L_{0,0}$ and $L_{0,1}$ contain all non-empty binary words that end with a zero. Analogously, for the states $q_{1,0}$ and $q_{1,1}$, the regular languages $L_{1,0}$ and $L_{1,1}$ contain all non-empty binary words that end with a one. Each of these four regular languages $L_{0,0}$, $L_{0,1}$, $L_{1,0}$, and $L_{1,1}$ can be accepted by a simple three-state 1DFA.

Now, for each of the five regular languages (L_s, $L_{0,0}$, $L_{0,1}$, $L_{1,0}$, and $L_{1,1}$), we construct an MH-1DFT solving all-suffix-pattern matching. The MH-1DFT for L_s has three reading heads and the MH-1DFTs for $L_{0,0}$, $L_{0,1}$, $L_{1,0}$, and $L_{1,1}$ have four reading heads each. Hence, the resulting MH-1DFT $\tilde{A}$ simulating A has $1 + 3 + 4 \cdot 4 = 20$ reading heads.

Let us analyze $\tilde{A}$'s computation on the input word $w = 01101$. The MH-1DFT for L_s computes that the entire input word is accepted by the underlying automaton of A (because w is non-empty). Hence, there exists an accepting computation of A on w that $\tilde{A}$ will simulate. The MH-1DFTs for $L_{0,0}$ and $L_{0,1}$ compute that neither of the first two suffixes is in either of these two regular languages (because w does not end with a zero). In contrast, the MH-1DFTs for $L_{1,0}$ and $L_{1,1}$ compute that both the first two suffixes are in both these regular languages (because w ends with a one). Now, the MH-1DFT $\tilde{A}$ simulating A must decide between the states $q_{1,0}$ and $q_{1,1}$ based on the actual transition relation of A. Since the first symbol of w is a zero, the next simulated state of A is going to be $q_{1,0}$ and the first output symbol is 1 (which is the correct guess of A about the last symbol of the input word). The rest of A's computation on w is in fact deterministic and we omit it.

4.2 The Multi-Head Transducer

We define an MH-1DFT simulating a f-1NFT $A = (\Sigma, \Gamma, Q, q_s, Q_F, \delta)$. Suppose the set of states of A is $Q = \{q_1, q_2, \ldots, q_n\}$. For each $i \in [n]$, let L_i be the regular language accepted by the 1NFA $A_i = (\Sigma, Q, q_i, Q_F, \delta')$, where $\delta' \subseteq Q \times \Sigma \times Q$ is the transition relation obtained from δ by ignoring the output word Γ^*. In particular, note that A_s is the underlying automaton of A and A_i is obtained from A_s by changing the initial state to q_i.

By Theorem 7, there exists an MH-1DFT $\tilde{A}_{L_i} = (\Sigma, \dashv, \{0,1\}, \kappa_i, \tilde{Q}^i, \tilde{q}^i_s, \tilde{Q}^i_F, \tilde{\delta}_i)$ computing the function t_{L_i}, for each $i \in [n]$. We define our MH-1DFT $\tilde{A} = (\Sigma, \dashv, \Gamma, \kappa, \tilde{Q}, \tilde{q}_s, \tilde{Q}_F, \tilde{\delta})$ simulating A as follows. We set the number of reading heads to $\kappa = 1 + \sum_{k=1}^{n} \kappa_i$ (the extra reading head is needed to simulate the actual step of A using its transition relation δ, in particular, to obtain the output word $\tilde{o} \in \Gamma^*$). The set of states is $\tilde{Q} = (Q \times (\tilde{Q}^1 \times \{\epsilon, 0, 1\}^2) \times \cdots \times (\tilde{Q}^n \times \{\epsilon, 0, 1\}^2)) \cup \{\tilde{q}_f\}$, where $\tilde{q}_f$ is a designated accepting state. The first component of a state $\tilde{q} \in \tilde{Q}$ stores the current simulated state $q = q_{i_j} \in Q$ of A. The $(i+1)$-th component of a state $\tilde{q} \in \tilde{Q}$ stores a tuple $(\tilde{q}^i, t_{i,j}, t_{i,j+1})$, where $\tilde{q}^i \in \tilde{Q}_i$ is a state of the MH-1DFT $\tilde{A}_{L_i}$, and $t_{i,j}$ and $t_{i,j+1}$ are the decision for the current and next suffix (or ϵ if they have not been computed yet). The initial state is $\tilde{q}_s = (q_s, (\tilde{q}^1_s, \epsilon, \epsilon), \ldots, (\tilde{q}^n_s, \epsilon, \epsilon))$. The set of accepting states is $\tilde{Q}_F = \{\tilde{q}_f\}$. The transition function $\tilde{\delta} : (\tilde{Q} \setminus \tilde{Q}_F) \times (\Sigma \cup \{\dashv\})^\kappa \to \tilde{Q} \times \Gamma^* \times \{0,1\}^\kappa$ is defined using pseudocode in Figure 6. We point out that the next simulated state $q_{j'}$ of A (Line 14 in Figure 6) needs not be unique and the MH-1DFT $\tilde{A}$ chooses (deterministically) an arbitrary state if it is not unique.

Recall that the only way for an MH-1DFT to reject an input is by getting stuck. In Figure 6, this is represented by the command "reject" (Lines 4 and 19). If the control flow reaches "reject", then the transition function is not defined for the respective input arguments.

```
1  δ̃((q, (q̃^1, t_{1,1}, t_{1,2}), ..., (q̃^n, t_{n,1}, t_{n,2})), es) :
2    if es_1 = ⊣ then
3      if q ∈ Q_F then return q̃_f
4      else reject fi fi
5    let i be the smallest index such that (t_{i,1}, t_{i,2}) ∉ {0,1}^2
6    let es^i be the symbols belonging to the reading heads of Ã_{L_i}
7    (q̃^i, t, ms^i) := δ̃_i(q̃^i, es^i)
8    advance reading heads belonging to Ã_{L_i} according to the offsets ms^i
9    if t ∈ {0,1} then
10     if t_{i,1} = ϵ then t_{i,1} := t
11     else t_{i,2} := t fi fi
12   if ∀i ∈ [n]. (t_{i,1}, t_{i,2}) ∈ {0,1}^2 then
13     let q_j := q
14     if ∃j' ∈ [n]. t_{j,1} = 1 ∧ t_{j',2} = 1 ∧ (q_j, es_1, q_{j'}) ∈ δ' then
15       let õ be the output of a transition δ(q_j, es_1, q_{j'}, õ, 1) in output õ
16       advance reading head 1
17       return (q_{j'}, (q̃^1, t_{1,2}, ϵ), ..., (q̃^n, t_{n,2}, ϵ))
18     else
19       reject fi fi
20   return (q, (q̃^1, t_{1,1}, t_{1,2}), ..., (q̃^n, t_{n,1}, t_{n,2}))
```

Figure 6 The transition function $\tilde{\delta}$ of the MH-1DFT $\tilde{A}$.

The first reading head simulates the only reading head of the f-1NFT A. If it reads the right endmarker $\dashv$, then the computation is accepted if and only if the current simulated state q is accepting (Lines 2–4). Otherwise, the MH-1DFT $\tilde{A}$ performs a transition of an MH-1DFT $\tilde{A}_{L_i}$ for which either $t_{i,1}$ or $t_{i,2}$ have not been computed yet (Lines 5–8). The MH-1DFT $\tilde{A}$ maintains the invariant that one such unknown decision exists. Then it updates $t_{i,1}$, $t_{i,2}$ accordingly (Lines 9–11). The definition of the transition function $\tilde{\delta}_i$ (Figure 4) implies that at most a single Boolean decision is produced in each transition.

Once all the decisions $t_{i,1}$ and $t_{i,2}$ have been computed, a step of the f-1NFT A can be simulated (Lines 13–19). If there exists a transition from the current state $q = q_j$ to a new state $q_{j'}$ from which the next suffix is accepted by A, then it is taken (Lines 15–17). The decisions for the next suffix become the decisions for the current suffix and the decisions for the next suffix become unknown (Line 17). Otherwise, the input word is rejected (Line 19).

4.3 Proof of Correctness

To prove that the MH-1DFT $\tilde{A}$ simulates the f-1NFT A, we formulate an invariant on a configuration of $\tilde{A}$ that is satisfied by each configuration from a computation on an input word. For the accepting state $\tilde{q}_f$, the invariant merely states that the input word is accepted by A and the output is correct: $\mathcal{J}(w, o, \tilde{q}_f, ps) \equiv (w, o) \in L$ (D0).

For a state $(q, (\tilde{q}^1, t_{1,1}, t_{1,2}), \ldots, (\tilde{q}^n, t_{n,1}, t_{n,2})) \in \tilde{Q}$, the invariant captures the properties of a state mentioned previously. To express them, it uses the invariant $\mathcal{I}$ on the configurations of the MH-1DFT $\tilde{A}_{L_i}$. In addition, it guarantess the existence of the index i from Line 5 in Figure 6 and that the simulation does not get stuck after it successfully performs its first

step. We denote the positions of the reading heads belonging to $\tilde{A}_{L_i}$, $i \in [n]$, by ps^i.

$$\mathcal{J}(w, o, (q, (\tilde{q}^1, t_{1,1}, t_{1,2}), \ldots, (\tilde{q}^n, t_{n,1}, t_{n,2})), ps) \equiv (q_s, w[1..ps_1), q, o) \in \delta^* \wedge \tag{D1}$$

$$\forall i \in [n].\ \exists ts.\ (|ts| = ps_1 - 1 \wedge \mathcal{I}(w, ts \cdot t_{i_1} \cdot t_{i,2}, \tilde{q}^i, ps^i)) \wedge \tag{D2}$$

$$\exists i \in [n].\ (t_{i,1}, t_{i,2}) \notin \{0,1\}^2 \wedge \tag{D3}$$

$$(ps_1 > 1 \Longrightarrow \exists q_f \in Q_F.\ \exists o'.\ (q, w[ps_1..|w|], q_f, o') \in \delta^*) \tag{D4}$$

▶ **Lemma 9.** *For any input word w, the initial configuration of $\tilde{A}$ satisfies the invariant, i.e., $\mathcal{I}(w, \epsilon, (q_s, (\tilde{q}_s^1, \epsilon, \epsilon), \ldots, (\tilde{q}_s^n, \epsilon, \epsilon)), 1^\kappa)$ holds.*

Proof. The invariant follows directly from Lemma 5. ◀

▶ **Lemma 10.** *Let $c_1 = (w, o, (q, (\tilde{q}^1, t_{1,1}, t_{1,2}), \ldots, (\tilde{q}^n, t_{n,1}, t_{n,2})), ps)$ and $c_2 = (w, o', q', ps')$ be two configurations of $\tilde{A}$ such that $\mathcal{J}(c_1)$ holds and $(c_1, c_2) \in s^{\tilde{A}}$. Then $\mathcal{J}(c_2)$ holds.*

Proof. We refer to Line l in Figure 6 as Ll (e.g., L7 denotes Line 7). Furthermore, we refer to the i-th conjunct from $\mathcal{J}(c_1)$ and $\mathcal{J}(c_2)$ as Di and Di', respectively and to the i-th fact labeled in the proof as Fi. To derive that $\mathcal{J}(c_2)$ holds, we analyze the transition function $\tilde{\delta}$ in Figure 6.

If $es_1 = \dashv$, then $ps_1 = |w| + 1$ and D1 implies $(q_s, w[1..|w|], q, o) \in \delta^*$ (F1). The fact that a step from c_1 to c_2 was taken implies that $q \in Q_F$ (otherwise, the transition from c_1 would be undefined, due to L3–4). The fact F1 together with $q \in Q_F$ imply D0, i.e., $\mathcal{J}(c_2)$ holds as c_2 is accepting and thus $\mathcal{J}(c_2) \equiv (w, o) \in L \equiv$ D0.

Suppose that $es_1 \neq \dashv$. Conjunct D3 implies that the index from L5 is well-defined. Lines L5–11, Conjunct D2, and Lemma 6 imply D2' after L11. Furthermore, L16–17 imply D2' also if the branch L15–17 is reached. If the branch L13–19 is not reached, then D1, D4 immediately imply D1', D4', and L12 implies D3'. Otherwise, the branch L15–17 must be reached (otherwise, the transition from c_1 would be undefined, due to L19). Then Conjunct D1 and Lines L14–17 imply D1'. Furthermore, Lines L12 and L17 directly imply D3' (note that reaching L19 would make the transition from c_1 to c_2 undefined, which is a contradiction). Finally, Lines L13–14 and conjuncts D1–2 together with the definition of the invariant $\mathcal{I}$ imply D4'. ◀

▶ **Theorem 11.** *For any f-1NFT A, there exists an* equivalent MH-*1DFT $\tilde{A}$, i.e., both A and $\tilde{A}$ accept the same language.*

Proof. We again refer to Line l in Figure 6 as Ll. Let $L_{\tilde{A}}$ denote the language accepted by $\tilde{A}$. Let $c_1, c_2, \ldots, c_l$ be the computation of $\tilde{A}$ on an input word w. We refer to the i-th conjunct from $\mathcal{J}(c_l)$ as Di. Let o be the output of the computation of $\tilde{A}$ on w. Lemmas 9 and 10 imply that $\mathcal{J}(c_i)$ holds for all configurations c_i, $i \in [1..l]$.

If the computation is accepting (i.e., $(w, o) \in L_{\tilde{A}}$), then $\mathcal{J}(c_l)$ implies that $(w, o) \in L$. Moreover, since A is functional and $\tilde{A}$ deterministic, there exists no $o' \neq o$ such that $(w, o') \in L$ or $(w, o') \in L_{\tilde{A}}$. If the computation is rejecting, then the transition from c_l is undefined due to L4 or L19. If L4 is reached, then $ps_1 = 1$ (otherwise, D4 would imply $q \in Q_F$, which is a contradiction). With D1 and L2, this implies that $q = q_s$ and $w = \epsilon$. The facts that $q = q_s$, $w = \epsilon$, and $q \notin Q_F$ (due to L3–4) imply that the input word w is rejected by A (i.e., no $(w, o) \in L$). Moreover, no $(w, o) \in L_{\tilde{A}}$, since the deterministic computation of $\tilde{A}$ on w is rejecting.

If L19 is reached, then again $ps_1 = 1$ (otherwise, the branch L15–17 would have been taken by D4 and D2). Then L14 and D2 imply that the input word is rejected by A (i.e., no $(w, o) \in L$). Moreover, no $(w, o) \in L_{\tilde{A}}$, since the deterministic computation of $\tilde{A}$ on w is rejecting. ◀

5 Discussion and Related Work

We review results on the expressiveness of transducer models (as depicted in Figure 1) and connections to a practical application.

Expressiveness of Related Formalisms. It is well-known that neither nondeterminism nor a two-way reading head extends the expresiveness of a one-head finite-state automaton beyond the regular languages [7]. Formally, $\mathcal{L}(\text{1DFA}) = \mathcal{L}(\text{1NFA}) = \mathcal{L}(\text{2DFA}) = \mathcal{L}(\text{2NFA})$. But adding reading heads does make a difference: in fact, there is a strict hierarchy of languages accepted by finite-state automata when increasing the number of reading heads [10]. Formally, $\mathcal{L}(\text{2NFA}) \subsetneq \mathcal{L}(\text{MH-1DFA})$. By viewing a finite-state automaton as a functional finite-state transducer that does not produce any output, this further implies that $\mathcal{L}(\text{MH-1DFT}) \not\subseteq \mathcal{L}(f\text{-2NFT})$. Theorem 11 implies that $\mathcal{L}(f\text{-1NFT}) \subseteq \mathcal{L}(\text{MH-1DFT})$. This yields the proper inclusion $\mathcal{L}(f\text{-1NFT}) \subsetneq \mathcal{L}(\text{MH-1DFT})$.

Let us consider the function f that maps a non-empty binary word w to $0^{|w|}$, if w ends with a zero, and $1^{|w|}$, otherwise. The function f can be computed by the f-1NFT from Example 1. Nevertheless, f cannot be computed by a 1DFT. Intuitively, a 1DFT cannot start producing any output before seeing the last symbol of the input word; but it cannot remember the input word's length needed to produce the output. We conclude that the expressiveness of f-1NFTs strictly extends that of 1DFTs, i.e., $\mathcal{L}(\text{1DFT}) \subsetneq \mathcal{L}(f\text{-1NFT})$.

Now consider the function $w \mapsto w^R$ that maps a binary word to its reverse. It can be computed by a f-2NFT that first moves its reading head to the end of the input word and then reads the word backwards while outputting its symbols in the reversed order (in fact, this transducer behaves deterministically). Nevertheless, $w \mapsto w^R$ cannot be computed by a f-1NFT [5]. We conclude that the expressiveness of f-2NFTs strictly extends the expressiveness of f-1NFTs, i.e., $\mathcal{L}(f\text{-1NFT}) \subsetneq \mathcal{L}(f\text{-2NFT})$. Surprisingly, adding nondeterminism to functional two-way finite-state transducers does not extend their expressiveness [4], i.e., $\mathcal{L}(\text{2DFT}) = \mathcal{L}(f\text{-2NFT})$. We further conjecture that the function $w \mapsto w^R$ cannot be computed by a MH-1DFT either, and thus the languages accepted by MH-1DFT and 2DFT are incomparable.

We also conjecture that MH-1DFTs are closed under composition, i.e., whenever $f : X \to Y$ and $g : Y \to Z$ are computed by some MH-1DFTs, then there exists an MH-1DFT computing $g \circ f : X \to Z$. Such a composition result could give rise to a more modular construction of MH-1DFTs from f-1NFTs, which would recast our MH-1DFT from Figure 6 as a composition.

Monitoring. The use of MH-1DFT and all-suffix regular matching has applications to *monitoring* (also called *runtime verification*), which is the problem of checking the compliance of an event stream, at each position in the stream, to a policy formalized in a specification language. Our recent work [8] provides evidence that exploiting multiple one-way reading heads significantly improves the efficiency of monitoring policies formalized in metric temporal logic (MTL). Since MTL is less expressive than (timed) regular expressions [3], the monitor from [8] does not implement the proposed solution to all-suffix regular matching. Moreover, its underlying transducer's space complexity depends logarithmically on the input length.

6 Conclusion

We proposed multi-head finite-state transducers as a combination of multi-head finite-state automata and finite-state transducers. We showed that multiple one-way reading heads can replace nondeterminism on functions computable by finite-state transducers. The key insight is that multiple one-way reading heads suffice to solve the all-suffix regular matching problem.

As future work, we plan to use the MH-1DFT construction for all-suffix regular matching to implement an efficient monitor for timed regular expressions [1], which are strictly more expressive than metric temporal logic supported by our multi-head monitor [8]. The problem of all-suffix regular matching has already inspired another monitor of ours [2]. In that monitor, as soon as two past suffixes yield the same state of the automaton, the equivalence between them is output (namely, the monitor outputs that the positions associated with the suffixes will have the same verdict); outputting the actual Boolean value for the equivalent positions is postponed, potentially until the input's end. We envision designing an MH-1DFT that outputs an explicit Boolean value for each position in the input word in the order of their appearance.

References

1 Eugene Asarin, Paul Caspi, and Oded Maler. Timed regular expressions. *J. ACM*, 49(2):172–206, 2002.

2 David Basin, Bhargav Bhatt, Srđan Krstić, and Dmitriy Traytel. Almost Event-Rate Independent Monitoring. *Form. Meth. Sys. Des.*, 2019 (published online February 2019).

3 Patricia Bouyer, Fabrice Chevalier, and Nicolas Markey. On the expressiveness of TPTL and MTL. *Inf. Comput.*, 208(2):97–116, 2010.

4 Joost Engelfriet and Hendrik Jan Hoogeboom. Two-Way Finite State Transducers and Monadic Second-Order Logic. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *ICALP 1999*, volume 1644 of *LNCS*, pages 311–320. Springer, 1999.

5 Emmanuel Filiot, Olivier Gauwin, Pierre-Alain Reynier, and Frédéric Servais. From Two-Way to One-Way Finite State Transducers. In *LICS 2013*, pages 468–477. IEEE Computer Society, 2013.

6 Markus Holzer, Martin Kutrib, and Andreas Malcher. Complexity of multi-head finite automata: Origins and directions. *Theor. Comput. Sci.*, 412(1-2):83–96, 2011.

7 Michael O. Rabin and Dana S. Scott. Finite Automata and Their Decision Problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959.

8 Martin Raszyk, David Basin, Srđan Krstić, and Dmitriy Traytel. Multi-head monitoring of metric temporal logic. In Yu-Fang Chen, Chih-Hong Cheng, and Javier Esparza, editors, *ATVA 2019*, LNCS. Springer, 2019. To appear. `http://people.inf.ethz.ch/trayteld/papers/atva19-hydra/hydra.pdf`.

9 Ivan Hal Sudborough. On Tape-Bounded Complexity Classes and Multihead Finite Automata. *J. Comput. Syst. Sci.*, 10(1):62–76, 1975.

10 Andrew Chi-Chih Yao and Ronald L. Rivest. k+1 Heads Are Better than k. *J. ACM*, 25(2):337–340, 1978.

Sequentiality of String-to-Context Transducers

Pierre-Alain Reynier
Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France
pierre-alain.reynier@lis-lab.fr

Didier Villevalois
Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France
didier.villevalois@lis-lab.fr

Abstract

Transducers extend finite state automata with outputs, and describe transformations from strings to strings. Sequential transducers, which have a deterministic behaviour regarding their input, are of particular interest. However, unlike finite-state automata, not every transducer can be made sequential. The seminal work of Choffrut allows to characterise, amongst the functional one-way transducers, the ones that admit an equivalent sequential transducer.

In this work, we extend the results of Choffrut to the class of transducers that produce their output string by adding simultaneously, at each transition, a string on the left and a string on the right of the string produced so far. We call them the string-to-context transducers. We obtain a multiple characterisation of the functional string-to-context transducers admitting an equivalent sequential one, based on a Lipschitz property of the function realised by the transducer, and on a pattern (a new twinning property). Last, we prove that given a string-to-context transducer, determining whether there exists an equivalent sequential one is in coNP.

2012 ACM Subject Classification Theory of computation → Models of computation; Theory of computation → Formal languages and automata theory

Keywords and phrases Transducers, Sequentiality, Twinning Property, Two-Way Transducers

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.128

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version An extended version can be found at `https://arxiv.org/abs/1902.11263`.

Funding This work has been funded by the DeLTA project (ANR-16-CE40-0007).

1 Introduction

Transducers are a fundamental model to describe programs manipulating strings. They date back to the very first works in theoretical computer science, and are already present in the pioneering works on finite state automata [25, 1]. While finite state automata are very robust w.r.t. modifications of the model such as non-determinism and two-wayness, this is not the case for transducers. These two extensions do affect the expressive power of the model. Non-determinism is a feature very useful for modelisation and specification purposes. However, when one turns to implementation, deriving a sequential, *i.e.* input-deterministic, transducer is a major issue. A natural and fundamental problem thus consists, given a (non-deterministic) transducer, in deciding whether there exists an equivalent sequential transducer. This problem is called the *sequentiality problem*.

In [12], Choffrut addressed this problem for the class of functional (one-way) finite state transducers, which corresponds to so-called *rational functions*. He proved a multiple characterisation of the transducers admitting an equivalent sequential transducer. This characterisation includes a machine-independent property, namely a Lipschitz property of the function realised by the transducer. It also involves a pattern property, namely the twinning property, that allows to prove that the sequentiality problem is decidable in polynomial

time for the class of functional finite state transducers [27]. This seminal work has led to developments on the sequentiality of finite state transducers [9, 8]. These results have also been extended to weighted automata [11, 22, 17] and to tree transducers [26]. See also [23] for a survey on sequentiality problems.

While the model of one-way transducers is now rather well-understood, a current challenge is to address the so-called class of *regular functions*, which corresponds to functions realised by two-way transducers. This class has attracted a lot of interest during the last years. It is closed under composition [13] and enjoys alternative presentations using logic [16], a deterministic one-way model equipped with registers, named streaming string transducers [2] (SST for short), a set of regular combinators akin to regular expressions for regular languages [5, 7, 14] as well as a class of functions operating on lists [10]. The class of regular functions is much more expressive than the class of rational functions, as it captures for instance the mirror image and the copy. Yet, it has good decidability properties: equivalence and type-checking are decidable in PSpace [21, 3], Similarly, functionality of non-deterministic SST is also decidable in PSpace [4]. We refer the interested reader to [19] for a recent survey. Intuitively, two-way finite state transducers (resp. SST) extend one-way finite state transducers with two important features: firstly, they can go through the input word both ways (resp. they can prepend and append words to registers), and secondly, they can perform multiple passes (resp. they can perform register concatenation).

In this paper, we lift the results of Choffrut [12] to a class of transducers that can perform the first of the two features mentioned above, thus falling strictly between the classes of rational and regular functions. More precisely, we consider non-deterministic transducers which, at each transition, extend the output word produced so far by prepending and appending two words to it. This operation can be defined as the extension of a word with a context, and we call these transducers the *string-to-context transducers*. However, it is important to notice that they still describe functions from strings to strings.

We characterise the functional string-to-context transducers that admit an equivalent *sequential* string-to-context transducer through i) a machine independent property: the function realised by the transducer satisfies a Lipschitz property that involves an original *factor distance* and ii) a pattern property of the transducer which we call *contextual twinning property*, and that generalises the twinning property to contexts. We also prove that the sequentiality problem for these transducers is in the class coNP.

A key technical tool of the result of [12] was a combinatorial analysis of the loops, showing that the output words of synchronised loops have conjugate primitive roots. For string-to-context transducers, the situation is more complex, as the combinatorics may involve the words of the two sides of the context. Intuitively, when these words do commute with the output word produced so far, it is possible for instance to move to the right a part of the word produced on the left. In order to prove our results, we thus dig into the combinatorics of contexts associated with loops, identifying different possible situations, and we then use this analysis to describe an original determinisation construction.

Our results also have a strong connection with the register minimisation problem for SST. This problem consists in determining, given an SST and a natural number k, whether there exists an equivalent SST with k registers. It has been proven in [15] that the problem is decidable for SST that can only append (and not prepend) words to registers, and the proof crucially relies on the fact that the $k = 1$ case exactly corresponds to the sequentiality problem of one-way finite state transducers. Hence, our results constitute a first step towards register minimisation for SST without register concatenation. The register minimisation problem for *non-deterministic* SST has also been studied in [6] for the case of concatenation-free SST. The targeted model being non-deterministic, the two problems are independent.

2 Models

Words, contexts and partial functions

Let A be a finite alphabet. The set of finite words (or strings) over A is denoted by A^*. The empty word is denoted by ϵ. The length of a word u is denoted by $|u|$. We say that a word u is a *prefix* (resp. *suffix*) of a word v if there exists a word y such that $uy = v$ (resp. $yu = v$). We say that two words $u, v \in A^*$ are *conjugates* if there exist two words $t_1, t_2 \in A^*$ such that $u = t_1 t_2$ and $v = t_2 t_1$. If this holds, we write $u \sim v$. The *primitive root* of a word $u \in A^*$, denoted $\rho(u)$, is the shortest word x such that $u = x^p$ for some $p \geqslant 1$. A word is said to be *primitive*, if it is equal to its primitive root. Given a word $u \in B^*$, we say that v is a *factor* of u if there exist words x, y such that $u = xvy$. For $n, m \in \mathbb{N}_{>0}$, we note by $\gcd(n, m)$ the greatest common divisor of n and m.

▶ **Lemma 1** (Fine and Wilf, [20], Chapter 9 of [24]). *Let $x, y \in A^*$ and $m, n \in \mathbb{N}$. If x^m and y^n have a common factor of length at least $|x| + |y| - \gcd(|x|, |y|)$, then their primitive roots are conjugates.*

Given two words $u, v \in A^*$, the *longest common prefix* (resp. *suffix*) of u and v is denoted by $\mathsf{lcp}(u, v)$ (resp. $\mathsf{lcs}(u, v)$). We define the *prefix distance* between u and v, denoted by $dist_p(u, v)$, as $|u| + |v| - 2|\mathsf{lcp}(u, v)|$.

Given two words $u, v \in B^*$, a *longest common factor* of u and v is a word w of maximal length that is a factor of both u and v. Note that this word is not necessarily unique. We denote such a word by $\mathsf{lcf}(u, v)$. The *factor distance* between u and v, denoted by $dist_f(u, v)$, is defined as $dist_f(u, v) = |u| + |v| - 2|\mathsf{lcf}(u, v)|$. This definition is correct as $|\mathsf{lcf}(u, v)|$ is independent of the choice of the common factor of maximal length.

Using a careful case analysis, we can prove that $dist_f$ is indeed a distance, the only difficulty lying in the subadditivity:

▶ **Lemma 2.** *$dist_f$ is a distance.*

Given a finite alphabet B, a *context* on B is a pair of words $(u, v) \in B^* \times B^*$. The set of contexts on B is denoted $\mathcal{C}(B)$. The empty context is denoted by c_ε. For a context $c = (u, v)$, we denote by $\overleftarrow{c}$ (resp. $\overrightarrow{c}$) its left (resp. right) component: $\overleftarrow{c} = u$ (resp. $\overrightarrow{c} = v$). The *length* of a context c is defined by $|c| = |\overleftarrow{c}| + |\overrightarrow{c}|$. The *lateralized length* of a context c is defined by $\|c\| = (|\overleftarrow{c}|, |\overrightarrow{c}|)$. For a context $c \in \mathcal{C}(B)$ and a word $w \in B^*$, we write $c[w]$ for the word $\overleftarrow{c}\, w\, \overrightarrow{c}$. We define the concatenation of two contexts $c_1, c_2 \in \mathcal{C}(B)$ as the context $c_1 c_2 = (\overleftarrow{c_1}\,\overleftarrow{c_2}, \overrightarrow{c_2}\,\overrightarrow{c_1})$. Last, given a context c and a word u, we denote by $c^{-1}[u]$ the unique word v such that $c[v] = u$, when such a word exists.

Given a set of contexts $C \subseteq \mathcal{C}(B)$, we denote by $\mathsf{lcc}(C)$ the longest common context of elements in C, defined as $\mathsf{lcc}(C) = (\mathsf{lcs}(\{\overleftarrow{c} \mid c \in C\}), \mathsf{lcp}(\{\overrightarrow{c} \mid c \in C\}))$. We also write $C.\mathsf{lcc}(C)^{-1} = \{c' \mid c'.\mathsf{lcc}(C) \in C\}$.

We consider two sets X, Y. Given $\Delta \subseteq X \times Y$, we let $\mathsf{dom}(\Delta) = \{x \in X \mid \exists y, (x, y) \in \Delta\}$. We denote the set of partial functions from X to Y as $\mathcal{F}(X, Y)$. Given $f \in \mathcal{F}(X, Y)$, we write $f : X \hookrightarrow Y$, and we denote by $\mathsf{dom}(f)$ its domain. When more convenient, we may also see elements of $\mathcal{F}(X, Y)$ as subsets of $X \times Y$. Last, given $\Delta \subseteq X \times Y$, we let $\mathsf{choose}(\Delta)$ denote some $\Delta' \in \mathcal{F}(X, Y)$ such that $\Delta' \subseteq \Delta$ and $\mathsf{dom}(\Delta) = \mathsf{dom}(\Delta')$.

String-to-Context and String-to-String Transducers

▶ **Definition 3.** *Let A, B be two finite alphabets. A string-to-context transducer (S2C for short) $\mathcal{T}$ from A^* to B^* is a tuple $(Q, t_{init}, t_{final}, T)$ where Q is a finite set of states, $t_{init} : Q \hookrightarrow \mathcal{C}(B)$ (resp. $t_{final} : Q \hookrightarrow \mathcal{C}(B)$) is the finite initial (resp. final) function, $T \subseteq Q \times A \times \mathcal{C}(B) \times Q$ is the finite set of transitions.*

A state q is said to be *initial* (*resp. final*) if $q \in \mathsf{dom}(t_{init})$ (resp. $q \in \mathsf{dom}(t_{final})$). We depict as $\xrightarrow{c} q$ (resp. $q \xrightarrow{c}$) the fact that $t_{init}(q) = c$ (resp. $t_{final}(q) = c$). A run ρ from a state q_1 to a state q_k on a word $w = w_1 \cdots w_k \in A^*$ where for all i, $w_i \in A$, is a sequence of transitions: $(q_1, w_1, c_1, q_2), (q_2, w_2, c_2, q_3), \ldots, (q_k, w_k, c_k, q_{k+1})$. The *output* of such a run is the context $c = c_k \ldots c_2 c_1 \in \mathcal{C}(B)$, and is denoted by $\mathsf{out}(\rho)$. We depict this situation as $q_1 \xrightarrow{w|c} q_{k+1}$. The set of runs of $\mathcal{T}$ is denoted $\mathcal{R}(\mathcal{T})$. The run ρ is said to be *accepting* if q_1 is initial and q_{k+1} final. This string-to-context transducer $\mathcal{T}$ computes a relation $[\![\mathcal{T}]\!] \subseteq A^* \times B^*$ defined by the set of pairs $(w, edc[\varepsilon])$ such that there are $p, q \in Q$ with $\xrightarrow{c} p \xrightarrow{w|d} q \xrightarrow{e}$. Thus, even if its definition involves contexts on B, the semantics of $\mathcal{T}$ is a relation between words on A and words on B. Given an S2C $\mathcal{T} = (Q, t_{init}, t_{final}, T)$, we define the constant $M_{\mathcal{T}}$ as $M_{\mathcal{T}} = \max\{|c| \mid (p, a, c, q) \in T \text{ or } (q, c) \in t_{init} \cup t_{final}\}$. Given $\Delta : Q \hookrightarrow \mathcal{C}(B)$, we denote by $\mathcal{T}_\Delta$ the S2C obtained by replacing t_{init} with Δ. An S2C is *trimmed* if each of its states appears in some accepting run. W.l.o.g., we assume that the string-to-context transducers we consider are trimmed. An S2C $\mathcal{T}$ from A^* to B^* is *functional* if the relation $[\![\mathcal{T}]\!]$ is a function from A^* to B^*. An S2C $\mathcal{T} = (Q, t_{init}, t_{final}, T)$ is *sequential* if $\mathsf{dom}(t_{init})$ is a singleton and if for every transitions $(p, a, c, q), (p, a, c', q') \in T$, we have $q = q'$ and $c = c'$.

The classical model of finite-state transducers is recovered in the following definition:

▶ **Definition 4.** *Let A, B be two finite alphabets. A string-to-context transducer $\mathcal{T} = (Q, t_{init}, t_{final}, T)$ is a string-to-string transducer (S2S for short) from A^* to B^* if, for all $(q, c) \in t_{init} \cup t_{final}$, $\overleftarrow{c} = \varepsilon$, and for all $(q, a, c, q') \in T$, $\overleftarrow{c} = \varepsilon$.*

Notations defined for S2C hold for classical transducers as is. For an S2S, we write $\xrightarrow{w} q$ (resp. $q \xrightarrow{w}$, and $q \xrightarrow{u|w} q'$) instead of $\xrightarrow{(\varepsilon,w)} q$ (resp. $q \xrightarrow{(\varepsilon,w)}$, and $q \xrightarrow{u|(\varepsilon,w)} q'$).

Given an S2C $\mathcal{T} = (Q, t_{init}, t_{final}, T)$, we define its *right S2S*, denoted $\overrightarrow{\mathcal{T}}$, as the tuple $(Q, \overrightarrow{t_{init}}, \overrightarrow{t_{final}}, \overrightarrow{T})$ where, for all $q \in Q$, $\overrightarrow{t_{init}}(q) = \overrightarrow{t_{init}(q)}$ and $\overrightarrow{t_{final}}(q) = \overrightarrow{t_{final}(q)}$, and, for all $(p, a, c, q) \in T$, $(p, a, \overrightarrow{c}, q) \in \overrightarrow{T}$. Its *left S2S* $\overleftarrow{\mathcal{T}}$ is defined similarly, and by applying the mirror image on its output labels.

▶ **Example 5.** Two examples of S2C (not realisable by S2S) are depicted on Figure 1.

(a) $\mathcal{T}_{mirror}$. **(b)** $\mathcal{T}_{partition}$.

Figure 1 1a Example of a S2C $\mathcal{T}_{mirror}$ computing the function $f_{mirror} : u_1 \ldots u_n \in \{a, b\}^* \mapsto u_n \ldots u_1$. 1b Example of a S2C $\mathcal{T}_{partition}$ computing the function $f_{partition} : u \in \{a, b\}^* \mapsto a^{|u|_a} b^{|u|_b}$.

3 Lipschitz and Twinning Properties

We recall the properties considered in [12], and the associated results.

▶ **Definition 6.** *We say that a function $f : A^* \hookrightarrow B^*$ satisfies the* Lipschitz property *if there exists $K \in \mathbb{N}$ such that $\forall u, v \in \mathsf{dom}(f), dist_p(f(u), f(v)) \leqslant K\, dist_p(u, v)$.*

▶ **Definition 7.** *We consider an S2S and $L \in \mathbb{N}$. Two states q_1 and q_2 are said to be L-*twinned *if for any two runs $\xrightarrow{w_1} p_1 \xrightarrow{u|x_1} q_1 \xrightarrow{v|y_1} q_1$ and $\xrightarrow{w_2} p_2 \xrightarrow{u|x_2} q_2 \xrightarrow{v|y_2} q_2$, where p_1 and p_2 are initial, we have for all $j \geqslant 0$, $dist_p(w_1 x_1 y_1^j, w_2 x_2 y_2^j) \leqslant L$. An S2S satisfies the* twinning property (TP) *if there exists $L \in \mathbb{N}$ such that any two of its states are L-twinned.*

▶ **Theorem 8** ([12]). *Let $\mathcal{T}$ be a functional S2S. The following assertions are equivalent:*
1. *there exists an equivalent sequential S2S,*
2. *$[\![\mathcal{T}]\!]$ satisfies the Lipschitz property,*
3. *$\mathcal{T}$ satisfies the twinning property.*

We present the adaptation of these properties to string-to-context transducers.

▶ **Definition 9.** *We say that $f : A^* \hookrightarrow B^*$ satisfies the* contextual Lipschitz property (CLip) *if there exists $K \in \mathbb{N}$ such that $\forall u, v \in \mathsf{dom}(f), dist_f(f(u), f(v)) \leqslant K\ dist_p(u, v)$.*

▶ **Definition 10.** *We consider an S2C and $L \in \mathbb{N}$. Two states q_1 and q_2 are said to be L-*contextually twinned *if for any two runs $\xrightarrow{c_1} p_1 \xrightarrow{u|d_1} q_1 \xrightarrow{v|e_1} q_1$ and $\xrightarrow{c_2} p_2 \xrightarrow{u|d_2} q_2 \xrightarrow{v|e_2} q_2$, where p_1 and p_2 are initial, we have for all $j \geqslant 0$, $dist_f(e_1^j d_1 c_1[\varepsilon], e_2^j d_2 c_2[\varepsilon]) \leqslant L$. An S2C satisfies the* contextual twinning property (CTP) *if there exists $L \in \mathbb{N}$ such that any two of its states are L-contextually twinned.*

4 Main Result

The main result of the paper is the following theorem, which extends to string-to-context transducers the characterisation of sequential transducers amongst functional ones.

▶ **Theorem 11.** *Let $\mathcal{T}$ be a functional S2C. The following assertions are equivalent:*
1. *there exists an equivalent sequential string-to-context transducer,*
2. *$[\![\mathcal{T}]\!]$ satisfies the contextual Lipschitz property,*
3. *$\mathcal{T}$ satisfies the contextual twinning property.*

Proof. The implications $1 \Rightarrow 2$ and $2 \Rightarrow 3$ are proved in Proposition 12 and Proposition 13 respectively. The implication $3 \Rightarrow 1$ is more involved, and is based on a careful analysis of word combinatorics of loops of string-to-context transducers satisfying the CTP. This analysis is summarised in Lemma 22 and used in Section 6 to describe the construction of an equivalent sequential S2C. ◀

▶ **Proposition 12.** *Let $\mathcal{T}$ be a sequential S2C realizing the function f. Then f satisfies the contextual Lipschitz property.*

Proof. We claim that f is context-Lipschitzian with coefficient $3M_{\mathcal{T}}$. Consider two input words u, v in the domain of f. If $u = v$, then the result is trivial. Otherwise, let $w = \mathsf{lcp}(u, v)$ and let $u = w.u'$, with $0 \leqslant |u'|$. Then we have $[\![\mathcal{T}]\!](u) = c_3 c_2 c_1[\epsilon]$ where c_1 is the context produced along w, c_2 the one produced along u', and c_3 is the final output context. Similarly, we can write (with $v = w.v'$, and $0 \leqslant |v'|$) $[\![\mathcal{T}]\!](v) = d_3 d_2 d_1[\epsilon]$. As $\mathcal{T}$ is sequential, we have $d_1 = c_1$. We also have $|c_3| \leqslant M_{\mathcal{T}}$, $|d_3| \leqslant M_{\mathcal{T}}$, $|c_2| \leqslant M_{\mathcal{T}}\ |u'|$ and $|d_2| \leqslant M_{\mathcal{T}}\ |v'|$. Finally, as $u \neq v$, we have $dist_p(u, v) = |u'| + |v'| \geqslant 1$ and we obtain:

$$dist_f(f(u), f(v)) \leqslant |c_3 c_2| + |d_3 d_2| \leqslant M_{\mathcal{T}}\ (2 + |u'| + |v'|) \leqslant 3M_{\mathcal{T}}\ dist_p(u, v)$$ ◀

▶ **Proposition 13.** *Let $\mathcal{T}$ be a functional S2C realizing the function f. If f satisfies the contextual Lipschitz property, then $\mathcal{T}$ satisfies the contextual twinning property.*

Proof. We consider an instance of the CTP and stick to the notations of Definition 10. We denote by n the number of states of $\mathcal{T}$. As $\mathcal{T}$ is trimmed, there exist runs $q_i \xrightarrow{w_i|f_i} r_i \xrightarrow{g_i}$, with $|w_i| \leqslant n$, for $i \in \{1, 2\}$. We consider the input words $\alpha_j = uv^j w_1$ and $\beta_j = uv^j w_2$, for all $j \geqslant 0$. We have, for every j, $dist_p(\alpha_j, \beta_j) \leqslant |w_1| + |w_2| \leqslant 2n$.

The following property of $dist_f$ can be proven using a case analysis:

Fact. For every $w, w' \in B^*$, $c, c' \in \mathcal{C}(B)$, we have $dist_f(w, w') \leqslant dist_f(c[w], c'[w']) + |c| + |c'|$.

As f is K context-Lipschitzian, for some fixed K, we obtain, for all j:

$$\begin{aligned} dist_f(e_1^j d_1 c_1[\varepsilon], e_2^j d_2 c_2[\varepsilon]) &\leqslant dist_f(g_1 f_1 e_1^j d_1 c_1[\varepsilon], g_2 f_2 e_2^j d_2 c_2[\varepsilon]) + 2(n+1)M_{\mathcal{T}} \\ &\leqslant dist_f(f(\alpha_j), f(\beta_j)) + 2(n+1)M_{\mathcal{T}} \\ &\leqslant K\, dist_p(\alpha_j, \beta_j) + 2(n+1)M_{\mathcal{T}} \leqslant 2Kn + 2(n+1)M_{\mathcal{T}} \end{aligned}$$ ◀

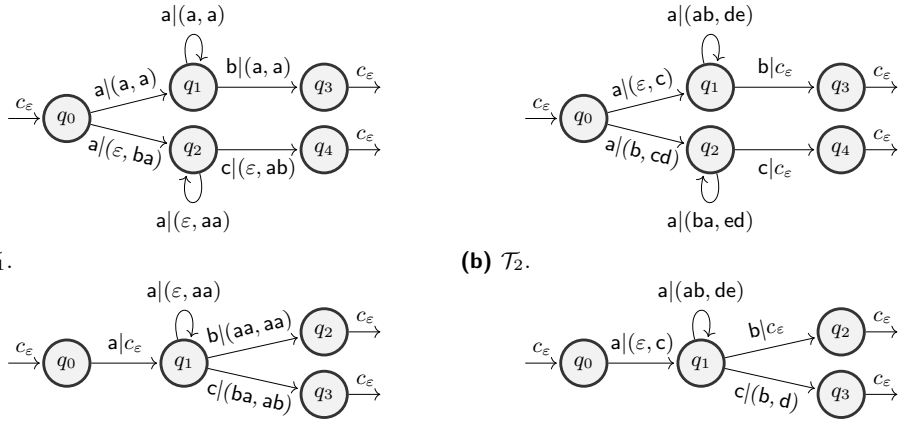
5 Analysis of Loop Combinatorics

The classical twinning property forces the outputs of two runs reading the same input to only diverge by a finite amount. This constraint in turn makes for strong combinatorial bindings between runs involving loops: for two runs $\xrightarrow{w_1} p_1 \xrightarrow{u|x_1} q_1 \xrightarrow{v|y_1} q_1$ and $\xrightarrow{w_2} p_2 \xrightarrow{u|x_2} q_2 \xrightarrow{v|y_2} q_2$, we have $|y_1| = |y_2|$, and $\rho(y_1) \sim \rho(y_2)$. Similar behaviours are expected with string-to-context transducers and lead us to study the combinatorial properties of synchronised runs involving loops in those machines. Throughout this section, we consider a string-to-context transducer $\mathcal{T} = (Q, t_{init}, t_{final}, T)$ that satisfies the contextual twinning property.

5.1 Behaviours of Loops

We start with two examples illustrating how output contexts of synchronised loops can be modified to obtain an equivalent sequential S2C.

▶ **Example 14.** Figure 2a shows an example of a non-sequential functional S2C transducer $\mathcal{T}_1$. The contexts produced on loops around states q_1 and q_2 both commute with word a. This observation can be used to build an equivalent sequential S2C $\mathcal{D}_1$, depicted on Figure 2c. Figure 2b shows an example of a non-sequential functional S2C transducer $\mathcal{T}_2$ where output contexts are non-commuting, but can be slightly shifted so as to be aligned. This observation can be used to build an equivalent sequential S2C $\mathcal{D}_2$, depicted on Figure 2d.



(a) $\mathcal{T}_1$. **(b)** $\mathcal{T}_2$. **(c)** $\mathcal{D}_1$. **(d)** $\mathcal{D}_2$.

Figure 2 2a An S2C $\mathcal{T}_1$ computing the function that maps $a^n b$ to a^{2n+2} and $a^n c$ to $ba^{2n}b$. 2c A sequential S2C $\mathcal{D}_1$ equivalent to $\mathcal{T}_1$. 2b An S2C $\mathcal{T}_2$ computing the function that maps $a^n b$ to $(ab)^{n-1}c(de)^{n-1}$ and $a^n c$ to $b(ab)^{n-1}c(de)^{n-1}d$. 2d A sequential S2C $\mathcal{D}_2$ equivalent to $\mathcal{T}_2$.

The following definition follows from the intuition drawn by the previous example.

▶ **Definition 15** (Lasso, Aligned/Commuting/Non-commuting lasso). *A lasso around a state q is a run ρ of the form $\xrightarrow{c} p \xrightarrow{u|d} q \xrightarrow{v|e} q$ with p an initial state. ρ is said to be productive, if $|e| \neq 0$. We say that ρ is:*

- *aligned w.r.t. f and w, for some $f \in \mathcal{C}(B)$ and $w \in B^*$, denoted as $(f,w)-$*aligned*, if there exists a context $g \in \mathcal{C}(B)$ such that for all $i \in \mathbb{N}$, $e^i dc[\varepsilon] = gf^i[w]$.*
- *commuting w.r.t. x, for some $x \in B^+$, denoted as $x-$*commuting*, if there exists a context $f \in \mathcal{C}(B)$ such that for all $i \in \mathbb{N}_{>0}$, there exists $k \in \mathbb{N}$ such that $e^i dc[\varepsilon] = f[x^k]$.*
- *non-commuting if there exists no word $x \in B^+$ such that ρ is commuting w.r.t x.*

Two lassos $\xrightarrow{c_1} p_1 \xrightarrow{u_1|d_1} q_1 \xrightarrow{v_1|e_1} q_1$ and $\xrightarrow{c_2} p_2 \xrightarrow{u_2|d_2} q_2 \xrightarrow{v_2|e_2} q_2$ are said to be synchronised if $u_1 = u_2$ and $v_1 = v_2$. They are said to be strongly balanced if $\|e_1\| = \|e_2\|$.

Given an integer $k \geqslant 1$, we consider the k-th power of $\mathcal{T}$, that we denote by $\mathcal{T}^k$. A run in $\mathcal{T}^k$ naturally corresponds to k synchronised runs in $\mathcal{T}$, *i.e.* on the same input word. We lift the notion of lasso to $\mathcal{T}^k$, and we denote them by H_1H_2, where H_1 starts in initial states and ends in some state $q = (q_i)_{i\in\{1,\dots,k\}} \in Q^k$, and H_2 is a loop around state q. In the sequel, we will only consider lassos such that q contains pairwise distinct states ($q_i \neq q_j$ for all $i \neq j$). Those lassos are included in the lassos in $\mathcal{T}^{\leqslant|Q|} = \cup_{1\leqslant k\leqslant|Q|}\mathcal{T}^k$.

The intuition given by Example 14 is formalised in the following Lemma:

▶ **Lemma 16.** *Let $H_1H_2 = (\rho_j)_{j\in\{1,\dots,k\}}$ a lasso in $\mathcal{T}^k$, for some $1 \leqslant k \leqslant |Q|$. We write $\rho_j : \xrightarrow{c_j} p_j \xrightarrow{u_1|d_j} q_j \xrightarrow{u_2|e_j} q_j$ for each j. Then there exists an integer $m \in \mathbb{N}$ such that $|e_j| = m$ for all $j \in \{1,\dots,k\}$. If $m > 0$, we say that the lasso H_1H_2 is productive, and:*

- *either there exists $x \in B^+$ primitive such that ρ_j is $x-$*commuting* for all $j \in \{1,\dots,k\}$. In this case, we say that the lasso H_1H_2 is $x-$*commuting*, and we let $\mathsf{pow}_c(x,H_1,H_2) = m/|x|$ and $\mathsf{split}_c(x,H_1,H_2) = \{(q_j,f_j) \mid j \in \{1,\dots,k\}\}$ where $f_j \in \mathcal{C}(B)$ is such that $\forall\alpha \in \mathbb{N}, e_j^\alpha d_j c_j[\varepsilon] = f_j[x^{\alpha\ \mathsf{pow}_c(x,H_1,H_2)}]$.*[1]
- *or there exist $f \in \mathcal{C}(B)$ and $w \in B^*$ such that ρ_j is non-commuting and $(f,w)-$*aligned* for all $j \in \{1,\dots,k\}$. In this case, we say that the lasso H_1H_2 is $(f,w)-$*aligned*, and we let $\mathsf{split}_{nc}(f,w,H_1,H_2) = \{(q_j,g_j) \mid j \in \{1,\dots,k\}\}$ where $g_j \in \mathcal{C}(B)$ is such that $\forall\alpha \in \mathbb{N}, e_j^\alpha d_j c_j[\varepsilon] = g_j f^\alpha[w]$.*[1]

Proof Sketch. As $\mathcal{T}$ satisfies the CTP, the outputs must grow at the same pace when the loops are pumped. This entails that the lengths of the e_j must be equal. Next, the result is proved by considering two productive synchronised lassos, with loops producing respectively e_1 and e_2. If they are not strongly balanced or one of them is $x-$commuting, for some $x \in B^+$, then, using the result of Fine and Wilf (Lemma 1) between $\overleftarrow{e_1}, \overleftarrow{e_2}, \overrightarrow{e_1}$ and $\overrightarrow{e_2}$, we can prove that the other one is also $x-$commuting. Otherwise, they are both non-commuting and strongly balanced. Using again Lemma 1 but first between $\overleftarrow{e_1}$ and $\overleftarrow{e_2}$, and then between $\overrightarrow{e_1}$ and $\overrightarrow{e_2}$, we prove that there exist $f \in \mathcal{C}(B)$ and $w \in B^*$ such that ρ_1 and ρ_2 are $(f,w)-$aligned. Finally, the result is lifted to k productive synchronised lassos. ◀

▶ **Example 17.** We consider the example S2C in Figure 2. The lasso in $\mathcal{T}_1^2$ around (q_1,q_2) is $\mathsf{a}-$commuting. We can compute a pow_c of 2 and $\{(q_1,(\mathsf{a},\mathsf{a})),(q_2,(\mathsf{b},\mathsf{a}))\}$ as a possible split_c. The lasso in $\mathcal{T}_2^2$ around (q_1,q_2) is $((\mathsf{ab},\mathsf{de}),\mathsf{c})-$aligned. We can compute $\{(q_1,c_\varepsilon),(q_2,(\mathsf{b},\mathsf{d}))\}$ as a possible split_{nc}.

[1] Because we only consider lassos around pairwise distinct states, both $\mathsf{split}_c(x,H_1,H_2)$ and $\mathsf{split}_{nc}(f,w,H_1,H_2)$ are partial functions from Q to $\mathcal{C}(B)$.

5.2 Analysis of Loops Consecutive to a Productive Loop

Consider a run that contains two consecutive productive loops. We can observe that the type (commuting or non-commuting) of the lasso involving the first loop impacts the possible types of the lasso involving the second loop. For instance, it is intuitive that a non-commuting lasso cannot be followed by a commuting lasso. Similarly, an $x-$commuting lasso cannot be followed by a $y-$commuting lasso, if x and y are not conjugates. We will see that loops following a first productive loop indeed satisfy stronger combinatorial properties. The following definition characterises their properties.

▶ **Definition 18** (Strongly commuting/Strongly aligned lasso). *Let ρ be a productive lasso $\xrightarrow{c} p \xrightarrow{u|d} q \xrightarrow{v|e} q$ and $x \in B^+$. We say that ρ is:*

- *strongly commuting w.r.t. x, denoted as* strongly$-x-$commuting*, if there exists a context $f \in \mathcal{C}(B)$ such that for all $i, j \in \mathbb{N}_{>0}$, there exists $k \in \mathbb{N}$ such that $e^i dc[x^j] = f[x^k]$.*
- *strongly aligned w.r.t. g, f and x, denoted as* strongly$-(g,f,x)-$aligned*, if there exists a context $h \in \mathcal{C}(B)$ such that for all $i, j \in \mathbb{N}$, $e^j dc[x^i] = hg^j f[x^i]$.*

The following Lemma states the properties of a lasso consecutive to a commuting lasso. To prove it, we proceed as for Lemma 16 by proving the result first for two runs and then lifting it to k runs. The case of two runs is obtained by distinguishing whether they are strongly balanced or not, and using Lemma 1.

▶ **Lemma 19.** *Let H_1H_2 a productive $x-$*commuting *lasso in $\mathcal{T}^{\leqslant |Q|}$, for some $x \in B^+$. Let $\Delta = \mathsf{split}_c(x, H_1, H_2)$ and $H_3H_4 = (\rho_j)_{j\in\{1,\dots,k\}}$ a productive lasso in $\mathcal{T}^k_\Delta$, for some $1 \leqslant k \leqslant |Q|$. We write $\rho_j : \xrightarrow{c_j} p_j \xrightarrow{u_1|d_j} q_j \xrightarrow{u_2|e_j} q_j$ for each j. Then:*

- *either every ρ_j is* strongly$-x-$commuting*: we say that H_3H_4 is* strongly$-x-$commuting*,*
- *or there exist $g, h \in \mathcal{C}(B)$ such that every ρ_j is* strongly$-(h,g,x)-$aligned*. In this case, we say that H_3H_4 is* strongly$-(h,g,x)-$aligned *and we let $\mathsf{extract}_{nc}(h, g, x, \Delta, H_3, H_4) = \{(q_j, h_j) \mid j \in \{1, \dots, k\}\}$ where $h_j \in \mathcal{C}(B)$ is s.t. $\forall \alpha, \beta \in \mathbb{N}, e_j^\alpha d_j c_j[x^\beta] = h_j h^\alpha g[x^\beta]$.*

The following Lemma states that once a non-commuting loop is encountered, then the alignment of production is fixed, *i.e.* no transfer between left and right productions is possible anymore. Hence, the left and right S2S derived from the S2C both satisfy the twinning property:

▶ **Lemma 20.** *Let H_1H_2 be a productive non-commuting lasso that is either*

- *$(f,w)-$*aligned *in $\mathcal{T}^{\leqslant |Q|}$, for some $f \in \mathcal{C}(B)$ and $w \in B^*$, and $\Delta' = \mathsf{split}_{nc}(f, w, H_1, H_2)$,*
- *or* strongly$-(g,f,x)-$aligned *in $\mathcal{T}^{\leqslant |Q|}_\Delta$, for some $g, f \in \mathcal{C}(B)$ and $\Delta \in \mathcal{F}(Q, \mathcal{C}(B))$, and $\Delta' = \mathsf{extract}_{nc}(g, f, x, \Delta, H_1, H_2)$.*

Then $\overleftarrow{\mathcal{T}_{\Delta'}}$ and $\overrightarrow{\mathcal{T}_{\Delta'}}$ both satisfy the twinning property.

5.3 A Two-loop Pattern Property

The following 2-loop property summarises the combinatorial properties of the synchronised runs involving loops in string-to-context transducers that satisfy the CTP.

▶ **Definition 21** (2-loop property). *Given four runs H_1, H_2, H_3, H_4 in $\mathcal{T}^{\leqslant |Q|}$, such that H_1H_2 and $(H_1H_3)H_4$ are lassos in $\mathcal{T}^{\leqslant |Q|}$, we say that they satisfy the 2-loop property if:*

1. *H_1H_2 is*
   - **a.** *either non productive,*
   - **b.** *or productive and $x-$*commuting*, for some $x \in B^+$,*
   - **c.** *or productive, non-commuting and $(f,w)-$*aligned*, for some $f \in \mathcal{C}(B)$ and $w \in B^*$.*

2. *if* H_1H_2 *is productive and* $x-$commuting*, we let* $\Delta = \mathsf{split}_c(x, H_1, H_2)$*, then* H_3H_4 *is a lasso in* $\mathcal{T}_\Delta^{\leqslant |Q|}$*. If productive then it is:*
   a. *either* strongly$-x-$commuting*,*
   b. *or non-commuting and* strongly$-(h, g, x)-$aligned*, for some* $g, h \in \mathcal{C}(B)$*. We let* $\Delta' = \mathsf{extract}_{nc}(h, g, x, \Delta, H_3, H_4)$*, then* $\overleftarrow{\mathcal{T}_{\Delta'}}$ *and* $\overrightarrow{\mathcal{T}_{\Delta'}}$ *both satisfy the twinning property.*
3. *if* H_1H_2 *is productive, non-commuting and* $(f, w)-$aligned*, we let* $\Delta = \mathsf{split}_{nc}(f, w, H_1, H_2)$*, then* $\overleftarrow{\mathcal{T}_\Delta}$ *and* $\overrightarrow{\mathcal{T}_\Delta}$ *both satisfy the twinning property.*

A string-to-context transducer $\mathcal{T}$ *is said to satisfy the 2-loop property if for all runs* H_1, H_2, H_3, H_4 *as above, they satisfy the 2-loop property.*

As a consequence of Lemmas 16, 19 and 20, we have:

▶ **Lemma 22.** *If an S2C* $\mathcal{T}$ *satisfies the CTP then it satisfies the 2-loop property.*

6 Determinisation

Throughout this section, we consider a string-to-context transducer $\mathcal{T} = (Q, t_{init}, t_{final}, T)$ from A^* to B^* that satisfies the 2-loop property. Intuitively, our construction stores the set of possible runs of $\mathcal{T}$, starting in an initial state, on the input word read so far. These runs are incrementally simplified by erasing synchronised loops, and by replacing a prefix by a partial function $\Delta : Q \hookrightarrow \mathcal{C}(B)$. These simplifications are based on the 2-loop property.

Observation. It is worth noticing that, as $\mathcal{T}$ is functional, if two runs reach the same state, it is safe to keep only one of them. This allows us to maintain a set of at most $|Q|$ runs.

Notations. Given $\Delta \in \mathcal{F}(Q, \mathcal{C}(B))$, $c \in \mathcal{C}(B)$, $w \in B^*$, $a \in A$ and $H \in \mathcal{R}(\mathcal{T}^{\leqslant |Q|})$, we define the following notations and operations:

- $\Delta c = \{(q, dc) \mid (q, d) \in \Delta\}$,
- $\Delta[w] = \{(q, d[w]) \mid (q, d) \in \Delta\}$,
- $\Delta \bullet a = \mathsf{choose}(\{(q', dc) \mid (q, c) \in \Delta \text{ and } q \xrightarrow{a|d} q'\})$,
- $H \bullet a \in \mathcal{R}(\mathcal{T}^{\leqslant |Q|})$ is the run obtained by extending runs of H with consecutive transitions of $\mathcal{T}$ associated with input symbol a, and by eliminating runs so as to ensure that runs reach pairwise distinct states of $\mathcal{T}$,
- $\Delta \bullet H = \mathsf{choose}(\{(q', dc) \mid (q, c) \in \Delta \text{ and there is a run } \rho : q \xrightarrow{x|d} q' \in H\})$,
- $id_\Delta = (q_i)_{1 \leqslant i \leqslant k} \in \mathcal{R}(\mathcal{T}^k)$, for some enumeration $\{q_1, \ldots, q_k\}$ of $\mathsf{dom}(\Delta)$.

Construction. We define an equivalent deterministic string-to-context transducer $\overline{\mathcal{D}} = (\overline{Q}, \overline{t_{init}}, \overline{t_{final}}, \overline{T})$, and we denote by $\mathcal{D}$ its trim part. While $\overline{\mathcal{D}}$ may have infinitely many states, we will prove that $\mathcal{D}$ is finite. Formally, we define $\overline{Q} = \overline{Q}_{\mathsf{start}} \uplus \overline{Q}_{\mathsf{com}} \uplus \overline{Q}_{\neg\mathsf{com}}$ where:

- $\overline{Q}_{\mathsf{start}} = \{(\varepsilon, t_{init}, H) \mid H \in \mathcal{R}(\mathcal{T}^{\leqslant |Q|})\}$
- $\overline{Q}_{\mathsf{com}} = \{(x, \Delta, H) \mid x \in B^+, \Delta \in \mathcal{F}(Q, \mathcal{C}(B)), H \in \mathcal{R}(\mathcal{T}^{\leqslant |Q|})\}$
- $\overline{Q}_{\neg\mathsf{com}} = \{(\bot, \Delta, id_\Delta) \mid \Delta \in \mathcal{F}(Q, \mathcal{C}(B))\}$.

By definition, we have $\overline{Q} \subseteq (B^* \cup \{\bot\}) \times \mathcal{F}(Q, \mathcal{C}(B)) \times \mathcal{R}(\mathcal{T}^{\leqslant |Q|}) = \overline{Q}_\infty$. Given $\overline{q} = (x, \Delta, H) \in \overline{Q}_\infty$, we let $\Delta_{\overline{q}} = \Delta \bullet H \in \mathcal{F}(Q, \mathcal{C}(B))$. An invariant of our construction is that every starting state of a run in H belongs to $\mathsf{dom}(\Delta)$.

Intuitively, the semantics of a state $\overline{q} = (x, \Delta, H) \in \overline{Q}$ can be understood as follows: x is used to code the type of state ($\overline{Q}_{\mathsf{start}}$, $\overline{Q}_{\mathsf{com}}$ or $\overline{Q}_{\neg\mathsf{com}}$), and Δ and H are used to represent the runs that remain to be executed to faithfully simulate the runs of $\mathcal{T}$ on the input word u read so far. As we have seen in the previous section, loops may either be commuting, allowing to shift some parts of the output from one side of the context to the other side, or they are non-commuting, and then should be aligned, forbidding such modifications. Intuitively, states in $\overline{Q}_{\mathsf{start}}$ correspond to situations in which no productive loop has been encountered yet. States in $\overline{Q}_{\mathsf{com}}$ (with $x \in B^+$) correspond to situations in which only x-commuting loops have been encountered. States in $\overline{Q}_{\neg\mathsf{com}}$ correspond to situations in which a non-commuting loop has been encountered. A representation of $\mathcal{D}$ is given in Figure 3.

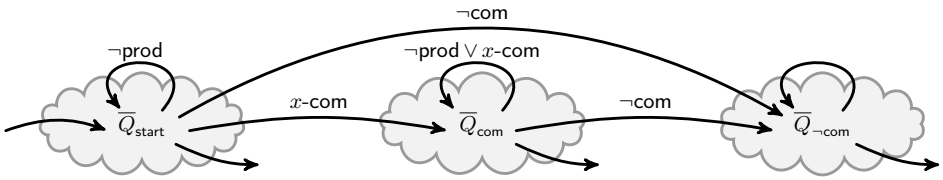


Figure 3 A schematic representation of states and transitions of $\mathcal{D}$.

Initial and final states. They are defined as follows:

- $\overline{t_{init}} = \{(\overline{i}, c_\varepsilon)\}$ where $\overline{i} = (\varepsilon, t_{init}, id_{t_{init}}) \in \overline{Q}_{\mathsf{start}}$
- $\overline{t_{final}} = \mathsf{choose}(\{(\overline{q}, dc) \mid \overline{q} \in \overline{Q}, (p, c) \in \Delta_{\overline{q}}, (p, d) \in t_{final}\})$

Transitions. They are defined as follows:

- $\overline{\mathcal{D}} = \{\overline{p} \xrightarrow{a|c} \overline{q} \mid \overline{p} = (x, \Delta, H) \in \overline{Q}, a \in A \text{ and } (\overline{q}, c) = \textsc{simplify}((x, \Delta, H \bullet a))\}$

Intuitively, a transition of $\overline{\mathcal{D}}$ leaving some state $\overline{p} = (x, \Delta, H) \in \overline{Q}$ with letter $a \in A$ aims at first extending H with a, obtaining the new set of runs $H \bullet a$, and then simplifying this set of runs by removing loops, using the function SIMPLIFY. This function is implemented as Algorithm 2, which calls Algorithm 1 to remove all loops of $H \bullet a$ one by one. Depending on the type of the loop encountered, the type of the state is updated.

We first define EXTEND_WITH_LOOP$(\overline{p}, H_2)$ in Algorithm 1 that takes as input a state $\overline{p} = (x, \Delta, H_1) \in \overline{Q}_{\mathsf{start}} \cup \overline{Q}_{\mathsf{com}}$ and a run H_2 in $\mathcal{T}^{\leqslant |Q|}$ such that $H_1 H_2$ is a lasso in $\mathcal{T}_\Delta^{\leqslant |Q|}$. The algorithm enumerates the possible cases for the type of this lasso, depending on the type of $\overline{p}$. This enumeration strongly relies on the 2-loop property. Depending on the case, the loop is processed, and a pair composed of a new state and a context is returned. This context will be part of the output associated with the transition. By a case analysis, we prove:

▶ **Lemma 23.** *Let $\overline{p} = (x, \Delta, H_1) \in \overline{Q}_{start} \cup \overline{Q}_{com}$ and $H_2 \in \mathcal{R}(\mathcal{T}^{\leqslant |Q|})$ such that $H_1 H_2$ is a lasso in $\mathcal{T}_\Delta^{\leqslant |Q|}$. We let $(\overline{q}, c) =$ EXTEND_WITH_LOOP$(\overline{p}, H_2)$.*

- *If $x = \varepsilon$ then $(\Delta_{\overline{p}} \bullet H_2)[\varepsilon] = \Delta_{\overline{q}} c[\varepsilon]$.*
- *If $x \in B^+$ then for all $k \in \mathbb{N}$, $(\Delta_{\overline{p}} \bullet H_2)[x^k] = \Delta_{\overline{q}} c[x^k]$.*

We then define SIMPLIFY$(\overline{p})$ in Algorithm 2 that takes as input a state $\overline{p} \in \overline{Q}_\infty$ (we need to consider $\overline{Q}_\infty$ as input and not only $\overline{Q}$ because of the recursive calls) and returns a pair composed of a new state and a context. Intuitively, it recursively processes the lassos present in the runs stored by the state $\overline{p}$, by using calls to the previous algorithm. The following result is proved by induction, using Lemma 23:

Algorithm 1 Extending a state $\overline{p} = (x, \Delta, H_1) \in \overline{Q}_{\mathsf{start}} \cup \overline{Q}_{\mathsf{com}}$ with $H_2 \in \mathcal{R}(\mathcal{T}^{\leqslant |Q|})$ s.t. H_1H_2 is a lasso in $\mathcal{T}_\Delta^{\leqslant |Q|}$.

1: **function** EXTEND_WITH_LOOP($\overline{p}, H_2$)
2: **if** H_2 is non-productive **then**
3: **return** $(\overline{p}, c_\varepsilon)$
4: **else if** $\overline{p} = (\varepsilon, t_{init}, H_1)$ **then**
5: **if** H_1H_2 is $x-$commuting, for some $x \in B^+$, **then**
6: **let** $\Delta = \mathsf{split}_c(x, H_1, H_2)$ and $k = \mathsf{pow}_c(x, H_1, H_2)$
7: **return** $((x, \Delta, id_\Delta), (\varepsilon, x^k))$
8: **else if** H_1H_2 is $(f, w)-$aligned, for some $f \in \mathcal{C}(B)$ and $w \in B^*$, **then**
9: **let** $\Delta = \mathsf{split}_{nc}(f, w, H_1, H_2)$
10: **return** $((\bot, \Delta, id_\Delta), f \cdot (\varepsilon, w))$
11: **end if**
12: **else if** $\overline{p} = (x, \Delta_0, H_1)$, where $x \in B^+$, **then**
13: **if** H_1H_2 is strongly$-x-$commuting **then**
14: **let** $k = |\mathsf{out}(H_2)|/|x|$
15: **return** $(\overline{p}, (\varepsilon, x^k))$
16: **else if** H_1H_2 is strongly$-(g, f, x)-$aligned, for some $g, f \in \mathcal{C}(B)$, **then**
17: **let** $\Delta = \mathsf{extract}_{nc}(g, f, x, \Delta_0, H_1, H_2)$
18: **return** $((\bot, \Delta, id_\Delta), gf)$
19: **end if**
20: **end if**
21: **end function**

▶ **Lemma 24.** *Let $\overline{p} = (x, \Delta, H) \in \overline{Q}_\infty$ and $(\overline{q}, c) = \text{SIMPLIFY}(\overline{p})$. Then $\overline{q} \in \overline{Q}$ and we have:*

- *If $x = \varepsilon$ then $\Delta_{\overline{p}}[\varepsilon] = \Delta_{\overline{q}}c[\varepsilon]$.*
- *If $x \in B^+$ then for all $k \in \mathbb{N}$, $\Delta_{\overline{p}}[x^k] = \Delta_{\overline{q}}c[x^k]$.*
- *If $x = \bot$ then $\Delta_{\overline{p}} = \Delta_{\overline{q}}c$.*

▶ **Theorem 25.** *$\mathcal{D}$ is a finite sequential string-to-context transducer equivalent to $\mathcal{T}$.*

Proof Sketch. First observe that $\mathcal{D}$ is sequential. The correctness of $\mathcal{D}$ is a consequence of the following property, that we prove using Lemma 24 and an induction on $|u|$: for all $u \in A^*$, if we have $\overline{i} \xrightarrow{u|c} \overline{q}$ in $\mathcal{D}$, then $\Delta_{\overline{q}}c[\varepsilon] = (t_{init} \bullet u)[\varepsilon]$. Last, we prove that $\mathcal{D}$ is finite. By construction, for every state $\overline{q} = (x, \Delta, H)$ of $\mathcal{D}$, H contains no loop, hence its length is bounded by $|Q|^{|Q|}$. This can be used to bound the size of x, as well as the size of Δ, for states in $\overline{Q}_{\mathsf{start}} \cup \overline{Q}_{\mathsf{com}}$. The case of states in $\overline{Q}_{\neg\mathsf{com}}$ is different: when such a state $(\bot, \Delta, id_\Delta)$ is reached, then by the 2-loop property, the transducers $\overleftarrow{\mathcal{T}_\Delta}$ and $\overrightarrow{\mathcal{T}_\Delta}$ both satisfy the (classical) twinning property. It remains to observe that the operations performed on Line 24 precisely correspond to two determinisations of [12], on both sides of the S2C. ◀

7 Decision

In this section, we prove the following result:

▶ **Theorem 26.** *Given a string-to-context transducer, determining whether there exists an equivalent sequential string-to-context transducer is in coNP.*

Algorithm 2 Simplifying a state $\overline{p} = (x, \Delta, H) \in \overline{Q}_\infty$.

22: **function** SIMPLIFY($\overline{p}$)
23: **if** $\overline{p} = (\bot, \Delta, H)$ **then**
24: **let** $\Delta' = \Delta \bullet H$, $c = \mathsf{lcc}(\Delta')$ and $\overline{q} = (\bot, \Delta'.c^{-1}, id_{\Delta'})$
25: **return** $(\overline{q}, c)$
26: **else if** $\overline{p} = (x, \Delta, H_1H_2H_3)$, where $x \in B^*$ and H_2 is the first loop in H, **then**
27: **let** $\overline{q} = (x, \Delta, H_1)$
28: **let** $(\overline{r}, c) =$ EXTEND_WITH_LOOP$(\overline{q}, H_2)$ with $\overline{r} = (x', \Delta', H')$
29: **let** $(\overline{s}, d) =$ SIMPLIFY$((x', \Delta', H'.H_3))$
30: **return** $(\overline{s}, dc)$
31: **else**
32: **return** $(\overline{p}, c_\varepsilon)$
33: **end if**
34: **end function**

In order to show this result, we introduce a restriction of the 2-loop property:

▶ **Definition 27** (small-2-loop property). *A string-to-context transducer $\mathcal{T}$ is said to satisfy the small-2-loop property if, for all runs $H_1, H_2, H_3, H_4 \in \mathcal{T}^2$ with $|H_i| \leqslant |Q|^2$ for each i, H_1H_2, $H_1H_3H_4$ are lassos and they satisfy the 2-loop property (in the sense of Definition 21).*

By definition, if a string-to-context transducer satisfies the 2-loop property then it also satisfies the small-2-loop property. We will show that the two properties are equivalent.

▶ **Lemma 28.** *If a string-to-context transducer $\mathcal{T}$ satisfies the small-2-loop property then $[\![\mathcal{T}]\!]$ satisfies the contextual Lipschitz property.*

Proof Sketch. We claim there exists $K \in \mathbb{N}$ such that for every pair of synchronised runs $H : \xrightarrow{(c_0,d_0)} (p_0, q_0) \xrightarrow{u|(c_1,d_1)} (p_1, q_1)$ in $\mathcal{T}^2$, we have $dist_f(c_1c_0[\varepsilon], d_1d_0[\varepsilon]) \leqslant K$. The result then easily follows. To prove this claim, we apply the main procedure SIMPLIFY (see Section 6) to the state $\overline{p} = (\varepsilon, t_{init}, H)$. This procedure can indeed be applied: as it always processes the *first* loop (see Line 29), the lassos considered satisfy the premises of the small-2-loop property. The claim follows from the proof of finiteness of $\mathcal{D}$. ◀

Proof Sketch of Theorem 26. By Theorem 11 and Lemma 28, $\mathcal{T}$ admits an equivalent sequential S2C transducer iff $\mathcal{T}$ satisfies the small-2-loop property (see also Figure 4). Thus, we describe a procedure to decide whether $\mathcal{T}$ satisfies the small-2-loop property.

The procedure first non-deterministically guesses a counter-example to the small-2-loop property and then verifies that it is indeed a counter-example. By definition of the small-2-loop property, the counter-example can have finitely many shapes. Those shapes require the verification of the properties of the involved lassos: being productive or not, being commuting or not, being aligned or not, satisfying the (classical) twinning property, etc.

Verifying that a lasso in $\mathcal{T}^2$ is not commuting (resp. not aligned) boils down to checking whether there exists no $x \in B^+$ such that the lasso is $x-$commuting (resp. no $f \in \mathcal{C}(B)$ and $w \in B^*$ such that the lasso is $(f, w)-$aligned). In both cases, the search space for the words x, w and context f can be narrowed down to factors of the output contexts of the given lasso. Thus these verifications can be done in polynomial time. The classical twinning property can also be checked in polynomial time. As a summary, we can show that the verifications for all the shapes can be done in polynomial time. Furthermore, all the shapes are of polynomial size, by definition of the small-2-loop property, yielding the result. ◀

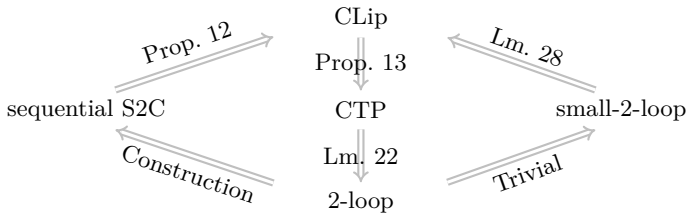


Figure 4 Overview of the equivalent properties we consider.

Note that if one can express in the logic of [18] that a lasso in $\mathcal{T}^2$ is not commuting (resp. not aligned), then this would show that the problem can be solved in polynomial time. However, this seems difficult because of the universal quantification on the factor x.

8 Conclusion

We have proposed a multiple characterisation of string-to-context transducers that admit an equivalent sequential S2C, including a machine independent property, a pattern property, as well as a "small" pattern property allowing to derive a decision procedure running in non-deterministic polynomial time. All these equivalences are summarised in Figure 4. Future work includes a lower bound for the complexity of the problem, the extension of this work to the register minimisation problem for streaming string transducers without register concatenation, and the extension of our results to infinite words.

References

1 Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. A General Theory of Translation. *Mathematical Systems Theory*, 3(3):193–221, 1969. doi:10.1007/BF01703920.

2 Rajeev Alur and Pavol Černý. Expressiveness of streaming string transducers. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 8 of *LIPIcs*, pages 1–12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.

3 Rajeev Alur and Pavol Černý. Streaming transducers for algorithmic verification of single-pass list-processing programs. In *Proc. of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011*, pages 599–610. ACM, 2011.

4 Rajeev Alur and Jyotirmoy V. Deshmukh. Nondeterministic Streaming String Transducers. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II*, volume 6756 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2011.

5 Rajeev Alur, Adam Freilich, and Mukund Raghothaman. Regular combinators for string transformations. In *CSL-LICS '14*, pages 9:1–9:10. ACM, 2014.

6 Félix Baschenis, Olivier Gauwin, Anca Muscholl, and Gabriele Puppis. Minimizing Resources of Sweeping and Streaming String Transducers. In *ICALP 2016*, volume 55 of *LIPIcs*, pages 114:1–114:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

7 Nicolas Baudru and Pierre-Alain Reynier. From Two-Way Transducers to Regular Function Expressions. In *Developments in Language Theory - 22nd International Conference, DLT 2018, Tokyo, Japan, September 10-14, 2018, Proceedings*, volume 11088 of *Lecture Notes in Computer Science*, pages 96–108. Springer, 2018.

8 Marie-Pierre Béal and Olivier Carton. Determinization of transducers over finite and infinite words. *Theoretical Computer Science*, 289(1):225–251, 2002. doi:10.1016/S0304-3975(01)00271-7.

9 Marie-Pierre Béal, Olivier Carton, Christophe Prieur, and Jacques Sakarovitch. Squaring transducers: an efficient procedure for deciding functionality and sequentiality. *Theoretical Computer Science*, 292(1):45–63, 2003.

10 Mikolaj Boja'nczyk, Laure Daviaud, and Shankara Narayanan Krishna. Regular and First-Order List Functions. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 125–134. ACM, 2018.

11 Adam L. Buchsbaum, Raffaele Giancarlo, and Jeffery Westbrook. On the Determinization of Weighted Finite Automata. *SIAM J. Comput.*, 30(5):1502–1531, 2000. doi:10.1137/S0097539798346676.

12 Christian Choffrut. Une Caractérisation des Fonctions Séquentielles et des Fonctions Sous-Séquentielles en tant que Relations Rationnelles. *Theor. Comput. Sci.*, 5(3):325–337, 1977. doi:10.1016/0304-3975(77)90049-4.

13 Michal Chytil and Vojtech Jákl. Serial Composition of 2-Way Finite-State Transducers and Simple Programs on Strings. In *Automata, Languages and Programming, Fourth Colloquium, University of Turku, Finland, July 18-22, 1977, Proceedings*, volume 52 of *Lecture Notes in Computer Science*, pages 135–147. Springer, 1977.

14 Vrunda Dave, Paul Gastin, and Shankara Narayanan Krishna. Regular Transducer Expressions for Regular Transformations. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 315–324. ACM, 2018.

15 Laure Daviaud, Pierre-Alain Reynier, and Jean-Marc Talbot. A Generalised Twinning Property for Minimisation of Cost Register Automata. In *LICS '16*, pages 857–866. ACM, 2016.

16 Joost Engelfriet and Hendrik Jan Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Log.*, 2(2):216–254, 2001.

17 Emmanuel Filiot, Raffaella Gentilini, and Jean-François Raskin. Quantitative Languages Defined by Functional Automata. *Logical Methods in Computer Science*, 11(3), 2015. doi:10.2168/LMCS-11(3:14)2015.

18 Emmanuel Filiot, Nicolas Mazzocchi, and Jean-François Raskin. A Pattern Logic for Automata with Outputs. In *Developments in Language Theory - 22nd International Conference, DLT 2018, Tokyo, Japan, September 10-14, 2018, Proceedings*, volume 11088 of *Lecture Notes in Computer Science*, pages 304–317. Springer, 2018.

19 Emmanuel Filiot and Pierre-Alain Reynier. Transducers, logic and algebra for functions of finite words. *SIGLOG News*, 3(3):4–19, 2016. URL: https://dl.acm.org/citation.cfm?id=2984453.

20 N. J. Fine and H. S. Wilf. Uniqueness theorems for periodic functions. *Proceedings of the American Mathematical Society*, 16:109–114, 1965. doi:10.2307/2034009.

21 Eitan M. Gurari. The Equivalence Problem for Deterministic Two-Way Sequential Transducers is Decidable. *SIAM J. Comput.*, 11(3):448–452, 1982. doi:10.1137/0211035.

22 Daniel Kirsten and Ina Mäurer. On the Determinization of Weighted Automata. *Journal of Automata, Languages and Combinatorics*, 10(2/3):287–312, 2005. doi:10.25596/jalc-2005-287.

23 Sylvain Lombardy and Jacques Sakarovitch. Sequential? *Theor. Comput. Sci.*, 356(1-2):224–244, 2006. doi:10.1016/j.tcs.2006.01.028.

24 M. Lothaire. *Algebraic Combinatorics on Words*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2002. doi:10.1017/CBO9781107326019.

25 Dana S. Scott. Some Definitional Suggestions for Automata Theory. *J. Comput. Syst. Sci.*, 1(2):187–212, 1967. doi:10.1016/S0022-0000(67)80014-X.

26 Helmut Seidl. When Is a Functional Tree Transduction Deterministic? In *TAPSOFT'93: Theory and Practice of Software Development, International Joint Conference CAAP/FASE, Orsay, France, April 13-17, 1993, Proceedings*, volume 668 of *Lecture Notes in Computer Science*, pages 251–265. Springer, 1993.

27 Andreas Weber and Reinhard Klemm. Economy of Description for Single-Valued Transducers. *Inf. Comput.*, 118(2):327–340, 1995. doi:10.1006/inco.1995.1071.

The Parametric Complexity of Lossy Counter Machines

Sylvain Schmitz
LSV, ENS Paris Saclay & CNRS, Université Paris-Saclay, France
IUF, France
sylvain.schmitz@lsv.fr

Abstract

The reachability problem in lossy counter machines is the best-known ACKERMANN-complete problem and has been used to establish most of the ACKERMANN-hardness statements in the literature. This hides however a complexity gap when the number of counters is fixed. We close this gap and prove F_d-completeness for machines with d counters, which provides the first known uncontrived problems complete for the fast-growing complexity classes at levels $3 < d < \omega$. We develop for this an approach through antichain factorisations of bad sequences and analysing the length of controlled antichains.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases Counter machine, well-structured system, well-quasi-order, antichain, fast-growing complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.129

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version Full version available from `https://hal.archives-ouvertes.fr/hal-02020728`.

Funding ANR-14-CE28-0005 PRODAQ, ANR-17-CE40-0028 BRAVAS

Acknowledgements I thank Philippe Schnoebelen for his comments on a preliminary draft.

1 Introduction

Mayr and Meyer exhibited in 1981 "the first uncontrived decidable problems which are not primitive-recursive," namely the finite containment and equality problems in Petri nets [31]. McAloon [33], Clote [8], and Howell and Yen [23] subsequently proved Ackermannian upper bounds for these two problems, essentially matching Mayr and Meyer's lower bound.

Such an astronomical complexity could have been an isolated phenomenon with only a few examples related to the original problems, but uncontrived problems with a similar complexity actually occur in logic (e.g., relevance logic [47], data logics [11, 15], interval temporal logic [35], linear logic [28], metric temporal logic [27]), verification (e.g., counter machines [43, 22], fragments of the π-calculus [4], broadcast protocols [42], rewriting systems [20], register automata [11, 17]), and games (e.g., partial observation energy games [36], bisimulation games on pushdown automata [24]), and even higher complexities also occur naturally [29, 6, 37, 19, 18, 10]; see [38, Sec. 6] for an overview.

This abundance of results is largely thanks to a framework [41, 42, 39] that comprises:

- The definition of an ordinal-indexed hierarchy $(\mathsf{F}_\alpha)_\alpha$ of *fast-growing complexity classes*, along with assorted notions of reductions and completeness suitable to work with such high complexities [38]. The previous decision problems are complete for $\mathsf{ACKERMANN} = \mathsf{F}_\omega$ under primitive-recursive reductions; F_ω is the lowest non primitive-recursive class in the hierarchy, where $\mathsf{TOWER} = \mathsf{F}_3$ corresponds to problems solvable in time bounded by a tower of exponentials and where each F_k for a finite k is primitive-recursive.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 129; pp. 129:1–129:15


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

- The identification of *master* decision problems, which allow to establish completeness more easily than from first principles. For instance, reachability in lossy counter machines [47, 43, 45] plays a similar role for ACKERMANN as, e.g., 3SAT for NP or QBF for PSPACE, and has been used to derive most of the known ACKERMANN-hardness results [4, 11, 15, 35, 27, 22, 20, 17, 24].
- Lower bound techniques for establishing the complexity of such master problems: this typically relies on implementing *weak computers* for *Hardy functions* and their inverses in the formalism at hand, allowing to build a large but bounded working space on which a Turing or a Minsky machine can then be simulated [47, 43, 45, 6, 37, 19, 18].
- Upper bound techniques relying on combinatorial statements, called *length function theorems*, on the length of *controlled bad sequences* over well-quasi-orders, which are used to prove the termination of the decision procedures [33, 8, 49, 7, 14, 40, 3, 37].

From an algorithmic perspective, these results are negative and one could qualify such problems as merely "not undecidable". What we gain however are insights into the computational power of the models, allowing to compare them and to identify the main sources of complexity – e.g., in lossy counter machines, the key parameter is the number of counters. Furthermore, from a modelling perspective, a formalism with a tremendous computational power that nevertheless falls short of Turing completeness can be quite satisfactory.

Contributions. In this paper, we revisit the proof of the best-known result in this area, namely the ACKERMANN-completeness of reachability in lossy counter machines (LCMs). Those are simply multi-counter Minsky machines with a *lossy semantics* that allows the counters to decrease in an uncontrollable manner during executions; see Section 2.

The gap in the current state of knowledge appears when one fixes the key complexity parameter, i.e., the number d of counters. Indeed, the best known lower bound for LCM reachability is F_d-hardness when $d \geq 3$ [39, Thm. 4.9], but the best known upper bound is F_{d+1} [14, 41, 3]. This complexity gap reveals a serious shortcoming of the framework advertised earlier in this introduction, and also impacts the complexity of many problems shown hard through a reduction from LCM reachability.

Our first main contribution in Proposition 8 is an F_d upper bound, which together with the lower bound from [39, Thm. 4.9] entails the following completeness result.

▶ **Theorem 1.** *LCM Reachability is F_ω-complete, and F_d-complete if the number $d \geq 3$ of counters is fixed.*

Note that this provides an uncontrived decision problem for every class F_k with $3 \leq k \leq \omega$, whereas no natural F_k-complete problems were previously known for the intermediate primitive-recursive levels strictly between TOWER and ACKERMANN, i.e., for $3 < k < \omega$.

As we recall in Section 3, reachability in lossy counter machines can be solved using the generic *backward coverability algorithm* for *well-structured* systems [1, 16]. As usual, we derive our complexity upper bound by bounding the length of the bad sequences that underlie the termination argument for this algorithm. The main obstacle here is that the length function theorems in [14, 41, 3] – i.e., the bounds on the length of controlled bad sequences over $\mathbb{N}^d$ – are essentially optimal and only yield an F_{d+1} complexity upper bound.

We circumvent this using a new approach in Section 4. We restrict our attention to *strongly controlled* bad sequences rather than the more general *amortised controlled* ones (see Section 4.1), which in turn allows us to work on the *antichain factorisations* of bad sequences (see Section 4.2). This entails that, in order to bound the length of strongly controlled

bad sequences, it suffices to bound the length of strongly controlled antichains. This is tackled in Section 5, where we prove a *width function theorem* on the length of controlled antichains over $\mathbb{N}^d$; to the best of our knowledge, this is the first statement of this kind specific to antichains rather than bad sequences. We wrap up with the proof of Proposition 8 in Section 6.

The developments of Sections 4 and 5 form our second main contribution. They are of wide interest beyond lossy counter machines, as they can be applied whenever the termination of an algorithm relies on $\mathbb{N}^d$ having finite (controlled) bad sequences or antichains.

2 Lossy Counter Machines

Syntax. A *lossy counter machine* (LCM) [32] is syntactically identical to a Minsky machine $M = (Q, \mathtt{C}, \delta)$, where the transitions in $\delta \subseteq Q \times \mathtt{C} \times \{\mathtt{=0?}, \mathtt{++}, \mathtt{--}\} \times Q$ operate on a finite set Q of control locations and a finite set $\mathtt{C}$ of counters through *zero-tests* `c=0?`, *increments* `c++` and *decrements* `c--`.

Operational Semantics. The semantics of an LCM differ from the usual, 'reliable' semantics of a counter machine in that the counter values can decrease in an uncontrolled manner at any point of the execution. Formally, a *configuration* $q(\boldsymbol{v})$ associates a control location q in Q with a counter valuation $\boldsymbol{v}$ in $\mathbb{N}^{\mathtt{C}}$, i.e. counter values can never go negative. The set of configurations $Q \times \mathbb{N}^{\mathtt{C}}$ is ordered by the product ordering: $q(\boldsymbol{v}) \leq q'(\boldsymbol{v}')$ if $q = q'$ and $\boldsymbol{v}(\mathtt{c}) \leq \boldsymbol{v}'(\mathtt{c})$ for all $\mathtt{c} \in \mathtt{C}$.

A transition of the form $(q, \mathtt{c}, \mathtt{op}, q') \in \delta$ defines a set of *reliable computation steps* $q(\boldsymbol{v}) \to q'(\boldsymbol{v}')$, where $\boldsymbol{v}(\mathtt{c}') = \boldsymbol{v}'(\mathtt{c}')$ for all $\mathtt{c} \neq \mathtt{c}'$ in $\mathtt{C}$ and

- if $\mathtt{op} = \mathtt{=0?}$, then $\boldsymbol{v}(\mathtt{c}) = \boldsymbol{v}'(\mathtt{c}) = 0$,
- if $\mathtt{op} = \mathtt{++}$, then $\boldsymbol{v}(\mathtt{c}) + 1 = \boldsymbol{v}'(\mathtt{c})$, and
- if $\mathtt{op} = \mathtt{--}$, then $\boldsymbol{v}(\mathtt{c}) = \boldsymbol{v}'(\mathtt{c}) + 1$.

A *lossy computation step* is then defined by allowing counter values to decrease arbitrarily between reliable steps: $q(\boldsymbol{v}) \to_\ell q'(\boldsymbol{v}')$ if there exist $\boldsymbol{w} \leq \boldsymbol{v}$ and $\boldsymbol{w}' \geq \boldsymbol{v}'$ such that $q(\boldsymbol{w}) \to q'(\boldsymbol{w}')$. We write as usual $\to_\ell^*$ for the transitive reflexive closure of $\to_\ell$.

Reachability. The decision problem we tackle in this paper is the following.

▶ **Problem** (LCM Reachability).
instance A lossy counter machine $(Q, \mathtt{C}, \delta)$ and two configurations $q_0(\boldsymbol{v}_0)$ and $q_f(\boldsymbol{v}_f)$.
question Is $q_f(\boldsymbol{v}_f)$ reachable from $q_0(\boldsymbol{v}_0)$, i.e., does $q_0(\boldsymbol{v}_0) \to_\ell^* q_f(\boldsymbol{v}_f)$?

Note that, due to the lossy semantics, this is equivalent to the *coverability* problem, which asks instead whether there exists $\boldsymbol{v} \geq \boldsymbol{v}_f$ such that $q_0(\boldsymbol{v}_0) \to_\ell^* q_f(\boldsymbol{v})$. Indeed, such a $\boldsymbol{v}$ exists if and only if $q_0(\boldsymbol{v}_0) \geq q_f(\boldsymbol{v}_f)$ or $q_0(\boldsymbol{v}_0) \to_\ell^* q_f(\boldsymbol{v}_f)$.

While many problems are undecidable in LCMs [32, 44], these systems are in fact *well-structured* in the sense of [1, 16], which means that their coverability problem is decidable, as further discussed in Section 3. The ACKERMANN-hardness of reachability was first shown by Schnoebelen [43] in 2002,[1] while an ACKERMANN upper bound follows from the length function theorems for Dickson's Lemma [33, 8, 14, 41, 3]. Note that LCM reachability is equivalent to reachability in counter machines with *incrementing errors* [11] and to coverability in *reset counter machines* [45, Sec. 6], and this also holds if we fix the number of counters.

[1] Urquhart [47] showed independently in 1999 and using a similar approach the same result for the closely related model of alternating expansive vector addition systems.

3 Well Structured Systems

Well-structured transition systems (WSTS) [1, 16] form a family of computational models where the (usually infinite) set of configurations is equipped with a well-quasi-ordering (see Section 3.1) that is "compatible" with the computation steps (see Section 3.2). The existence of this well-quasi-ordering allows for the decidability of some important behavioural properties like termination (from a given initial configuration) or coverability, see Section 3.3.

3.1 Well-Quasi-Orders

A *quasi-order* (qo) is a pair $(X, \leq)$ where $\leq \subseteq X \times X$ is transitive and reflexive; we write $x < y$ for the associated strict ordering, when $x \leq y$ and $y \not\leq x$, $x \perp y$ for incomparable elements, when $x \not\leq y$ and $y \not\leq x$, and $x \equiv y$ for equivalent elements, when $x \leq y$ and $y \leq x$. The *upward-closure* $\uparrow Y$ of some $Y \subseteq X$ is defined as $\uparrow Y \stackrel{\text{def}}{=} \{x \in X \mid \exists y \in Y \,.\, x \geq y\}$; we write $\uparrow x$ instead of $\uparrow\{x\}$ for singletons and say that a set $U \subseteq X$ is *upwards-closed* when $U = \uparrow U$. We call a finite or infinite sequence $x_0, x_1, x_2, \ldots$ over X *bad* if for all indices $i < j$, $x_i \not\leq x_j$; if $x_i \perp x_j$ for all $i < j$, then $x_0, x_1, x_2, \ldots$ is an *antichain*.

A *well-quasi-order* (wqo) [21, 26] is a qo $(X, \leq)$ where bad sequences are finite. Equivalently, $(X, \leq)$ is a wqo if and only if it is both *well-founded*, i.e., there does not exist any infinite decreasing sequences $x_0 > x_1 > x_2 > \cdots$ of elements in X, and has the *finite antichain condition*, i.e., there are no infinite antichains. Still equivalently, $(X, \leq)$ is a wqo if and only if it has the *ascending chain condition*: any increasing sequence $U_0 \subseteq U_1 \subseteq U_2 \subseteq \cdots$ of upwards-closed subsets of X eventually stabilises, i.e., $\bigcup_{i\in\mathbb{N}} U_i = U_k = U_{k+1} = U_{k+2} = \cdots$ for some k. Still equivalently, $(X, \leq)$ is a wqo if and only if it has the *finite basis property*: any non-empty subset contains at least one, and at most finitely many minimal elements (up to equivalence); thus if $U \subseteq X$ is upwards-closed, then $\min U$ is finite and $U = \uparrow(\min U)$.

For a basic example, consider any finite set Q along with the equality relation, which is a wqo $(Q, =)$ by the pigeonhole principle. Any well-order is a wqo, thus the set of natural numbers and any of its initial segments $[k] \stackrel{\text{def}}{=} \{0, \ldots, k-1\}$ along with their natural ordering are also wqos. More examples can be constructed using algebraic operations: for instance, if $(X_0, \leq_{X_0})$ and $(X_1, \leq_{X_1})$ are wqos, then so are:

- their disjoint sum $(X_0 \sqcup X_1, \leq)$ where $X_0 \sqcup X_1 \stackrel{\text{def}}{=} \{(x, 0) \mid x \in X_0\} \cup \{(x, 1) \mid x \in X_1\}$ and $(x, i) \leq (y, j)$ if $i = j$ and $x \leq_{X_i} y$;
- their Cartesian product $(X_0 \times X_1, \leq)$ where $(x_0, x_1) \leq (y_0, y_1)$ if $x_i \leq_{X_i} y_i$ for all $0 \leq i \leq 1$; in the case of $(\mathbb{N}^d, \leq)$, this result is also known as Dickson's Lemma [12].

Note that the set of configurations $(Q \times \mathbb{N}^C, \leq)$ of an LCM is a wqo for the product ordering.

3.2 Compatibility

An *ordered transition system* $\mathcal{S} = (S, \to, \leq)$ combines a set S of configurations with a transition relation $\to \subseteq S \times S$ and a quasi-ordering $\leq$ of its configurations. An ordered transition system $\mathcal{S} = (S, \to, \leq)$ is *well-structured* if $(S, \leq)$ is a wqo and

$$\forall s_1, s_2, t_1 \in S,\ \big(s_1 \to s_2 \text{ and } s_1 \leq t_1\big) \text{ implies } \exists t_2 \in S,\ \big(t_1 \to t_2 \text{ and } s_2 \leq t_2\big)\,. \tag{1}$$

This property is also called *compatibility* (of the ordering with the transitions). Formally, it just means that $\leq$ is a *simulation* relation for $(S, \to)$, in precisely the classical sense of [34]. The point of (1) is to ensure that a larger configuration can do at least as much as a smaller configuration. For instance, lossy steps in a LCM are visibly compatible with $\leq$ according to (1), and thus the transition system $(Q \times \mathbb{N}^C, \to_\ell, \leq)$ defined by the lossy operational semantics of a LCM is a WSTS.

3.3 Coverability

We focus here on the *coverability problem*: given a WSTS $(S, \to, \leq)$ and two configurations $s, t \in S$, does s *cover* t, i.e., does there exist $t' \geq t$ such that $s \to^* t'$? The decidability of this problem uses a set-saturation method first introduced by Arnold and Latteux [5] for reset Petri nets, but the algorithm was independently rediscovered by Abdulla et al. [2] for lossy channel systems and its generic formulation was popularised in the surveys [1, 16].

Backward Coverability. The algorithm computes $\mathrm{Pre}^*_\exists(\uparrow t) \stackrel{\text{def}}{=} \{s' \in S \mid \exists t' \geq t,\, s' \to^* t'\}$, i.e., the set of configurations that cover t; it only remains to check whether $s \in \mathrm{Pre}^*_\exists(\uparrow t)$ in order to answer the coverability instance. More precisely, for a set of configurations $U \subseteq S$, let us define its (existential) *predecessor set* as $\mathrm{Pre}_\exists(U) \stackrel{\text{def}}{=} \{s \in S \mid \exists s' \in U,\, s \to s'\}$. The algorithm computes the limit of the sequence $U_0 \subseteq U_1 \subseteq \cdots$ defined by

$$U_0 \stackrel{\text{def}}{=} \uparrow t\,, \qquad\qquad U_{n+1} \stackrel{\text{def}}{=} U_n \cup \mathrm{Pre}_\exists(U_n)\,. \tag{2}$$

Note that for all n, $U_n = \{s' \in S \mid \exists t' \geq t,\, s' \to^{\leq n} t'\}$ is the set of configurations that cover t in at most n steps, and that we can stop this computation as soon as $U_{n+1} \subseteq U_n$.

There is no reason for the chain defined by (2) to stabilise in general ordered transition systems, but it does in the case of a WSTS. Indeed, $\mathrm{Pre}_\exists(U)$ is upwards-closed whenever $U \subseteq S$ is upwards-closed, thus the sequence defined by (2) stabilises to $\bigcup_{i\in\mathbb{N}} U_i = \mathrm{Pre}^*_\exists(\uparrow t)$ after a finite amount of time thanks to the ascending chain condition. Moreover, the finite basis property ensures that all the sets U_i can be finitely represented using their minimal elements, and the union or inclusion of two upwards-closed sets can be computed on this representation. The last ingredients are two effectiveness assumptions:

- $(S, \leq)$ should be effective, meaning that S is recursive and the ordering $\leq$ is decidable,
- there exists an algorithm returning the set of minimal predecessors $\min \mathrm{Pre}_\exists(\uparrow s')$ of any given configuration s'; this is known as the *effective pred-basis* assumption.

These two assumptions hold in LCMs: $(Q \times \mathbb{N}^{\mathtt{C}}, \leq)$ is certainly effective, and the minimal predecessors of a configuration $q'(\boldsymbol{v}')$ can be computed by

$$\min \mathrm{Pre}_\exists(\uparrow q'(\boldsymbol{v}')) = \min\{q(\mathrm{pre}_{\mathtt{c\,op}}(\boldsymbol{v}')) \mid (q, \mathtt{c}, \mathtt{op}, q') \in \delta \text{ and } \boldsymbol{v}'(\mathtt{c}) = 0 \text{ if } \mathtt{op} = \mathtt{=0?}\} \tag{3}$$

where $\mathrm{pre}_{\mathtt{c\,op}}(\boldsymbol{v}')$ is a vector in $\mathbb{N}^{\mathtt{C}}$ defined by $\mathrm{pre}_{\mathtt{c\,op}}(\boldsymbol{v}')(\mathtt{c}') \stackrel{\text{def}}{=} \boldsymbol{v}'(\mathtt{c}')$ for all $\mathtt{c}' \neq \mathtt{c}$ in $\mathtt{C}$ and

$$\mathrm{pre}_{\mathtt{c=0?}}(\boldsymbol{v}')(\mathtt{c}) \stackrel{\text{def}}{=} 0\,, \quad \mathrm{pre}_{\mathtt{c++}}(\boldsymbol{v}')(\mathtt{c}) \stackrel{\text{def}}{=} \max\{0, \boldsymbol{v}'(\mathtt{c}) - 1\}\,, \quad \mathrm{pre}_{\mathtt{c--}}(\boldsymbol{v}')(\mathtt{c}) \stackrel{\text{def}}{=} \boldsymbol{v}'(\mathtt{c}) + 1\,. \tag{4}$$

Coverability Pseudo-Witnesses. Let us reformulate the termination argument of the backward coverability algorithm in terms of bad sequences. We can extract a sequence of elements $t_0, t_1, \ldots$ from the ascending sequence $U_0 \subsetneq U_1 \subsetneq \cdots$ defined by (2) before saturation: $t_0 \stackrel{\text{def}}{=} t$ and $t_{i+1} \in U_{i+1} \setminus U_i$ for all i. Note that if $i < j$, then $t_j \in U_j \setminus U_i$ and therefore $t_i \not\leq t_j$: the sequence $t_0, t_1, \ldots$ is bad and therefore finite. In fact, we can even pick t_{i+1} at each step among the minimal elements of $\mathrm{Pre}_\exists(\uparrow t_i)$; we call such a bad sequence $t_0, t_1, \ldots, t_n$ with

$$t_0 \stackrel{\text{def}}{=} t\,, \qquad\qquad t_{i+1} \in \min \mathrm{Pre}_\exists(\uparrow t_i) \setminus U_i\,. \tag{5}$$

a *pseudo-witness* of the coverability of t. The maximal length of pseudo-witnesses is therefore equal to the number of steps of the backward coverability algorithm, and this is what we will bound in the upcoming Sections 4 and 5.

4 Controlled Bad Sequences and Antichains

As we have just discussed, the running time of the backward coverability algorithm is essentially bounded by the length of the bad sequences constructed by its termination argument. Though bad sequences over a wqo are finite, we cannot bound their lengths in general; e.g., $(0, n+1), (0, n), \ldots, (0, 0)$ and $(1, 0), (0, n), (0, n-1), \ldots, (0, 1), (0, 0)$ are bad sequences over $\mathbb{N}^2$ of length $n + 2$ for all n. Nevertheless, a bad sequence produced by an algorithm like the backward coverability algorithm of Section 3.3 is not arbitrary, because its elements are determined by the algorithm's input and the complexity of its operations. We capture this intuition formally through *controlled sequences*.

4.1 Controlling Sequences

Norms. Given a wqo $(X, \leq_X)$, we posit a *norm* function $|\,.\,|_X : X \to \mathbb{N}$; if $x \leq_X x'$ implies $|x|_X \leq |x'|_X$, we call this norm *monotone*. In order to be able to derive combinatorial statements, we require $X_{\leq n} \stackrel{\text{def}}{=} \{x \in X \mid |x|_X \leq n\}$ to be finite for every n; we call the resulting structure $(X, \leq_X, |\cdot|_X)$ a *normed wqo* (nqo).

We will use the following monotone norms on the wqos we defined in Section 3.1: over a finite Q, all the elements have the same norm 0; over $\mathbb{N}$ or $[d]$, n has norm $|n|_{\mathbb{N}} = |n|_{[d]} = n$; over disjoint sums $X_0 \sqcup X_1$, (x, i) uses the norm $|x|_{X_i}$ of its underlying set; finally, over Cartesian products $X \times Y$, (x, y) uses the infinite norm $\max(|x|_X, |y|_Y)$.

Controls. Let $n_0 \in \mathbb{N}$ and let $g : \mathbb{N} \to \mathbb{N}$ be a monotone and inflationary function, i.e., for all $x \leq x'$, $g(x) \leq g(x')$ and $x \leq g(x)$. We say that a sequence $x_0, x_1, x_2, \ldots$ of elements in X is *amortised* (g, n_0)*-controlled* if

$$\forall i \,.\, |x_i|_X \leq g^i(n_0) \,, \tag{6}$$

where g^i denotes the ith iterate of g. We say that it is *strongly* (g, n_0)*-controlled* if

$$|x_0|_X \leq n_0 \quad \text{and} \quad \forall i \,.\, |x_{i+1}|_X \leq g(|x_i|_X) \,. \tag{7}$$

By definition, a strongly controlled sequence is also amortised controlled: $|x_0|_X \leq g^0(n_0) = n_0$, which prompts the name of *initial norm* for n_0, and amortised steps cannot grow faster than g the *control function*.

Previous works like [14, 41, 3] focused on the more general amortised controlled sequences, but strong controlled ones are actually more relevant in practice. For instance, in the case of LCM coverability, the computation of minimal predecessors in (3–4) shows that the pseudo-witnesses from (5) of the coverability of a target configuration $q_f(\boldsymbol{v}_f)$ are strongly $(H, |\boldsymbol{v}_f|)$-controlled by the initial norm $|\boldsymbol{v}_f| = \max_{\mathsf{c} \in \mathsf{C}} \boldsymbol{v}_f(\mathsf{c})$ and the control function $H(x) \stackrel{\text{def}}{=} x + 1$.

Length, Norm, and Width Functions. The point of controlled sequences is that their length can be bounded. Consider for this the tree obtained by sharing the common prefixes of all the strongly (g, n_0)-controlled bad sequences over a normed wqo $(X, \leq_X, |\,.\,|_X)$.

- This tree is finitely branching by (7) – its branching degree is bounded by the cardinal of $X_{\leq g^i(n_0)}$ for a node at depth i – , and
- it has no infinite branches since bad sequences over $(X, \leq_X)$ are finite.

By Kőnig's Lemma, this tree of bad sequences is therefore finite, of some height $L^{\mathsf{s}}_{g,X}(n_0)$ representing the length of the maximal strongly (g, n_0)-controlled bad sequence(s) over X, and we also let $N^{\mathsf{s}}_{g,X}(n_0)$ bound the norms encountered along such sequences; note that

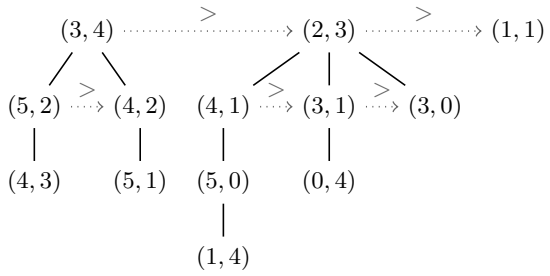


Figure 1 The antichain factorisation of the strongly $(H, 4)$-controlled bad sequence $(3,4)$ $(5,2)$ $(4,3)$ $(4,2)$ $(5,1)$ $(2,3)$ $(4,1)$ $(5,0)$ $(1,4)$ $(3,1)$ $(0,4)$ $(3,0)$ $(1,1)$ over $\mathbb{N}^2$ for $H(x) \stackrel{\text{def}}{=} x + 1$.

$N^{\mathsf{s}}_{g,X}(n_0) \leq g^{L^{\mathsf{s}}_{g,X}(n_0)}(n_0)$ since g is monotone inflationary. Similarly, there exists a maximal length, denoted by $W^{\mathsf{a}}_{g,X}(n_0)$ (resp. $W^{\mathsf{s}}_{g,X}(n_0)$), for amortised (resp. strongly) (g, n_0)-controlled antichains over X. We call $L^{\mathsf{s}}_{g,X}$, $N^{\mathsf{s}}_{g,X}$, and $W^{\mathsf{s}}_{g,X}$ the *strong length*, *strong norm*, and *strong width* functions respectively, and $W^{\mathsf{a}}_{g,X}$ the *width* function. By definition, a strongly controlled antichain is amortised controlled, i.e.,

$$W^{\mathsf{s}}_{g,X}(n_0) \leq W^{\mathsf{a}}_{g,X}(n_0)\,, \tag{8}$$

and the length of a bad sequence where all the elements are of norm at most $N = N^{\mathsf{s}}_{g,X}(n_0)$ is bounded by the cardinal of $X_{\leq N}$, i.e.,

$$L^{\mathsf{s}}_{g,X}(n_0) \leq |X_{\leq N^{\mathsf{s}}_{g,X}(n_0)}|\,. \tag{9}$$

Observe that $L^{\mathsf{s}}_{g,X}(|t|_X)$ bounds the number of steps required by the backward coverability algorithm for a WSTS over $(X, \leq_X, |\,.\,|_X)$ with target configuration t where $s' \in \min \mathrm{Pre}(\uparrow t')$ implies $|s'|_X \leq g(|t'|_X)$. In the case of LCMs, we are therefore interested in $L^{\mathsf{s}}_{H,Q\times N^{\mathsf{c}}}(|\boldsymbol{v}_f|)$.

4.2 Antichain Factorisations

Let $(X, \leq_X, |\,.\,|_X)$ be a nwqo where $|\,.\,|_X$ is monotone – i.e., $x \leq x'$ implies $|x|_X \leq |x'|_X$ –, and let $x_0, x_1, \ldots, x_{\ell-1} \in X^*$ be a strongly (g, n_0)-controlled bad sequence over $(X, \leq_X, |\,.\,|_X)$. Informally, the *antichain factorisation* of $x_0, x_1, \ldots, x_{\ell-1}$ is an ordered forest A where all the branches are strongly (g, n_0)-controlled antichains, siblings are ordered left-to-right by the strict ordering $>_X$, and such that the pre-order traversal of A yields back the bad sequence. Consider for instance the example of Figure 1: this bad sequence has length 13, thus the norm of its elements is at most $H^{12}(4) = 16$, but because the height of its antichain factorisation is 4, we can actually bound the norm by $H^3(4) = 7$.

We can compute this factorisation from any strongly (g, n_0)-controlled bad sequence. Formally, $A \subseteq X^*$ is a prefix-closed finite set of antichains with the prefix ordering as vertical ordering. Two antichains u and v in A are *siblings* if $u = w \cdot x$ and $v = w \cdot y$ for some $w \in X^*$ and $x, y \in X$, and we order such siblings by letting $u >_X v$ if $x >_X y$. Given $x_0, \ldots, x_{\ell-1}$, we let $A \stackrel{\text{def}}{=} \mathrm{Fact}(x_0, 1)$ where

$$\mathrm{Fact}(y_0 \cdots y_m, i) \stackrel{\text{def}}{=} \{y_0 \cdots y_m\} \cup \begin{cases} \emptyset & \text{if } i = \ell\,, \\ \mathrm{Fact}(y_0 \cdots y_m x_i, i+1) & \text{if } \forall j\,.\, y_j \not>_X x_i\,, \\ \mathrm{Fact}(y_0 \cdots y_{k-1} x_i, i+1) & \text{if } k = \min\{j \mid y_j >_X x_i\}\,. \end{cases} \tag{10}$$

This corresponds to scanning the elements x_i of the bad sequence from left to right while building the current "rightmost branch" $y_0 \cdots y_m \in A$, which is (by induction on i) a strongly (g, n_0)-controlled antichain and a scattered subword of $x_0 \cdots x_{i-1}$ (thus $y_j \not\leq x_i$ for all $0 \leq j \leq m$) such that $y_m = x_{i-1}$. If $y_j \not>_X x_i$ for all $0 \leq j \leq m$, then $y_0 \cdots y_m x_i$ is a (g, n_0)-controlled antichain and a scattered subword of $x_0 \cdots x_i$. If otherwise $y_j >_X x_i$ for some y_j, we let k be the minimal such j and we start a new rightmost branch with x_i out of y_{k-1} (x_i is possibly a new root if $k = 0$); crucially, because $|.|_X$ is monotone and $x_i <_X y_k$, we have $|x_i|_X \leq |y_k|_X \leq g(|y_{k-1}|_X)$ (or $|x_i|_X \leq |y_k|_X \leq n_0$ at the root), thus $y_0 \cdots y_{k-1} x_i$ is again a strongly (g, n_0)-controlled antichain and a scattered subword of $x_0 \cdots x_i$.

We deduce a bound on the strong norm function in terms of the strong width function.

▶ **Lemma 2** (Antichain Factorisation). *Let $(X, \leq_X, |.|_X)$ be a normed wqo with $|.|_X$ monotone, n_0 in $\mathbb{N}$, and $g : \mathbb{N} \to \mathbb{N}$ monotone inflationary. Then $N^{\mathsf{s}}_{g,X}(n_0) \leq g^{W^{\mathsf{s}}_{g,X}(n_0)}(n_0)$.*

Lemma 2 combined with (8) shows that the strong norm function $N^{\mathsf{s}}_{g,X}$ can be bounded in terms of the width function $W^{\mathsf{a}}_{g,X}$. By (9), this will also yield a bound on the strong length function $L^{\mathsf{s}}_{g,X}$. We focus therefore on the width function in the upcoming Section 5.

5 Width Function Theorem

As seen in Section 4, by suitably controlling how large the elements can grow in antichains, we can derive upper bounds on the time and space required by the backward coverability algorithm of Section 3. We prove in this section a *width function theorem*, a combinatorial statement on the length of amortised controlled antichains over tuples of natural numbers, which will allow to derive a complexity upper bound for reachability in lossy counter machines.

The high complexities at play here require the use of ordinal-indexed *subrecursive functions* in order to denote non-elementary growths. We first recall the definitions of two families of such functions in Section 5.1; we refer the reader to [46, 41] for further details. We then prove in Section 5.2 a bound on the width function $W^{\mathsf{a}}_{g,\mathbb{N}^d}$ using the framework of [40, 41].

5.1 Subrecursive Hierarchies

Fundamental Sequences and Predecessors. A *fundamental sequence* for a limit ordinal λ is a strictly increasing sequence $(\lambda(x))_{x<\omega}$ of ordinal terms with supremum λ. We use the standard assignment of fundamental sequences to limit ordinals below ε_0 in Cantor normal form, defined inductively by

$$(\gamma + \omega^{\beta+1})(x) \stackrel{\text{def}}{=} \gamma + \omega^{\beta} \cdot (x+1) \,, \qquad (\gamma + \omega^{\lambda})(x) \stackrel{\text{def}}{=} \gamma + \omega^{\lambda(x)} \,. \tag{11}$$

This particular assignment satisfies e.g. $0 < \lambda(x) < \lambda(y)$ for all $x < y$. For instance, $\omega(x) = x + 1$, $(\omega^{\omega^4} + \omega^{\omega^3+\omega^2})(x) = \omega^{\omega^4} + \omega^{\omega^3+\omega\cdot(x+1)}$.

The *predecessor* $P_x(\alpha)$ of an ordinal term $0 < \alpha < \varepsilon_0$ at $x \in \mathbb{N}$ is defined inductively by

$$P_x(\alpha + 1) \stackrel{\text{def}}{=} \alpha \,, \qquad P_x(\lambda) \stackrel{\text{def}}{=} P_x(\lambda(x)) \,. \tag{12}$$

In essence, the predecessor of an ordinal is obtained by repeatedly taking the xth element in the fundamental sequence of limit ordinals, until we finally reach a successor ordinal and may remove 1. For instance, $P_x(\omega^2) = P_x(\omega \cdot (x+1)) = P_x(\omega \cdot x + x + 1) = \omega \cdot x + x$.

Hardy and Cichoń Hierarchies. Let $h\colon \mathbb{N} \to \mathbb{N}$ be a function. The *Hardy hierarchy* $(h^\alpha)_{\alpha\in\varepsilon_0}$ and the *Cichoń hierarchy* $(h_\alpha)_{\alpha\in\varepsilon_0}$ relative to h are defined for all $0 < \alpha < \varepsilon_0$ by

$$h^0(x) \stackrel{\text{def}}{=} x\,, \quad h^\alpha(x) \stackrel{\text{def}}{=} h^{P_x(\alpha)}(h(x))\,, \quad h_0(x) \stackrel{\text{def}}{=} 0\,, \quad h_\alpha(x) \stackrel{\text{def}}{=} 1 + h_{P_x(\alpha)}(h(x))\,. \tag{13}$$

Observe that $h^k(x) = h^{P_x(k)}(h(x)) = h^{k-1}(h(x))$ for some finite k is the kth iterate of h. This intuition carries over: h^α is a "transfinite" iteration of the function h, using diagonalisation in the fundamental sequences to handle limit ordinals. A standard choice for the function h is the successor function, noted $H(x) \stackrel{\text{def}}{=} x + 1$; in that case, we see that a first diagonalisation yields $H^\omega(x) = H^x(x+1) = 2x+1$. The next diagonalisation occurs at $H^{\omega\cdot 2}(x) = H^{\omega+x}(x+1) = H^\omega(2x+1) = 4x+3$. Fast-forwarding a bit, we get for instance a function of exponential growth $H^{\omega^2}(x) = 2^{x+1}(x+1) - 1$, and later a non-elementary function $H^{\omega^3}(x)$ akin to a tower of exponentials of height x, and an "Ackermannian" non primitive-recursive function H^{ω^ω}.

Both h^α and h_α are monotone and inflationary whenever h is monotone inflationary. Hardy functions are well-suited for expressing large iterates of a control function, and therefore for bounding the norms of elements in a controlled sequence. Cichoń functions are well-suited for expressing the length of controlled sequences: we can compute how many times we should iterate h in order to compute $h^\alpha(x)$ using the corresponding Cichoń function [7]:

$$h^\alpha(x) = h^{h_\alpha(x)}(x)\,. \tag{14}$$

5.2 Width Function for Dickson's Lemma

The starting point for our analysis is a *descent equation* for amortised controlled antichains through *residuals*, similar to the equations proven in [14, 40] for bad sequences (see Lemma 3). The key idea introduced in [14] is then to over-approximate the residuals of $\mathbb{N}^d$ by working over *polynomial nwqos*, where disjoint sums are also allowed. Then, the notion of "over-approximation" of residuals of polynomial nwqos is captured formally by showing the existence of a *normed reflection* into another polynomial nwqo. The final step lifts this to ordinals, allowing to relate $W^{\mathsf{a}}_{g,X}$ for a polynomial nwqo X to functions in the Cichoń hierarchy.

Strict Polynomial Normed wqos. Let us write $X \cdot p$ for $\overbrace{X \sqcup \cdots \sqcup X}^{p \text{ times}}$, X^d for $\overbrace{X \times \cdots \times X}^{d \text{ times}}$, and $\mathbf{0}$ for the empty nwqo. We call a nwqo of the form $\mathbb{N}^{d_1} \sqcup \cdots \sqcup \mathbb{N}^{d_m}$ for some $m \geq 0$ and $d_1, \cdots, d_m \geq 1$ a *strict polynomial nwqo*. The set of configurations $Q \times \mathbb{N}^{\mathtt{C}}$ of an LCM with $|Q| = q$ locations and $|\mathtt{C}| = d \geq 1$ counters, along with its ordering and infinite norm, is isomorphic to the strict polynomial nwqo $\mathbb{N}^d \cdot q$.

Residuals and a Descent Equation. Let $(X, \leq_X, |.|_X)$ be a normed wqo and x be an element of X. We write $X_{\perp x} \stackrel{\text{def}}{=} \{y \in X \mid x \perp y\}$ for the *residual* of X in x. By the finite antichain condition, there cannot be infinite sequences of residuations $(\cdots((X_{\perp x_0})_{\perp x_1})_{\perp x_2}\cdots)_{\perp x_i}$ because $x_i \perp x_j$ for all $i < j$ and it would create an infinite antichain.

Consider now an amortised (g, n_0)-controlled antichain $x_0, x_1, x_2, \ldots$ over X. Assuming the sequence is not empty, then for all $i > 0$, $x_0 \perp x_i$, i.e. the suffix $x_1, x_2, \ldots$ is actually an antichain over $X_{\perp x_0}$. This suffix is now amortised $(g, g(n_0))$-controlled, and thus of length bounded by $W^{\mathsf{a}}_{g,X_{\perp x_0}}(g(n_0))$. This yields the following *descent equation* when considering all the possible amortised (g, n_0)-controlled antichains; see the full version for a proof.

▶ **Lemma 3.** *Let $(X, \leq_X, |\,.\,|_X)$ be a nwqo, $n_0 \in \mathbb{N}$ and $g\colon \mathbb{N} \to \mathbb{N}$. Then $W^{\mathsf{a}}_{g,X}(n_0) = \max_{x\in X_{\leq n_0}} 1 + W^{\mathsf{a}}_{g,X_{\perp x}}(g(n_0))$.*

Reflecting Normed wqos. The descent equation, though it offers a way of computing the width function, quickly leads to complex residual expressions. We are going to over-approximate these $X_{\perp x}$'s using *nwqo reflections*, so that the computation can be carried out without leaving the realm of strict polynomial nwqos, leading to an *inductive* over-approximation of $X_{\perp x}$ over the structure of the strict polynomial nwqo X.

A *nwqo reflection* [40] is a mapping $r\colon X \to Y$ between two nwqos $(X, \leq_X, |\,.\,|_X)$ and $(Y, \leq_Y, |\,.\,|_Y)$ that satisfies the two following properties:

$$\forall x, x' \in X. \qquad r(x) \leq_Y r(x') \quad \text{implies} \quad x \leq_X x' \,, \tag{15}$$

$$\forall x \in X. \qquad |r(x)|_Y \leq |x|_X \,. \tag{16}$$

In other words, a nwqo reflection is an order reflection that is not norm-increasing. This induces a quasi-ordering between nwqos, written $X \hookrightarrow Y$. Remark that reflections are compatible with disjoint sums and products [40, Prop. 3.5]:

$$X_0 \hookrightarrow Y_0 \text{ and } X_1 \hookrightarrow Y_1 \quad \text{imply} \quad X_0 \sqcup X_1 \hookrightarrow Y_0 \sqcup Y_1 \text{ and } X_0 \times X_1 \hookrightarrow Y_0 \times Y_1 \,. \tag{17}$$

Crucially, nwqo reflections preserve amortised controlled antichains. Indeed, let $r\colon X \hookrightarrow Y$, and consider a sequence $x_0, x_1, \ldots$ over X. Then by (15), $r(x_0), r(x_1), \ldots$ is an antichain when $x_0, x_1, \ldots$ is, and by (16), it is (g, n)-controlled when $x_0, x_1, \ldots$ is. Hence

$$X \hookrightarrow Y \quad \text{implies} \quad W^{\mathsf{a}}_{g,X}(n) \leq W^{\mathsf{a}}_{g,Y}(n) \text{ for all } g, n \,. \tag{18}$$

Inductive Reflection of Residuals. We provide a strict polynomial wqo reflecting $X_{\perp x}$ by induction over the structure of the strict polynomial nwqo X. The key difference compared to the analysis of bad sequences in [14, 41] occurs for $X = \mathbb{N}$: if $k \in \mathbb{N}$,

$$\mathbb{N}_{\perp k} = \mathbf{0} \,. \tag{19}$$

Regarding disjoint sums $X_0 \sqcup X_1$, it is plain that

$$(X_0 \sqcup X_1)_{\perp(x,i)} = (X_i)_{\perp x} \sqcup X_{1-i} \,. \tag{20}$$

Consider now $(\mathbb{N}^d)_{\perp \boldsymbol{v}}$ where $d > 1$ and $\boldsymbol{v} \in \mathbb{N}^d$. Observe that if $\boldsymbol{u} \in \mathbb{N}^d$ is such that $\boldsymbol{u} \perp \boldsymbol{v}$, then there exists $1 \leq i \leq d$ such that $\boldsymbol{u}(i) < \boldsymbol{v}(i)$, as otherwise we would have $\boldsymbol{u} \geq \boldsymbol{v}$. Thus

$$(\mathbb{N}^d)_{\perp \boldsymbol{v}} \hookrightarrow \bigsqcup_{1 \leq i \leq d} \mathbb{N}^{d-1} \times [\boldsymbol{v}(i)] \hookrightarrow \mathbb{N}^{d-1} \cdot \sum_{1 \leq i \leq d} \boldsymbol{v}(i) \,. \tag{21}$$

Ordinal Notations. As it is more convenient to reason with ordinals than with polynomial nwqos, we use the following bijection between strict polynomial nwqos and ω^ω:

$$w(\mathbf{0}) \stackrel{\text{def}}{=} 0 \,, \qquad w(\mathbb{N}^d) \stackrel{\text{def}}{=} \omega^{d-1} \,, \qquad w(X_0 \sqcup X_1) = w(X_0) \oplus w(X_1) \,. \tag{22}$$

where "$\oplus$" denotes the *natural sum* (aka *Hessenberg sum*) on ordinals: the natural sum $\alpha \oplus \beta$ of two ordinals with Cantor normal forms $\alpha = \sum_{i=1}^{p} \omega^{\alpha_i}$ and $\beta = \sum_{j=1}^{m} \omega^{\beta_j}$ is $\omega^{\gamma_1} + \cdots + \omega^{\gamma_{p+m}}$ where the exponents $\gamma_1 \geq \cdots \geq \gamma_{p+m}$ are a reordering of $\alpha_1, \ldots, \alpha_p, \beta_1, \ldots, \beta_m$. Given a strict polynomial nwqo $X = \bigsqcup_{i=1}^{m} \mathbb{N}^{d_i}$, its associated ordinal is $w(X) = \bigoplus_{i=1}^{m} \omega^{d_i - 1}$. In the case of an LCM with $d \stackrel{\text{def}}{=} |\mathtt{C}|$ counters and $q \stackrel{\text{def}}{=} |Q|$ locations, $w(Q \times \mathbb{N}^{\mathtt{C}}) = \omega^{d-1} \cdot q$.

For each $n \in \mathbb{N}$, we define a relation ∂_n over ordinals in ω^ω that mirrors the inductive residuation and reflection operations on strict polynomial nwqos X over the ordinals $w(X)$:

$$\partial_n \alpha \stackrel{\text{def}}{=} \left\{ \gamma \oplus \partial_n \omega^d \mid \alpha = \gamma \oplus \omega^d \right\} , \qquad \partial_n \omega^d \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } d = 0, \\ \omega^{d-1} \cdot n(d+1) & \text{otherwise.} \end{cases} \tag{23}$$

The intuition here is that $w(Y) \in \partial_n w(X)$ implies $X_{\perp x} \hookrightarrow Y$ for some $x \in X_{\leq n}$. Observe that $\alpha' \in \partial_n \alpha$ implies $\alpha' < \alpha$, thus $\bigcup_n \partial_n$ is a well-founded relation. This leads to the definition of an over-approximation of the width function $W^{\mathsf{a}}_{g,X}$ (see the full version):

$$M_{g,\alpha}(n) \stackrel{\text{def}}{=} \max_{\alpha' \in \partial_n \alpha} \left\{1 + M_{g,\alpha'}(g(n))\right\} . \tag{24}$$

▶ **Proposition 4.** *Let $(X, \leq_X, |\,.\,|_X)$ be a strict polynomial nwqo, $n_0 \in \mathbb{N}$, and $g\colon \mathbb{N} \to \mathbb{N}$. Then $W^{\mathsf{a}}_{g,X}(n_0) \leq M_{g,w(X)}(n_0)$.*

It remains to compare $M_{g,\alpha}$ with standard subrecursive functions like the Cichoń functions, which was already done in [41, Sec. 2.4.3] for a very similar function; see the full version.

▶ **Proposition 5.** *Let n be in $\mathbb{N}$, $g\colon \mathbb{N} \to \mathbb{N}$ a monotone inflationary function, and $0 < \alpha < \omega^d$. Then $M_{g,\alpha}(n) \leq 1 + M_{g,P_{nd}(\alpha)}(n)$.*

This extra twist of using a predecessor function different from the standard one from (12) can be avoided by instead over-approximating the control function g.

▶ **Theorem 6** (Width Function for Strict Polynomial nwqos)**.** *Let $d > 0$, $(X, \leq_X, |\,.\,|_X)$ be a strict polynomial nwqo with $w(X) < \omega^d$, $n_0 \in \mathbb{N}$, $g\colon \mathbb{N} \to \mathbb{N}$ monotone inflationary, and let $h\colon \mathbb{N} \to \mathbb{N}$ be a monotone function such that $h(x \cdot d) \geq g(x) \cdot d$ for all x. Then $W^{\mathsf{a}}_{g,X}(n_0) \leq h_{w(X)}(n_0 d)$.*

Proof. By Proposition 4, it suffices to show that $M_{g,w(X)}(n) \leq h_{w(X)}(nd)$, which we do by induction over $\alpha \stackrel{\text{def}}{=} w(X)$. If $\alpha = 0$, then $\partial_n \alpha = \emptyset$ thus $M_{g,\alpha}(n) = 0 \leq h_\alpha(nd)$. Otherwise, by Proposition 5, $M_{g,\alpha}(n) \leq 1 + M_{g,P_{nd}(\alpha)}(g(n))$. Because $P_{nd}(\alpha) < \alpha$, we can apply the induction hypothesis, yielding $M_{g,\alpha}(n) \leq 1 + h_{P_{nd}(\alpha)}(g(n)d) \leq 1 + h_{P_{nd}(\alpha)}(h(nd)) = h_\alpha(nd)$, where the last inequality follows from $h(nd) \geq g(n)d$ and the monotonicity of $h_{P_{nd}(\alpha)}$. ◀

Setting $h(x) \stackrel{\text{def}}{=} g(x)d$ always satisfies the conditions of the theorem. There are cases where setting $h \stackrel{\text{def}}{=} g$ suffices: e.g., $g(x) \stackrel{\text{def}}{=} 2x$, $g(x) \stackrel{\text{def}}{=} x^2$, $g(x) \stackrel{\text{def}}{=} 2^x$, and more generally whenever g is *super-homogeneous*, i.e. satisfies $g(dx) \geq g(x)d$ for all $d, x \geq 1$. In the case of LCMs, where $w(Q \times \mathbb{N}^{\mathsf{c}}) < \omega^d$ if $d \stackrel{\text{def}}{=} |\mathsf{C}| > 0$, a control function $g(x) \stackrel{\text{def}}{=} x + 1 = H(x)$ fits, thus setting $h(x) \stackrel{\text{def}}{=} x + d = H^d(x)$ satisfies $h(dx) = dx + d = (x+1)d = g(x)d$.

By (8), Lemma 2, and (14), Theorem 6 also yields a bound on the strong norm function.

▶ **Corollary 7** (Strong Norm Function for Strict Polynomial nwqos)**.** *Let d, X, n_0, g, and h be as in Theorem 6. Then $N^{\mathsf{s}}_{g,X}(n_0) \leq h^{w(X)}(n_0 d)$.*

6 Wrapping up

We have now all the ingredients needed to prove an F_d upper bound on LCM Reachability. Let us first recall the definition of the fast-growing complexity classes from [38].

Fast-Growing Complexity Classes. The *fast-growing* complexity classes [38] form a strict ordinal-indexed hierarchy of complexity classes $(\mathsf{F}_\alpha)_{\alpha < \varepsilon_0}$ using the Hardy functions $(H^\alpha)_{\alpha < \varepsilon_0}$ relative to $H(x) \stackrel{\text{def}}{=} x + 1$ as a standard against which to measure high complexities. Let

$$\mathcal{F}_{<\alpha} \stackrel{\text{def}}{=} \bigcup_{\beta < \omega^\alpha} \mathsf{FDTIME}\big(H^\beta(n)\big) , \qquad \mathsf{F}_\alpha \stackrel{\text{def}}{=} \bigcup_{p \in \mathcal{F}_{<\alpha}} \mathsf{DTIME}\big(H^{\omega^\alpha}(p(n))\big) . \tag{25}$$

Then $\mathcal{F}_{<\alpha}$ is the class of functions computed by deterministic Turing machines in time $O(H^\beta(n))$ for some $\beta < \omega^\alpha$; this captures for instance the class of Kalmar elementary functions as $\mathcal{F}_{<3}$ and the class of primitive-recursive functions as $\mathcal{F}_{<\omega}$ [30, 48]. The class F_α is

the class of decision problems solved by deterministic Turing machines in time $O\big(H^{\omega^\alpha}(p(n))\big)$ for some function $p \in \mathcal{F}_{<\alpha}$. The intuition behind this quantification of p is that, just like e.g. $\mathsf{EXP} = \bigcup_{p\in\mathsf{poly}} \mathsf{DTIME}\big(2^{p(n)}\big)$ quantifies over polynomial functions to provide enough "wiggle room" to account for polynomial reductions, F_α is closed under $\mathcal{F}_{<\alpha}$ reductions [38, Thms. 4.7 and 4.8]. For instance, $\mathsf{TOWER} \stackrel{\text{def}}{=} \mathsf{F}_3$ defines the class of problems that can be solved in time bounded by a tower of exponentials of elementary height in the size of the input, $\bigcup_{k\in\mathbb{N}} \mathsf{F}_k$ is the class of primitive-recursive decision problems, and $\mathsf{ACKERMANN} \stackrel{\text{def}}{=} \mathsf{F}_\omega$ is the class of problems that can be solved in time bounded by the Ackermann function applied to some primitive-recursive function of the input size.

Upper Bound. Recall from Section 3.3 that a pseudo-witness for coverability of a configuration $q_f(\boldsymbol{v}_f)$ in a LCM with $d \stackrel{\text{def}}{=} |\mathtt{C}| > 0$ counters and $q \stackrel{\text{def}}{=} |Q|$ locations is a strongly $(H, |\boldsymbol{v}_f|)$-controlled bad sequence over $Q \times \mathbb{N}^{\mathtt{C}}$, which as discussed in Section 5.2 is a strict polynomial wqo with $w(Q \times \mathbb{N}^{\mathtt{C}}) = \omega^{d-1} \cdot q < \omega^d$, and that $h \stackrel{\text{def}}{=} H^d$ fits the conditions of Theorem 6 and Corollary 7. As stated in Theorem 1, together with the lower bounds from [39], the following entails the F_d-completeness of LCM Reachability with a fixed number $d \geq 3$ of counters.

▶ **Proposition 8** (Upper Bound for LCM Reachability)**.** *LCM Reachability is in F_ω, and in F_d if the number $d \geq 3$ of counters is fixed.*

Proof. Let $n_0 \stackrel{\text{def}}{=} |\boldsymbol{v}_f|$ be the infinite norm of the target configuration, $d \stackrel{\text{def}}{=} |\mathtt{C}| \geq 3$ be the number of counters, and $q \stackrel{\text{def}}{=} |Q| \geq 1$ the number of locations. By Corollary 7, the elements in a pseudo-witness of the coverability of $q_f(\boldsymbol{v}_f)$ are of norm at most $N \stackrel{\text{def}}{=} N^{\mathsf{s}}_{H,Q\times\mathbb{N}^{\mathtt{C}}}(n_0) = h^{\omega^{d-1}\cdot q}(n_0)$ for $h(x) \stackrel{\text{def}}{=} H^d(x)$. Let $n \stackrel{\text{def}}{=} \max\{qd-1, n_0\}$. As shown in the full version, this means that

$$N \leq H^{\omega^{d-1}\cdot qd}(n_0) \leq H^{\omega^{d-1}\cdot qd}(n) \leq H^{\omega^d}(n) \tag{26}$$

by monotonicity of the Hardy functions.

Note that there are at most $q(N+1)^d$ different configurations in $Q \times \mathbb{N}^{\mathtt{C}}$ of norm bounded by N, i.e., $|(Q \times \mathbb{N}^{\mathtt{C}})_{\leq N}| \leq q(N+1)^d$. By (9), this is also a bound on the strong length function $L^{\mathsf{s}}_{H,Q\times\mathbb{N}^d}(n_0)$. Thus the number of steps in the backward coverability algorithm is bounded by $q(N+1)^d$, and each step can be carried in time $O(N)$, hence the algorithm works in deterministic time $O(q(N+1)^{d+1}) = O(f(N)) = O(f(H^{\omega^d}(n)))$ for some elementary function $f \in \mathcal{F}_{<3}$. By [38, Cor. A.9], there exists an elementary inflationary function $p \in \mathcal{F}_{<3}$ such that $f(H^{\omega^d}(n)) \leq H^{\omega^d}(p(n))$: the backward coverability algorithm therefore works in deterministic time $O(H^{\omega^d}(p(n)))$ for some $p \in \mathcal{F}_{<3}$, which is an expression of the form (25).

Therefore, LCM Reachability is in F_d when d is fixed, and in F_ω otherwise because $p(n) \geq n \geq d-1$ and thus $H^{\omega^d}(p(n)) \leq H^{\omega^\omega}(p(n))$. ◀

7 Concluding Remarks

We have shown the F_d-completeness of reachability in lossy counter machines with a fixed number $d \geq 3$ of counters. The key novelty is that we analyse the length of controlled antichains over $\mathbb{N}^d$ rather than that of controlled bad sequences. A possible explanation why this leads to improved upper bounds is that the *ordinal width* of $\mathbb{N}^d$, i.e., the ordinal rank of its antichains, is conjectured to be ω^{d-1} [13], while its *maximal order type*, i.e., the ordinal rank of its bad sequences, is well-known to be ω^d [9].

Our approach might be employed to tackle related parameterised complexity gaps, like the one between $\mathsf{F}_{\omega^{m-2}}$-hardness [25] and $\mathsf{F}_{\omega^{m-1}+1}$ membership [40] of reachability in *lossy channel systems* with $m \geq 4$ channel symbols and a single channel. Those results rely however on the set of finite words over an alphabet of size m being a wqo for Higman's scattered subword ordering [21], for which the ordinal width and maximal order type coincide at $\omega^{\omega^{m-1}}$ [13, 9].

References

1 Parosh A. Abdulla, Karlis Čerāns, Bengt Jonsson, and Yih-Kuen Tsay. Algorithmic analysis of programs with well quasi-ordered domains. *Information and Computation*, 160(1–2):109–127, 2000. doi:10.1006/inco.1999.2843.

2 Parosh A. Abdulla and Bengt Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 1996. doi:10.1006/inco.1996.0053.

3 Sergio Abriola, Santiago Figueira, and Gabriel Senno. Linearizing well-quasi orders and bounding the length of bad sequences. *tcs*, 603:3–22, 2015. doi:10.1016/j.tcs.2015.07.012.

4 Roberto M. Amadio and Charles Meyssonnier. On Decidability of the Control Reachability Problem in the Asynchronous π-calculus. *Nordic Journal of Computing*, 9(2):70–101, 2002.

5 André Arnold and M. Latteux. Récursivité et cones rationnels fermés par intersection. *CALCOLO*, 15(4):381–394, 1978. doi:10.1007/BF02576519.

6 Pierre Chambart and Philippe Schnoebelen. The Ordinal Recursive Complexity of Lossy Channel Systems. In *Proceedings of LICS 2008*, pages 205–216. IEEE, 2008. doi:10.1109/LICS.2008.47.

7 E. Adam Cichoń and Elias Tahhan Bittar. Ordinal Recursive Bounds for Higman's Theorem. *tcs*, 201(1–2):63–84, 1998. doi:10.1016/S0304-3975(97)00009-1.

8 Peter Clote. On the finite containment problem for Petri nets. *tcs*, 43:99–105, 1986. doi:10.1016/0304-3975(86)90169-6.

9 Dick H. J. de Jongh and Rohit Parikh. Well-partial orderings and hierarchies. *Indagationes Mathematicae*, 39(3):195–207, 1977. doi:10.1016/1385-7258(77)90067-1.

10 Normann Decker and Daniel Thoma. On Freeze LTL with Ordered Attributes. In *Proceedings of FoSSaCS 2016*, volume 9634 of *Lecture Notes in Computer Science*, pages 269–284. Springer, 2016. doi:10.1007/978-3-662-49630-5_16.

11 Stéphane Demri and Ranko Lazić. LTL with the freeze quantifier and register automata. *ACM Transactions on Computational Logic*, 10(3):16:1–16:30, 2009. doi:10.1145/1507244.1507246.

12 Leonard Eugene Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *American Journal of Mathematics*, 35(4):413–422, 1913. doi:10.2307/2370405.

13 Mirna Džamonja, Sylvain Schmitz, and Philippe Schnoebelen. On Ordinal Invariants in Well Quasi Orders and Finite Antichain Orders. In Monika Seisenberger, Peter Schuster, and Andreas Weiermann, editors, *Well-Quasi Orders in Computation, Logic, Language and Reasoning*, Trends in Logic. Springer, 2019. To appear. arXiv:1711.00428[math.LO].

14 Diego Figueira, Santiago Figueira, Sylvain Schmitz, and Philippe Schnoebelen. Ackermannian and primitive-recursive bounds with Dickson's Lemma. In *Proceedings of LICS 2011*, pages 269–278. IEEE, 2011. doi:10.1109/LICS.2011.39.

15 Diego Figueira and Luc Segoufin. Future-looking logics on data words and trees. In *Proceedings of MFCS 2009*, volume 5734 of *Lecture Notes in Computer Science*, pages 331–343. Springer, 2009. doi:10.1007/978-3-642-03816-7_29.

16 Alain Finkel and Philippe Schnoebelen. Well-Structured Transition Systems Everywhere! *tcs*, 256(1–2):63–92, 2001. doi:10.1016/S0304-3975(00)00102-X.

17 Radu Grigore and Nikos Tzevelekos. History-Register Automata. *Logical Methods in Computer Science*, 12(1):7:1–7:32, 2016. doi:10.2168/LMCS-12(1:7)2016.

18 Christoph Haase, Sylvain Schmitz, and Philippe Schnoebelen. The Power of Priority Channel Systems. *Logical Methods in Computer Science*, 10(4):4:1–4:39, 2014. doi:10.2168/LMCS-10(4:4)2014.

19 Serge Haddad, Sylvain Schmitz, and Philippe Schnoebelen. The Ordinal Recursive Complexity of Timed-Arc Petri Nets, Data Nets, and Other Enriched Nets. In *Proceedings of LICS 2012*, pages 355–364. IEEE, 2012. doi:10.1109/LICS.2012.46.

20 Matthew Hague. Senescent Ground Tree Rewriting Systems. In *Proceedings of CSL-LICS 2014*, pages 48:1–48:10. ACM, 2014. doi:10.1145/2603088.2603112.

21 Graham Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, 3(2):326–336, 1952. doi:10.1112/plms/s3-2.1.326.

22 Piotr Hofman and Patrick Totzke. Trace Inclusion for One-Counter Nets Revisited. *tcs*, 735:50–63, 2018. doi:10.1016/j.tcs.2017.05.009.

23 Rodney R. Howell, Louis E. Rosier, Dung T. Huynh, and Hsu-Chun Yen. Some complexity bounds for problems concerning finite and 2-dimensional vector addition systems with states. *tcs*, 46:107–140, 1986. doi:10.1016/0304-3975(86)90026-5.

24 Petr Jančar and Sylvain Schmitz. Bisimulation Equivalence of First-Order Grammars is ACKERMANN-Complete. In *Proceedings of LICS 2019*. IEEE, 2019. To appear. arXiv:1901.07170[cs.LO].

25 Prateek Karandikar and Sylvain Schmitz. The Parametric Ordinal-Recursive Complexity of Post Embedding Problems. In *Proceedings of FoSSaCS 2013*, volume 7794 of *Lecture Notes in Computer Science*, pages 273–288. Springer, 2013. doi:10.1007/978-3-642-37075-5_18.

26 Joseph B. Kruskal. The theory of well-quasi-ordering: A frequently discovered concept. *Journal of Combinatorial Theory, Series A*, 13(3):297–305, 1972. doi:10.1016/0097-3165(72)90063-5.

27 Ranko Lazić, Joël O. Ouaknine, and James B. Worrell. Zeno, Hercules and the Hydra: Safety Metric Temporal Logic is ACKERMANN-complete. *ACM Transactions on Computational Logic*, 17(3), 2016. doi:10.1145/2874774.

28 Ranko Lazić and Sylvain Schmitz. Non-Elementary Complexities for Branching VASS, MELL, and Extensions. *ACM Transactions on Computational Logic*, 16(3):20:1–20:30, 2015. doi:10.1145/2733375.

29 Ranko Lazić and Sylvain Schmitz. The Complexity of Coverability in ν-Petri Nets. In *Proceedings of LICS 2016*, pages 467–476. ACM, 2016. doi:10.1145/2933575.2933593.

30 Martin H. Löb and Stanley S. Wainer. Hierarchies of number theoretic functions, I. *Archiv für Mathematische Logik und Grundlagenforschung*, 13:39–51, 1970. doi:10.1007/BF01967649.

31 Ernst W. Mayr and Albert R. Meyer. The Complexity of the Finite Containment Problem for Petri Nets. *J. ACM*, 28(3):561–576, 1981. doi:10.1145/322261.322271.

32 Richard Mayr. Undecidable problems in unreliable computations. *tcs*, 297(1–3):337–354, 2003. doi:10.1016/S0304-3975(02)00646-1.

33 Kenneth McAloon. Petri nets and large finite sets. *Theor. Comput. Sci.*, 32(1–2):173–183, 1984. doi:10.1016/0304-3975(84)90029-X.

34 Robin Milner. Operational and Algebraic Semantics of Concurrent Processes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 19, pages 1201–1242. Elsevier, 1990. doi:10.1016/B978-0-444-88074-1.50024-X.

35 Angelo Montanari, Gabriele Puppis, and Pietro Sala. Maximal Decidable Fragments of Halpern and Shoham's Modal Logic of Intervals. In *Proceedings of ICALP 2010*, volume 6199 of *Lecture Notes in Computer Science*, pages 345–356. Springer, 2010. doi:10.1007/978-3-642-14162-1_29.

36 Guillermo A. Pérez. The Fixed Initial Credit Problem for Partial-Observation Energy Games is ACK-complete. *ipl*, 118:91–99, 2017. doi:10.1016/j.ipl.2016.10.005.

37 Fernando Rosa-Velardo. Ordinal recursive complexity of unordered data nets. *Information and Computation*, 254(1):41–58, 2017. doi:10.1016/j.ic.2017.02.002.

38 Sylvain Schmitz. Complexity Hierarchies Beyond ELEMENTARY. *ACM Transactions on Computation Theory*, 8(1):3:1–3:36, 2016. doi:10.1145/2858784.

39 Sylvain Schmitz. *Algorithmic Complexity of Well-Quasi-Orders*. Habilitation thesis, École Normale Supérieure Paris-Saclay, 2017. tel.archives-ouvertes.fr:tel-01663266.

40 Sylvain Schmitz and Philippe Schnoebelen. Multiply-Recursive Upper Bounds with Higman's Lemma. In *Proceedings of ICALP 2011*, volume 6756 of *Lecture Notes in Computer Science*, pages 441–452. Springer, 2011. doi:10.1007/978-3-642-22012-8_35.

41 Sylvain Schmitz and Philippe Schnoebelen. Algorithmic Aspects of WQO Theory. Lecture notes, 2012. cel.archives-ouvertes.fr:cel-00727025.

42 Sylvain Schmitz and Philippe Schnoebelen. The Power of Well-Structured Systems. In *Proceedings of Concur 2013*, volume 8052 of *Lecture Notes in Computer Science*, pages 5–24. Springer, 2013. doi:10.1007/978-3-642-40184-8_2.

43 Philippe Schnoebelen. Verifying Lossy Channel Systems has Nonprimitive Recursive Complexity. *Inf. Process. Lett.*, 83(5):251–261, 2002. doi:10.1016/S0020-0190(01)00337-4.

44 Philippe Schnoebelen. Lossy Counter Machines Decidability Cheat Sheet. In *Proceedings of RP 2010*, volume 6227 of *Lecture Notes in Computer Science*, pages 51–75. Springer, 2010. doi:10.1007/978-3-642-15349-5_4.

45 Philippe Schnoebelen. Revisiting Ackermann-Hardness for Lossy Counter Machines and Reset Petri Nets. In *Proceedings of MFCS 2010*, volume 6281 of *Lecture Notes in Computer Science*, pages 616–628. Springer, 2010. doi:10.1007/978-3-642-15155-2_54.

46 Helmut Schwichtenberg and Stanley S. Wainer. *Proofs and Computation*. Perspectives in Logic. Cambridge University Press, 2012.

47 Alasdair Urquhart. The Complexity of Decision Procedures In Relevance Logic II. *Journal of Symbolic Logic*, 64(4):1774–1802, 1999. doi:10.2307/2586811.

48 Stanley S. Wainer. Ordinal recursion, and a refinement of the extended Grzegorczyk hierarchy. *Journal of Symbolic Logic*, 37(2):281–292, 1972. doi:10.2307/2272973.

49 Andreas Weiermann. Complexity bounds for some finite forms of Kruskal's Theorem. *Journal of Symbolic Computation*, 18(5):463–488, 1994. doi:10.1006/jsco.1994.1059.

Varieties of Data Languages

Henning Urbat
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
henning.urbat@fau.de

Stefan Milius
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
mail@stefan-milius.eu

Abstract

We establish an Eilenberg-type correspondence for data languages, i.e. languages over an infinite alphabet. More precisely, we prove that there is a bijective correspondence between varieties of languages recognized by orbit-finite nominal monoids and pseudovarieties of such monoids. This is the first result of this kind for data languages. Our approach makes use of nominal Stone duality and a recent category theoretic generalization of Birkhoff-type theorems that we instantiate here for the category of nominal sets. In addition, we prove an axiomatic characterization of weak pseudovarieties as those classes of orbit-finite monoids that can be specified by sequences of nominal equations, which provides a nominal version of a classical theorem of Eilenberg and Schützenberger.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Nominal sets, Stone duality, Algebraic language theory, Data languages

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.130

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version `https://arxiv.org/abs/1903.08053`

Funding Henning Urbat acknowledges support by Deutsche Forschungsgemeinschaft (DFG) under project SCHR 1118/8-2. Stefan Milius acknowledges support by Deutsche Forschungsgemeinschaft (DFG) under project MI 717/5-1.

1 Introduction

In the algebraic theory of formal languages, one studies automata and the languages they represent in terms of associated algebraic structures. This approach has been successfully implemented for numerous types of languages and has proven extremely fruitful because it allows to import powerful algebraic methods into the realm of automata theory. As a prime example, regular languages can be described purely algebraically as the languages recognized by finite monoids, and a celebrated result by McNaughton, Papert, and Schützenberger [12, 19] asserts that a regular language is definable in first-order logic if and only if its syntactic monoid is aperiodic (i.e. it satisfies the equation $x^{n+1} = x^n$ for sufficiently large n). As an immediate application, this algebraic characterization yields an effective procedure for deciding first-order definability. The first systematic approach to correspondence results of this kind was initiated by Eilenberg [6] who proved that *varieties of languages* (i.e. classes of regular languages closed under the set-theoretic boolean operations, derivatives, and homomorphic preimages) correspond bijectively to *pseudovarieties of monoids* (i.e. classes of finite monoids closed under quotients monoids, submonoids, and finite products). Eilenberg's result thus establishes a generic relation between properties of regular languages and properties of finite monoids. In addition, Eilenberg and Schützenberger [7] contributed a model-theoretic description

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 130; pp. 130:1–130:14


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

of pseudovarieties: they are those classes of finite monoids that can be axiomatized by a sequence $(s_n = t_n)_{n\in\mathbb{N}}$ of equations, interpreted as "$s_n = t_n$ holds for sufficiently large n". For instance, the pseudovariety of aperiodic finite monoids is axiomatized by $(x^{n+1} = x^n)_{n\in\mathbb{N}}$.

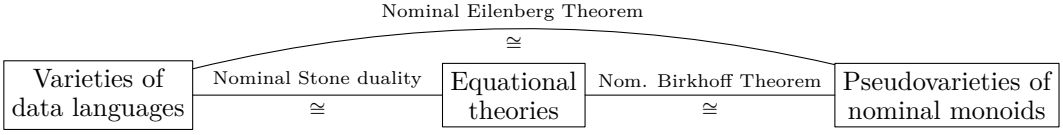
The goal of our present paper is to study *data languages*, i.e. languages over an infinite alphabet, from the perspective of algebraic language theory. Such languages have spurred significant interest in recent years, driven by practical applications in various areas of computer science, including efficient parsing of XML documents or software verification. Mathematically, data languages are modeled using *nominal sets*. Over the years, several machine models for handling data languages of different expressive power have been proposed; see [21, 20] for a comprehensive survey. The focus of this paper is on languages recognized by *orbit-finite nominal monoids*. They form an important subclass of the languages accepted by Francez and Kaminski's *finite memory automata* [10] (which are expressively equivalent to orbit-finite automata in the category of nominal sets [5]) and have been characterized by a fragment of monadic second-order logic over data words called *rigidly guarded MSO* [17]. In addition, Bojańczyk [4] and Colcombet, Ley, and Puppis [17] established nominal versions of the McNaughton-Papert-Schützenberger theorem and showed that the first-order definable data languages are precisely the ones recognizable by aperiodic orbit-finite monoids.

In the light of these results, it is natural to ask whether a generic variety theory akin to Eilenberg's seminal work can be developed for data languages. As the main contribution of our paper, we answer this positively by establishing nominal generalizations of two key results known from the algebraic theory of regular languages. The first one is a counterpart of Eilenberg's variety theorem, which is the first result of this kind for data languages:

▶ **Nominal Eilenberg Theorem.** *Varieties of data languages correspond bijectively to pseudovarieties of nominal monoids.*

Here, the notion of a *pseudovariety of nominal monoids* is as expected: a class of orbit-finite nominal monoids closed under quotient monoids, submonoids, and finite products. In contrast, the notion of a *variety of data languages* requires two extra conditions unfamiliar from other Eilenberg-type correspondences, most notably a technical condition called *completeness* (Definition 4.13). Like the original Eilenberg theorem, its nominal version gives rise to a generic relation between properties of data languages and properties of nominal monoids. For instance, the aperiodic orbit-finite monoids form a pseudovariety, and the first-order definable data languages form a variety, and thus the equivalence of these concepts can be understood as an instance of the nominal Eilenberg correspondence.

On a conceptual level, our results crucially make use of *duality*, specifically an extension of Petrişan's [15] nominal version of Stone duality which gives a dual equivalence between nominal sets and nominal complete atomic boolean algebras. To derive the nominal Eilenberg correspondence, we make two key observations. First, we show that varieties of data languages dualize (under nominal Stone duality) to the concept of an *equational theory* in the category of nominal sets. Second, we apply a recent categorical generalization of Birkhoff-type variety theorems [14] to show that equational theories correspond to pseudovarieties of nominal monoids. Our approach is summarized by the diagram below:



The idea that Stone-type dualities play a major role in algebraic language theory was firmly established by Gehrke, Grigorieff, and Pin [9]. It is also at the heart of our recent line of work [1, 2, 22, 3], which culminated in a uniform category theoretic proof of more than a dozen Eilenberg correspondences for various types of languages. A related, yet more abstract, approach was pursued by Salamánca [18]. The key insight of [22, 18] is that Eilenberg-type correspondences arise by combining a Birkhoff-type correspondence with a Stone-type duality. Our present approach to data languages is an implementation of this principle in the nominal setting. Since the existing categorical frameworks for algebraic language theory consider algebraic-like base categories (which excludes nominal sets) and the recognition of languages by finite structures, our Nominal Eilenberg Theorem is not covered by any previous categorical work and requires new techniques. However, our approach can be seen as an indication of the robustness of the duality-based methodology for algebraic recognition.

As our second main contribution, we complement the Nominal Eilenberg Theorem with a model-theoretic description of pseudovarieties of nominal monoids in terms of sequences of *nominal equations*, generalizing the classical result of Eilenberg and Schützenberger for ordinary monoids. Our result applies more generally to the class of *weak pseudovarieties* of nominal monoids, which are only required to be closed under support-reflecting (rather than arbitrary) quotients. We then obtain the

▶ **Nominal Eilenberg-Schützenberger Theorem.** *Weak pseudovarieties are exactly the classes of nominal monoids axiomatizable by sequences of nominal equations.*

While our main results apply to languages recognizable by orbit-finite monoids, the underlying methods are of fairly general nature and can be extended to other recognizing structures in the category of nominal sets. We illustrate this in section 5 by deriving a (local) Eilenberg correspondence for languages accepted by deterministic nominal automata.

Full proofs of all results can be found in the extended arXiv version [23] of our paper.

2 Nominal Sets

We start by recalling basic definitions and facts from the theory of nominal sets [16]. Some of the concepts considered in this paper are most clearly and conveniently formulated in the language of category theory, but only very basic knowledge of category theory is required from the reader. Fix a countably infinite set $\mathbb{A}$ of *atoms*, and denote by $\mathrm{Perm}(\mathbb{A})$ the group of finite permutations of $\mathbb{A}$ (i.e. bijections $\pi\colon \mathbb{A} \to \mathbb{A}$ that move only finitely many elements of $\mathbb{A}$). A $\mathrm{Perm}(\mathbb{A})$-*set* is a set X with an operation $\mathrm{Perm}(\mathbb{A}) \times X \to X$, denoted as $(\pi, x) \mapsto \pi \cdot x$, such that $(\sigma\pi)\cdot x = \sigma \cdot (\pi \cdot x)$ and $id \cdot x$ for all $\sigma, \pi \in \mathrm{Perm}(\mathbb{A})$ and $x \in X$. If the group action is trivial, i.e. $\pi \cdot x = x$ for all $\pi \in \mathrm{Perm}(\mathbb{A})$ and $x \in X$, we call X *discrete*. For any set $S \subseteq \mathbb{A}$ of atoms, denote by $\mathrm{Perm}_S(\mathbb{A}) \subseteq \mathrm{Perm}(\mathbb{A})$ the subgroup of all finite permutations π that fix S, i.e. $\pi(a) = a$ for all $a \in S$. The set S is called a *support* of an element $x \in X$ if for every $\pi \in \mathrm{Perm}_S(\mathbb{A})$ one has $\pi \cdot x = x$. The intuition is that x is some kind of syntactic object (e.g. a string, a tree, a term, or a program) whose free variables are contained in S. Thus, a variable renaming π that leaves S fixed does not affect x. A *nominal set* is a $\mathrm{Perm}(\mathbb{A})$-set X such that every element of X has a finite support. This implies that every element $x \in X$ has a least support, denoted by $\mathsf{supp}_X(x) \subseteq \mathbb{A}$. A nominal set X is *strong* if, for every $x \in X$ and $\pi \in \mathrm{Perm}(\mathbb{A})$, one has $\pi \cdot x = x$ if and only if $\pi(a) = a$ for all $a \in \mathsf{supp}_X(x)$. The *orbit* of an element x of a nominal set X is the set $\{\pi \cdot x \ : \ \pi \in \mathrm{Perm}(\mathbb{A})\}$. The orbits form a partition of X. If X has only finitely many orbits, then X is called *orbit-finite*. More generally, for any finite set $S \subseteq \mathbb{A}$ of atoms, the S-*orbit* of an element $x \in X$ is the set $\{\pi \cdot x \ : \ \pi \in \mathrm{Perm}_S(\mathbb{A})\}$, and the S-orbits form a partition of X.

▶ **Lemma 2.1.** *Let S be a finite subset of $\mathbb{A}$. Then every orbit-finite nominal set has only finitely many S-orbits.*

A map $f\colon X \to Y$ between nominal sets is *equivariant* if $f(\pi\cdot x) = \pi\cdot f(x)$ for all $\pi \in \mathrm{Perm}(\mathbb{A})$ and $x \in X$, and *finitely supported* if there exists a finite set $S \subseteq \mathbb{A}$ such that $f(\pi\cdot x) = \pi\cdot f(x)$ for all $\pi \in \mathrm{Perm}_S(\mathbb{A})$ and $x \in X$. Equivariant maps do not increase supports, i.e. one has $\mathsf{supp}_Y(f(x)) \subseteq \mathsf{supp}_X(x)$ for all $x \in X$. We write $\mathbf{Nom}_{\mathsf{fs}}$ for the category of nominal sets and finitely supported maps, and **Nom** for the (non-full) subcategory of nominal sets and equivariant maps. We shall use the following standard results about **Nom**:

(1) **Nom** is complete and cocomplete. Finite limits and all colimits are formed on the level of underlying sets. In particular, finite products of nominal sets are given by cartesian products and coproducts by disjoint union.

(2) For every pair X, Y of nominal sets, the *exponential* $[X, Y]$ is the nominal set consisting of all finitely supported maps $f\colon X \to Y$, with the group action given by $(\pi\cdot f)(x) = \pi\cdot f(\pi^{-1}\cdot x)$. Moreover, for every nominal set X, the *nominal power set* $\mathcal{P}X$ is carried by the set of all subsets $X_0 \subseteq X$ with finite support; i.e. for which there exists a finite set $S \subseteq \mathbb{A}$ of atoms such that $\pi\cdot X_0 = X_0$ for $\pi \in \mathrm{Perm}_S(\mathbb{A})$, where $\pi\cdot X_0 = \{\pi\cdot x \;:\; x \in X_0\}$. In particular, every singleton $\{x\}$ is finitely supported by $\mathsf{supp}_X(x)$. The group action on $\mathcal{P}X$ is given by $X_0 \mapsto \pi\cdot X_0$, and we have $\mathcal{P}X \cong [X, 2]$, for the discrete nominal set $2 = \{0, 1\}$.

(3) *Quotients* and *subobjects* in **Nom** are represented by epimorphisms (= surjective equivariant maps) and monomorphisms (= injective equivariant maps), respectively. **Nom** has image factorizations, i.e. every equivariant map $f\colon X \to Y$ has a unique decomposition $f = m\cdot e$ into a quotient $e\colon X \twoheadrightarrow I$ followed by a subobject $m\colon I \rightarrowtail Y$. We call e the *coimage* of f.

(4) Orbit-finite nominal sets are closed under quotients, subobjects, and finite products.

(5) For each $n \geq 0$, the nominal set $\mathbb{A}^{\#n} = \{\,(a_1, \ldots, a_n) \in \mathbb{A}^n \;:\; a_i \neq a_j \text{ for } i \neq j\,\}$ with group action $\pi\cdot(a_1, \ldots, a_n) = (\pi(a_1), \ldots, \pi(a_n))$ is strong and has a single orbit. More generally, the (orbit-finite) strong nominal sets are up to isomorphism exactly the (finite) coproducts of nominal sets of the form $\mathbb{A}^{\#n}$.

3 Pseudovarieties of Nominal Monoids

In this section, we investigate classes of orbit-finite nominal monoids and establish two characterizations of such classes: a categorical one, relating pseudovarieties of nominal monoids to equational theories in the category of nominal sets, and an axiomatic one, describing weak pseudovarieties in terms of sequences of nominal equations. The first of these results is the algebraic foundation of our subsequent treatment of varieties of data languages.

A *nominal monoid* is a monoid $(M, \bullet, 1_M)$ in the category **Nom**; that is, M is equipped with the structure of a nominal set such that the multiplication $\bullet\colon M \times M \to M$ is an equivariant map and the unit $1_M \in M$ has empty support, i.e. it corresponds to an equivariant map $1 \to M$, where 1 is the nominal set with one element. We write **nMon** for the category of nominal monoids and equivariant monoid morphisms (usually just called *morphisms*), and $\mathbf{nMon}_{\mathsf{of}}$ for the full subcategory of orbit-finite nominal monoids. The forgetful functor from **nMon** to **Nom** has a left adjoint assigning to each nominal set Σ the free nominal monoid Σ^* of all words over Σ, with monoid multiplication given by concatentation of words, unit ε (the empty word) and group action $\pi\cdot(a_1\cdots a_n) = \pi(a_1)\cdots\pi(a_n)$ for $\pi \in \mathrm{Perm}(\mathbb{A})$ and $a_1\cdots a_n \in \Sigma^*$. The category **nMon** has products (formed on the level of **Nom**), image factorizations, and surjective and injective morphisms represent *quotients* and *submonoids* of

nominal monoids. A quotient $q\colon M \twoheadrightarrow M'$ is called *support-reflecting* if for every $x' \in M'$ there exists an $x \in M$ with $q(x) = x'$ and $\mathsf{supp}_M(x) = \mathsf{supp}_{M'}(x')$. The following result characterizes the quotient monoids of Σ^* in terms of unary operations:

▶ **Proposition 3.1** (Unary presentation for nominal monoids). *For every nominal set Σ and every surjective equivariant map $e\colon \Sigma^* \twoheadrightarrow M$, the following statements are equivalent:*

(1) *e carries a quotient monoid of Σ^*, i.e. there exists a nominal monoid structure $(M, \bullet, 1_M)$ on M such that $e\colon \Sigma^* \twoheadrightarrow (M, \bullet, 1_M)$ is a morphism of nominal monoids;*

(2) *the maps $\Sigma^* \xrightarrow{w\cdot -} \Sigma^*$ and $\Sigma^* \xrightarrow{-\cdot w} \Sigma^*$ $(w \in \Sigma^*)$ lift along e, i.e. there exist (necessarily unique) maps l_w and r_w making the following squares commute:*

$$\begin{array}{ccc} \Sigma^* & \xrightarrow{w\cdot -} & \Sigma^* \\ {\scriptstyle e}\downarrow & & \downarrow{\scriptstyle e} \\ M & \xrightarrow[l_w]{} & M \end{array} \qquad \begin{array}{ccc} \Sigma^* & \xrightarrow{-\cdot w} & \Sigma^* \\ {\scriptstyle e}\downarrow & & \downarrow{\scriptstyle e} \\ M & \xrightarrow[r_w]{} & M \end{array} \qquad \text{for every } w \in \Sigma^*.$$

In general, the maps $w \cdot -$ and $- \cdot w$ are not equivariant, but finitely supported (with support contained in the one of w). This implies that also l_w and r_w in 2 are finitely supported.

3.1 Equational Theories

In previous work [14] we studied varieties of objects in a general category and their relation to an abstract form of equations. In the following, we instantiate these concepts to the category of nominal sets to derive a characterization of pseudovarieties of orbit-finite monoids.

▶ **Definition 3.2.** Let Σ be a nominal set. A *Σ-generated nominal monoid* is a nominal quotient monoid $e\colon \Sigma^* \twoheadrightarrow M$ of the free monoid Σ^*. We denote by $\Sigma^*{\downarrow}\mathbf{nMon}_{\mathsf{of}}$ the poset of Σ-generated orbit-finite nominal monoids, ordered by $e \leq e'$ iff e' factorizes through e.

▶ **Definition 3.3.** A *local pseudovariety of Σ-generated nominal monoids* is a filter $\mathcal{T}_\Sigma \subseteq \Sigma^*{\downarrow}\mathbf{nMon}_{\mathsf{of}}$ in the poset of Σ-generated orbit-finite nominal monoids; that is, $\mathcal{T}_\Sigma$ is

(1) *upwards closed:* $e \in \mathcal{T}_\Sigma$ and $e \leq e'$ implies $e' \in \mathcal{T}_\Sigma$, and

(2) *downwards directed:* for each pair $e_0, e_1 \in \mathcal{T}_\Sigma$ there exists $e \in \mathcal{T}_\Sigma$ with $e \leq e_0, e_1$.

If we replace (1) by the weaker condition

(1') for each $e\colon \Sigma^* \twoheadrightarrow M$ in $\mathcal{T}_\Sigma$ and each support-reflecting $q\colon M \twoheadrightarrow N$ one has $q \cdot e \in \mathcal{T}_\Sigma$,

then $\mathcal{T}_\Sigma$ is called a *weak local pseudovariety of Σ-generated nominal monoids.*

▶ **Remark 3.4.** By Proposition 3.1, the definition of local pseudovariety can be equivalently stated as follows:

(1) $\mathcal{T}_\Sigma$ is a filter in the poset of orbit-finite quotients of Σ^* in $\mathbf{Nom}$;

(2) for every $e \in \mathcal{T}_\Sigma$ and $w \in \Sigma^*$, the unary operations $w \cdot -$ and $- \cdot w$ on Σ^* lift along e.

Let $\mathbf{Nom}_{\mathsf{of,s}}$ denote the full subcategory of $\mathbf{Nom}$ on orbit-finite strong nominal sets.

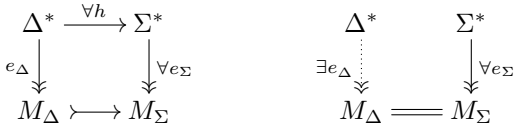
▶ **Definition 3.5** (Equational Theory). A *(weak) equational theory* is a family

$$\mathcal{T} = (\, \mathcal{T}_\Sigma \subseteq \Sigma^*{\downarrow}\mathbf{nMon}_{\mathsf{of}} \,)_{\Sigma \in \mathbf{Nom}_{\mathsf{of,s}}}$$

of (weak) local pseudovarieties with the following two properties (see the diagrams below):

(1) *Substitution invariance.* For each equivariant monoid morphism $h\colon \Delta^* \to \Sigma^*$ with $\Delta, \Sigma \in \mathbf{Nom}_{\mathsf{of,s}}$ and each $e_\Sigma\colon \Sigma^* \twoheadrightarrow M_\Sigma$ in $\mathcal{T}_\Sigma$, the coimage e_Δ of $e_\Sigma \cdot h$ lies in $\mathcal{T}_\Delta$.

(2) *Completeness.* For each $\Sigma \in \mathbf{Nom}_{\mathsf{of,s}}$ and each quotient $e\colon \Sigma^* \twoheadrightarrow M_\Sigma$ in $\mathcal{T}_\Sigma$, there exists $\Delta \in \mathbf{Nom}_{\mathsf{of,s}}$ and a support-reflecting quotient $e_\Delta\colon \Delta^* \twoheadrightarrow M_\Delta$ in $\mathcal{T}_\Delta$ with $M_\Delta = M_\Sigma$.



▶ **Remark 3.6.**

(1) Local pseudovarieties were previously called *equations* [14]. In fact, in many instances of the framework in *op. cit.*, a filter of quotients can be represented as a single quotient of a free algebra on an object Σ, which in turn corresponds to a set of pairs of terms given by the kernel of the quotient, i.e. to the usual syntactic concept of an equation.

(2) The restriction to *strong* nominal sets Σ as generators reflects that the latter are the "free" nominal sets [11], a property crucial for the proof of Theorem 3.8 below. More precisely, letting $\mathcal{P}_f\mathbb{A}$ denote the set of finite subsets of $\mathbb{A}$, the forgetful functor $U\colon \mathbf{Nom} \to \mathbf{Set}^{\mathcal{P}_f\mathbb{A}}$ mapping a nominal set X to the presheaf $S \mapsto \{\, x \in X \;:\; \mathsf{supp}_X(x) \subseteq S \,\}$ has a left adjoint F, and strong nominal sets are exactly the nominal sets of the form FP for $P \in \mathbf{Set}^{\mathcal{P}_f\mathbb{A}}$.

(3) The somewhat technical completeness property cannot be avoided, i.e. a substitution-invariant family of local pseudovarieties is generally incomplete. Indeed, consider the family

$$\mathcal{T} = (\, \mathcal{T}_\Sigma \subseteq \Sigma^* {\downarrow} \mathbf{nMon}_{\mathsf{of}} \,)_{\Sigma \in \mathbf{Nom}_{\mathsf{of,s}}},$$

where $\mathcal{T}_\Sigma$ consists of all Σ-generated orbit-finite nominal monoids $e\colon \Sigma^* \twoheadrightarrow M$ such that e maps each element of Σ^* with a support of size 1 to 1_M.

To see that $\mathcal{T}_\Sigma$ is a filter, suppose that $e\colon \Sigma^* \twoheadrightarrow M$ and $e'\colon \Sigma^* \twoheadrightarrow M'$ are two quotients in $\mathcal{T}_\Sigma$. Form their subdirect product q, viz. the coimage of the morphism $\langle e, e' \rangle\colon \Sigma^* \to M \times M'$. Each $w \in \Sigma^*$ with a support of size 1 is mapped by q to $(e(w), e'(w)) = (1_M, 1_{M'}) = 1_{M \times M'}$. Thus $q \in \mathcal{T}_\Sigma$ and $q \leq e, e'$, i.e. $\mathcal{T}_\Sigma$ is downwards directed. Clearly, $\mathcal{T}_\Sigma$ is also upwards closed.

For substitution invariance, let $h\colon \Delta^* \to \Sigma^*$ be a morphism and $e_\Sigma\colon \Sigma^* \twoheadrightarrow M_\Sigma$ a quotient in $\mathcal{T}_\Sigma$. Then $e_\Sigma \cdot h$ maps each element with a support of size 1 to 1_{M_Σ} since e_Σ does and the equivariant map h does not increase supports. Thus, the coimage of $e_\Sigma \cdot h$ lies in $\mathcal{T}_\Delta$.

Finally, we show that $\mathcal{T}$ is not complete. Fix an arbitrary orbit-finite nominal monoid M containing an element m with $|\,\mathsf{supp}_M\, m| = 1$. Note that $m \neq 1_M$ because 1_M has empty support. Moreover, choose an orbit-finite strong nominal set Σ such that all elements of Σ have least support of size at least 2, and M can be expressed as a quotient $e\colon \Sigma^* \twoheadrightarrow M$. (For instance, one may take $\Sigma = \coprod_{i<k} \mathbb{A}^{\#n}$ where k is the number of orbits of M and $n = \max\{2, |\,\mathsf{supp}_M(x)| : x \in M\}$.) Since all nonempty words in Σ^* have a least support of size at least 2, one has $e \in \mathcal{T}_\Sigma$. For every $\Delta \in \mathbf{Nom}_{\mathsf{of,s}}$ and every quotient $q\colon \Delta^* \twoheadrightarrow M$ in $\mathcal{T}_\Delta$, the set $q^{-1}[\{m\}] \subseteq \Delta^*$ contains no element with least support of size 1, since such elements are mapped by q to $1_M \neq m$. Consequently, q is not support-reflecting. This shows that M is not the codomain of any support-reflecting quotient in $\mathcal{T}_\Delta$.

▶ **Definition 3.7** (Pseudovariety and Weak Pseudovariety)**.** A *pseudovariety of nominal monoids* is a nonempty class $\mathcal{V}$ of orbit-finite nominal monoids closed under finite products, submonoids, and quotient monoids. That is,

(1) for each $M, N \in \mathcal{V}$ one has $M \times N \in \mathcal{V}$;

(2) for each $M \in \mathcal{V}$ and each nominal submonoid $N \rightarrowtail M$ one has $N \in \mathcal{V}$;

(3) for each $M \in \mathcal{V}$ and each nominal quotient monoid $M \twoheadrightarrow N$ one has $N \in \mathcal{V}$.

A *weak pseudovariety of nominal monoids* is a nonempty class of orbit-finite nominal monoids closed under finite products, submonoids, and support-reflecting quotient monoids.

The following result is a special case of the Generalized Variety Theorem [14, Theorem 3.15]. It asserts that equational theories and pseudovarieties are equivalent concepts. Note that (weak) equational theories form a poset ordered by $\mathcal{T} \leq \mathcal{T}'$ iff $\mathcal{T}_\Sigma \leq \mathcal{T}'_\Sigma$ for all $\Sigma \in \mathbf{Nom}_{\mathsf{of,s}}$, where $\mathcal{T}_\Sigma \leq \mathcal{T}'_\Sigma$ holds iff for every $e' \in \mathcal{T}'_\Sigma$ there exists an $e \in \mathcal{T}_\Sigma$ with $e \leq e'$. Similarly, (weak) pseudovarieties of nominal monoids form a poset w.r.t. the inclusion ordering.

▶ **Theorem 3.8.** *(Weak) equational theories and (weak) pseudovarieties of nominal monoids form dually isomorphic complete lattices.*

The isomorphism maps a (weak) equational theory $\mathcal{T}$ to the (weak) pseudovariety $\mathcal{V}(\mathcal{T})$ of all orbit-finite monoids M such that each morphism $h\colon \Sigma^* \to M$ with $\Sigma \in \mathbf{Nom}_{\mathsf{of,s}}$ factorizes through some $e_\Sigma \in \mathcal{T}_\Sigma$. The inverse maps a (weak) pseudovariety $\mathcal{V}$ to the (weak) equational theory $\mathcal{T}(\mathcal{V})$ where $\mathcal{T}(\mathcal{V})_\Sigma$ consists of all quotients $e\colon \Sigma^* \twoheadrightarrow M$ with codomain $M \in \mathcal{V}$.

3.2 The Nominal Eilenberg-Schützenberger Theorem

In addition to their abstract category theoretic characterization in Theorem 3.8, weak pseudovarieties of nominal monoids admit an axiomatic description in terms of sequences of equations, analogous to the classical result of Eilenberg and Schützenberger [7] for pseudovarieties of ordinary monoids. The appropriate concept of equation is as follows:

▶ **Definition 3.9.**

(1) An *equation* is a pair $(s,t) \in X^* \times X^*$, denoted as $s = t$, where X is an orbit-finite strong nominal set. A nominal monoid M *satisfies* $s = t$ if for every equivariant map $h\colon X \to M$ one has $\widehat{h}(s) = \widehat{h}(t)$, where $\widehat{h}\colon X^* \to M$ denotes the unique extension of h to an equivariant monoid morphism.

(2) Given a sequence $E = (s_n = t_n)_{n\in\mathbb{N}}$ of equations (possibly taken over different orbit-finite strong nominal sets X of generators), a nominal monoid M *eventually satisfies* E if there exists an index $n_0 \in \mathbb{N}$ such that M satisfies all the equations $s_n = t_n$ with $n \geq n_0$. We denote by $\mathcal{V}(E)$ the class of all orbit-finite nominal monoids that eventually satisfy E.

▶ **Remark 3.10.** Equations can be presented syntactically as expressions of the form

$$y_1 : S_1, \dots, y_n : S_n \vdash u = v, \tag{1}$$

where $Y = \{y_1, \dots, y_n\}$ is a finite set of variables, $S_1, \dots, S_n \subseteq \mathbb{A}$ are finite sets of atoms, and u, v are words in $(\mathrm{Perm}(\mathbb{A}) \times Y)^*$. A nominal monoid M is said to *satisfy* (1) if for every *variable interpretation*, i.e. every map $h\colon Y \to M$ with $\mathsf{supp}_M(h(y_i)) \subseteq S_i$ for $i = 1, \dots, n$, one has $\overline{h}(u) = \overline{h}(v)$. Here, $\overline{h}\colon (\mathrm{Perm}(\mathbb{A}) \times Y)^* \to M$ is the unique monoid morphism mapping (π, y_i) to $\pi \cdot h(y_i)$. Every equation can be transformed into an expressively equivalent syntactic equation, and vice versa [13, Lemma B.31].

▶ **Theorem 3.11** (Nominal Eilenberg-Schützenberger Theorem)**.** *A class $\mathcal{V}$ of orbit-finite nominal monoids forms a weak pseudovariety iff $\mathcal{V} = \mathcal{V}(E)$ for some sequence E of equations.*

Proof sketch. The proof proceeds along the lines of the one for ordinary monoids [7], although some subtle modifications are required. The "if" direction is a routine verification. For the "only if" direction, let $\mathcal{V}$ be a weak pseudovariety. Using that there are only countably many orbit-finite monoids up to isomorphism, one can construct a sequence $M_0, M_1, M_2, \dots$ of nominal monoids in $\mathcal{V}$ such that each $M \in \mathcal{V}$ is a quotient of all but finitely many M_n's. Let $X_0, X_1, X_2, \dots$ be the sequence of all (countably many) strong orbit-finite nominal sets up to isomorphism, and consider the equivariant congruence relation on X_n^* given by

$$s \equiv_n t \quad \text{iff} \quad M_n \text{ satisfies the equation } s = t.$$

One then shows that the congruence $\equiv_n$ is generated by a finite subset $W_n \subseteq \equiv_n$, and that $\mathcal{V} = \mathcal{V}(E)$ for every sequence E that lists all equations in the countable set $\bigcup_n W_n$. ◀

▶ **Example 3.12.** An orbit-finite nominal monoid M is called *aperiodic* [4, 17] if there exists a natural number $n \geq 1$ such that $x^{n+1} = x^n$ for all $x \in M$. The class of all orbit-finite aperiodic nominal monoids forms a pseudovariety. Taking the set $Y = \{y\}$ of variables, it is not difficult to see that this pseudovariety is specified by the sequence of syntactic equations

$$y : S_n \vdash y^{n+1} = y^n \qquad (n \in \mathbb{N}),$$

where $S_n = \{a_0, a_1, \ldots, a_{n-1}\}$ is the set of the first n atoms in the countably infinite set $\mathbb{A} = \{a_0, a_1, a_2 \ldots\}$ of all atoms, and we write y for $(id, y) \in \mathrm{Perm}(\mathbb{A}) \times Y$.

4 Duality and the Nominal Eilenberg Correspondence

In this section, we establish our nominal version of Eilenberg's variety theorem. It is based on a dual interpretation of the concepts of a (local) pseudovariety of nominal monoids and of an equational theory, introduced in the previous section, under *nominal Stone duality*.

4.1 Nominal Stone Duality

A classical result from duality theory, known as *discrete Stone duality*, states that the category of sets is dually equivalent to the category of complete atomic boolean algebras, i.e. complete boolean algebras in which every non-zero element is above some atom. An analogous duality holds for the category $\mathbf{Nom}_{\mathsf{fs}}$ of nominal sets and finitely supported maps.

▶ **Definition 4.1.** A *nominal complete atomic boolean algebra* (*ncaba*) is a boolean algebra $(B, \vee, \wedge, \neg, \bot, \top)$ in **Nom** such that every finitely supported subset of B has a supremum, and for every element $b \in B \setminus \{\bot\}$ there exists an atom (i.e. a minimal element) $a \in B$ with $a \leq b$. Here, the partial order $\leq$ is defined as usual by $a \leq b$ iff $a \wedge b = a$. We denote by $\mathbf{nCABA}_{\mathsf{fs}}$ the category of ncabas and finitely supported morphisms (i.e. finitely supported maps preserving all the boolean operations and suprema of finitely supported subsets), and by $\mathbf{nCABA}$ the (non-full) subcategory of ncabas and *equivariant* morphisms.

▶ **Theorem 4.2** (Nominal Stone Duality)**.** *The categories* $\mathbf{nCABA}_{\mathsf{fs}}$ *and* $\mathbf{Nom}_{\mathsf{fs}}$ *are dually equivalent. The duality restricts to one between the subcategories* $\mathbf{nCABA}$ *and* $\mathbf{Nom}$*.*

The restricted duality is due to Petrişan [15, Prop. 5.3.11].

▶ **Remark 4.3.**

(1) The equivalence functor $\mathbf{Nom}_{\mathsf{fs}} \xrightarrow{\simeq} \mathbf{nCABA}_{\mathsf{fs}}^{\mathsf{op}}$ maps a nominal set X to the ncaba $\mathcal{P}X$ of finitely supported subsets of X (equipped with the set-theoretic boolean operations), and a finitely supported map $f\colon X \to Y$ to the morphism $f^{-1}\colon \mathcal{P}Y \to \mathcal{P}X$ taking preimages. The inverse equivalence functor $\mathbf{nCABA}_{\mathsf{fs}}^{\mathsf{op}} \xrightarrow{\simeq} \mathbf{Nom}_{\mathsf{fs}}$ maps an ncaba B to the equivariant subset $\mathsf{At}(B)$ of its atoms, with group action restricting the one of B.

(2) The dual equivalence restricts to one between the full subcategories of orbit-finite nominal sets and *atom-finite* ncabas, i.e. ncabas whose set of atoms is orbit-finite. For atom-finite ncabas the property that every finitely supported subset has a supremum is equivalent to the weaker requirement that for every finite set $S \subseteq \mathbb{A}$, every S-orbit has a supremum. Indeed, given a finitely supported subset $X \subseteq B$ (say with finite support $S \subseteq \mathbb{A}$), put $X' := \{\, a \in \mathsf{At}(B) : a \leq x \text{ for some } x \in X \,\}$. Since $\leq$ is an equivariant relation, $X' \subseteq \mathsf{At}(B)$ is a subset with finite support S. Since $\mathsf{At}(B)$ is orbit-finite and thus has only finitely many S-orbits by Lemma 2.1, we can express X' as a finite union $X' = X'_1 \cup \ldots \cup X'_n$ of S-orbits. Using that every element of B is the join of the finitely supported set of all atoms below it, it follows that $\bigvee X = \bigvee X' = \bigvee X'_1 \vee \ldots \vee \bigvee X'_n$, so $\bigvee X$ is a finite join of joins of S-orbits.

4.2 Varieties of Data Languages

For the notion of a *language* over an alphabet $\Sigma \in \mathbf{Nom}$ and the corresponding concept of algebraic recognition by nominal monoids, there are two natural choices: consider equivariant subsets $L \subseteq \Sigma^*$ and their recognition by equivariant monoid morphisms [5, 17], or consider finitely supported subsets $L \subseteq \Sigma^*$ and their recognition by finitely supported monoid morphisms [4]. For our duality-based approach to data languages, it turns out that we need to work with an intermediate concept: finitely supported languages recognizable by equivariant monoid morphisms (see the discussion in Remark 4.12 below). That is, we work with the following

▶ **Definition 4.4.** A *data language* over the alphabet $\Sigma \in \mathbf{Nom}$ is a finitely supported map $L\colon \Sigma^* \to 2$. It is *recognized* by an equivariant monoid morphism $e\colon \Sigma^* \to M$ if there exists a finitely supported map $p\colon M \to 2$ with $L = p \cdot e$. In this case, we also say that M *recognizes* L. A data language is *recognizable* if it recognized by some orbit-finite nominal monoid.

▶ **Remark 4.5.**

(1) Identifying finitely supported maps into 2 with finitely supported subsets, Definition 4.4 can be restated: an equivariant monoid morphism $e\colon \Sigma^* \to M$ *recognizes* a language $L \subseteq \Sigma^*$ if there exists a finitely supported subset $P \subseteq M^*$ with $L = e^{-1}[P]$.

(2) If L is an equivariant recognizable language, then p in Definition 4.4 is also equivariant. Therefore, for equivariant languages we recover the notion of recognition of [5, 17].

(3) If Σ is a finite set (viewed as an orbit-finite discrete nominal set), a data language is just an ordinary formal language over the alphabet Σ. Indeed, the free nominal monoid Σ^* is discrete, and thus every subset of Σ^* is finitely supported. Moreover, every orbit-finite nominal quotient monoid of Σ^* is discrete and finite. Hence, the above notion of language recognition coincides with the classical recognition by finite monoids. In particular, for finite Σ, a recognizable data language is the same as a regular language.

▶ **Example 4.6.** Examples of recognizable data languages over the alphabet $\Sigma = \mathbb{A}$ include (1) every finite or cofinite subset $L \subseteq \mathbb{A}^*$ (see Remark 4.8 below), (2) $a\mathbb{A}^*$ for a fixed atom $a \in \mathbb{A}$, and (3) $\bigcup_{a\in\mathbb{A}} \mathbb{A}^*aa\mathbb{A}^*$. The languages (4) $\{a_1 \dots a_n \ : \ a_i \neq a_j \text{ for } i \neq j\}$, (5) $\bigcup_{a\in\mathbb{A}} a\mathbb{A}^*a\mathbb{A}^*$, and (6) $\mathbb{A}^*a\mathbb{A}^*$ for a fixed atom $a \in \mathbb{A}$ are not recognizable. The equivariant examples (3)–(5) are taken from [4, 5].

In previous work [1] we have given a categorical account of *local varieties of regular languages* [9], i.e. sets of regular languages over a fixed finite alphabet Σ closed under the set-theoretic boolean operations (finite union, finite intersection, complement) and derivatives. This concept can be generalized to data languages. The *derivatives* of a data language $L \subseteq \Sigma^*$ with respect to a word $w \in \Sigma^*$ are given by

$$w^{-1}L = \{\, v \in \Sigma^* \ : \ wv \in L \,\} \quad \text{and} \quad Lw^{-1} = \{\, v \in \Sigma^* \ : \ vw \in L \,\}.$$

Since $\mathsf{supp}(w^{-1}L), \mathsf{supp}(Lw^{-1}) \subseteq \mathsf{supp}(w) \cup \mathsf{supp}(L)$, the derivatives are again data languages.

▶ **Definition 4.7** (Local Variety of Data Languages)**.** Let $\Sigma \in \mathbf{Nom}$. A *local variety of data languages over* Σ is an equivariant set $\mathcal{V}_\Sigma \subseteq \mathcal{P}\Sigma^*$ of recognizable data languages closed under the set-theoretic boolean operations, unions of S-orbits for every finite set $S \subseteq \mathbb{A}$ of atoms (that is, for every $L \in \mathcal{V}_\Sigma$ the language $\bigcup_{\pi\in \mathrm{Perm}_S(\mathbb{A})} \pi \cdot L$ lies in $\mathcal{V}_\Sigma$), and derivatives.

▶ **Remark 4.8.**

(1) If Σ is a finite set (viewed as a discrete nominal set), then by Remark 4.5 a local variety $\mathcal{V}_\Sigma$ consists of regular languages, and the closure under unions of S-orbits is trivial: since $\mathcal{P}\Sigma^*$ is discrete, every S-orbit has a single element. Thus, in this case, a local variety of data languages is precisely a local variety of regular languages.

(2) However, in general the closedness under unions of S-orbits cannot be dropped, as it is neither trivial nor implied by the other conditions. To see this, consider the alphabet $\Sigma = \mathbb{A}$ and the equivariant set $\mathcal{V}_{\mathbb{A}} \subseteq \mathcal{P}\mathbb{A}^*$ of all finite or cofinite subsets of $\mathbb{A}^*$. Note that every finite language $L \subseteq \mathbb{A}^*$ is recognizable: let $n \geq 1$ be an upper bound on the length of words in L, and take the orbit-finite monoid $M = \mathbb{A}^{\leq n} \cup \{0\}$ consisting of all words over $\mathbb{A}$ of length at most n, and a zero element 0. The multiplication $\bullet$ is defined as follows: given $v, w \in \mathbb{A}^{\leq n}$, if the word vw has length at most n, put $v \bullet w = vw$. Otherwise, put $v \bullet w = 0$. Then the equivariant monoid morphism $e\colon \mathbb{A}^* \to M$ extending $a \mapsto a$ recognizes L since $L = e^{-1}[L]$. It follows that also $\mathbb{A}^* \setminus L = e^{-1}[M \setminus L]$. This shows that every language in $\mathcal{V}_{\mathbb{A}}$ is recognizable. Moreover, clearly $\mathcal{V}_{\mathbb{A}}$ is closed under the set-theoretic boolean operations and derivatives. However, the languages $\{a\}$, $a \in \mathbb{A}$, form an orbit in $\mathcal{V}_{\mathbb{A}}$, but their union $\mathbb{A} = \bigcup_{a \in \mathbb{A}} \{a\}$ is not in $\mathcal{V}_{\mathbb{A}}$. Thus $\mathcal{V}_{\mathbb{A}}$ is not a local variety of data languages in the sense of Definition 4.7.

A local variety $\mathcal{V}_\Sigma$ is generally not a subobject of $\mathcal{P}\Sigma^*$ in **nCABA**, because it is not required to be closed under unions of arbitrary finitely supported subsets and also not necessarily atomic as a boolean algebra. However, if the atomic languages in $\mathcal{V}_\Sigma$ form an orbit-finite subset and every language in $\mathcal{V}_\Sigma$ contains some atomic language, then $\mathcal{V}_\Sigma$ is an atom-finite subobject of $\mathcal{P}\Sigma^*$, see Remark 4.32. In this case, we call $\mathcal{V}_\Sigma$ an *atom-finite* local variety.

▶ **Theorem 4.9** (Finite Local Variety Theorem)**.** *The lattice of atom-finite local varieties of data languages over Σ is dually isomorphic to the lattice of Σ-generated orbit-finite monoids.*

The isomorphism maps a Σ-generated orbit-finite monoid $e\colon \Sigma^* \twoheadrightarrow M$ to the atom-finite local variety of all data languages recognized by e.

Proof. By the duality of **Nom** and **nCABA**, orbit-finite equivariant quotients $e\colon \Sigma^* \twoheadrightarrow M$ of Σ^* in **Nom** correspond bijectively to atom-finite subobjects $\mathcal{V}_\Sigma \rightarrowtail \mathcal{P}\Sigma^*$ in **nCABA**, i.e. atom-finite equivariant sets of languages closed under the set-theoretic boolean operations and unions of S-orbits for every finite $S \subseteq \mathbb{A}$. By Proposition 3.1 and the dual equivalence of $\mathbf{Nom}_{\mathsf{fs}}$ and $\mathbf{nCABA}_{\mathsf{fs}}$, the map e represents a nominal quotient monoid of Σ^* if and only if $\mathcal{V}_\Sigma$ is closed under derivatives, i.e. a local variety. The closure under left derivatives is illustrated by the two dual commutative squares below, where the left-hand one lives in $\mathbf{Nom}_{\mathsf{fs}}$ and the right-hand one in $\mathbf{nCABA}_{\mathsf{fs}}$.

$$\begin{array}{ccc} \Sigma^* & \xrightarrow{w\cdot -} & \Sigma^* \\ {\scriptstyle e}\downarrow & & \downarrow{\scriptstyle e} \\ M & \dashrightarrow_{\exists} & M \end{array} \qquad\qquad \begin{array}{ccc} \mathcal{P}\Sigma^* & \xleftarrow{w^{-1}(-)} & \mathcal{P}\Sigma^* \\ {\scriptstyle \subseteq}\uparrow & & \uparrow{\scriptstyle \subseteq} \\ \mathcal{V}_\Sigma & \dashleftarrow_{\exists} & \mathcal{V}_\Sigma \end{array}$$

The elements of $\mathcal{V}_\Sigma$ are precisely the languages recognized by e. Indeed, the former correspond to the morphisms $\mathbb{1} \to \mathcal{V}_\Sigma$ in $\mathbf{nCABA}_{\mathsf{fs}}$, where $\mathbb{1}$ is the free boolean algebra on one generator, the latter to the finitely supported maps $M \to 2$ in $\mathbf{Nom}_{\mathsf{fs}}$, and $\mathbb{1}$ and 2 are dual objects. ◀

Recall that an *ideal* in a poset is a downwards closed and upwards directed subset. For the lattice of local varieties of data languages over Σ (ordered by inclusion), we obtain

▶ **Lemma 4.10.** *The lattice of local varieties of data languages over Σ is isomorphic to the lattice of ideals in the poset of atom-finite local varieties over Σ.*

The isomorphism maps a local variety $\mathcal{V}_\Sigma$ to the ideal of all atom-finite local subvarieties of $\mathcal{V}_\Sigma$. Its inverse maps an ideal $\{\, \mathcal{V}_{\Sigma,i} : i \in I \,\}$ in the poset of atom-finite local varieties to the local variety $\bigcup_{i\in I} \mathcal{V}_{\Sigma,i}$. In order-theoretic terms, the above lemma states that local varieties of data languages form the ideal completion of the poset of atom-finite local varieties. Using Theorem 4.9, Lemma 4.10, and the fact that ideals are dual to filters, we obtain

▶ **Theorem 4.11** (Local Variety Theorem). *For each $\Sigma \in \mathbf{Nom}$, the lattice of local varieties of data languages over Σ is dually isomorphic to the lattice of local pseudovarieties of Σ-generated nominal monoids.*

▶ **Remark 4.12.**

(1) Since data languages are morphisms $L\colon \Sigma^* \to 2$ in $\mathbf{Nom}_{\mathsf{fs}}$, the reader may wonder why we do not entirely work in this category and use monoids with finitely supported multiplication and finitely supported monoid morphisms (rather than the equivariant ones) for the recognition of languages. The reason lies on the dual side: in the proof of Theorem 4.9, we used that equivariant injective maps $\mathcal{V}_\Sigma \rightarrowtail \mathcal{P}\Sigma^*$ can be uniquely identified with equivariant subsets of $\mathcal{P}\Sigma^*$. In contrast, finitely supported injective maps $\mathcal{V}_\Sigma \rightarrowtail \mathcal{P}\Sigma^*$ do not correspond to the finitely supported (or any other kind of) subsets of $\mathcal{P}\Sigma^*$.

(2) Similarly, we cannot restrict ourselves to the category $\mathbf{Nom}$ and only consider equivariant languages $L \subseteq \Sigma^*$ rather than finitely supported ones. Indeed, the Finite Local Variety Theorem then fails: the map sending a Σ-generated orbit-finite monoid $e\colon \Sigma^* \twoheadrightarrow M$ to the set of equivariant languages it recognizes is no longer bijective. To see this, consider the nominal monoids $M = \mathbb{A} \cup \{1\}$ with $a \bullet b = a$ for $a, b \in \mathbb{A}$, and $N = \{0, 1\}$ with $0 \bullet 0 = 0 \bullet 1 = 1 \bullet 0 = 0$. Then the two surjective morphisms $e\colon \mathbb{A}^* \twoheadrightarrow M$, extending $a \mapsto a$, and $f\colon \mathbb{A}^* \twoheadrightarrow N$, extending $a \mapsto 0$, recognize the same equivariant languages, namely $\mathbb{A}^*$, $\mathbb{A}^* \setminus \{\varepsilon\}$, $\{\varepsilon\}$ and $\emptyset$.

In the following, we consider data languages whose alphabet Σ is an orbit-finite strong nominal set (see Remark 3.62). By dualizing the concept of an equational theory, we obtain

▶ **Definition 4.13** (Variety of data languages). A *variety of data languages* is a family

$$\mathcal{V} = (\, \mathcal{V}_\Sigma \subseteq \mathcal{P}\Sigma^* \,)_{\Sigma \in \mathbf{Nom}_{\mathsf{of,s}}}$$

of local varieties of data languages with the following two properties:

(1) *Closedness under preimages.* For each equivariant monoid morphism $h\colon \Delta^* \to \Sigma^*$ with $\Sigma, \Delta \in \mathbf{Nom}_{\mathsf{of,s}}$ and each $L \in \mathcal{V}_\Sigma$, one has $h^{-1}[L] \in \mathcal{V}_\Delta$.

(2) *Completeness.* For each atom-finite local subvariety $\mathcal{V}'_\Sigma \rightarrowtail \mathcal{V}_\Sigma$, there exists an equivariant monoid morphism $h\colon \Sigma^* \to \Delta^*$ and an atom-finite local subvariety $\mathcal{V}'_\Delta \rightarrowtail \mathcal{V}_\Delta$ such that
   a. the map $L \mapsto h^{-1}[L]$ defines a bijection between $\mathcal{V}'_\Delta$ and $\mathcal{V}'_\Sigma$, and
   b. every atomic language $L \in \mathcal{V}'_\Delta$ contains a word $w \in \Delta^*$ with $\mathsf{supp}_{\mathcal{P}\Delta^*}(L) = \mathsf{supp}_{\Delta^*}(w)$.

▶ **Remark 4.14.** Except for the completeness condition, the above concept is analogous to Eilenberg's original notion of a variety of regular languages (i.e. a family of local varieties of regular languages closed under preimages of monoid morphisms). In fact, if Σ is a finite alphabet, and thus $\mathcal{V}_\Sigma$ is just a local variety of regular languages, completeness is trivial: given any finite local subvariety $\mathcal{V}'_\Sigma$ of $\mathcal{V}_\Sigma$, choose $\Delta = \Sigma$, $\mathcal{V}'_\Delta = \mathcal{V}'_\Sigma$, and $h = id\colon \Sigma^* \to \Delta^*$. Then 2a is clear, and 2b holds because each $L \in \mathcal{P}\Sigma^*$ and each $w \in \Sigma^*$ has empty support.

In general, however, the completeness property cannot be dropped. This follows from Remark 3.63 and the observation that the completeness of a variety dualizes to the completeness of the corresponding equational theory (see the proof of Theorem 4.15).

We are ready to state the main result of our paper:

▶ **Theorem 4.15** (Nominal Eilenberg Theorem). *Varieties of data languages and pseudovarieties of nominal monoids form isomorphic complete lattices.*

The isomorphism maps a variety $\mathcal{V}$ of data languages to the pseudovariety $\mathcal{V}$ of all orbit-finite nominal monoids that recognize only languages from $\mathcal{V}$. Its inverse maps a pseudovariety $\mathcal{V}$ to the variety $\mathcal{V}$ of all data languages recognized by some monoid in $\mathcal{V}$.

Proof sketch. We observe that the concept of a variety is dual to that of an equational theory. Indeed, by the Local Variety Theorem 4.11, a family $\mathcal{V} = (\mathcal{V}_\Sigma \subseteq \mathcal{P}\Sigma^*)_{\Sigma \in \mathbf{Nom}_{\mathsf{of,s}}}$ of local varieties of data languages bijectively corresponds to a family $\mathcal{T} = (\mathcal{T}_\Sigma \subseteq \Sigma^* \downarrow \mathbf{nMon}_{\mathsf{of}})_{\Sigma \in \mathbf{Nom}_{\mathsf{of,s}}}$ of local pseudovarieties of nominal monoids. One then shows that (1) $\mathcal{T}$ is substitution-invariant if and only if $\mathcal{V}$ is closed under preimages, and (2) $\mathcal{T}$ is complete if and only if $\mathcal{V}$ is complete. In particular, $\mathcal{T}$ is a theory if and only if $\mathcal{V}$ is a variety of data languages. Since theories correspond to pseudovarieties by Theorem 3.8, this proves the theorem. ◀

5 Adding Expressivity: Regular Data Languages

As recognizing structures for data languages, nominal monoids are of limited expressivity; in particular, they are strictly weaker than deterministic automata in the category of nominal sets [5, 4]. Therefore, we now show how to extend our results for monoid-recognizable data languages and establish a local variety theorem for languages accepted by nominal automata.

▶ **Definition 5.1.** Fix an input alphabet $\Sigma \in \mathbf{Nom}$. A *nominal Σ-automaton* $A = (Q, \delta, q_0)$ consists of a nominal set Q of states, an equivariant transition map $\delta\colon Q \times \Sigma \to Q$, and an initial state $q_0 \in Q$ with empty support. It is called *orbit-finite* if Q is orbit-finite. A *morphism* between nominal automata $A = (Q, \delta, q_0)$ and $A' = (Q', \delta', q_0')$ is an equivariant map $h\colon Q \to Q'$ such that $\delta(h(q), a) = h(\delta(q, a))$ for all $q \in Q$ and $a \in \Sigma$, and $h(q_0) = q_0'$.

The *initial nominal Σ-automaton* is given by $I = (\Sigma^*, \delta, \varepsilon)$ with transition map $\delta(w, a) = wa$ for $w \in \Sigma^*$ and $a \in \Sigma$. It is characterized by the universal property that for every nominal Σ-automaton $A = (Q, \delta, q_0)$, there exists a unique morphism $e_A\colon I \to A$, sending a word $w \in \Sigma^*$ to the state reached from q_0 after reading w. The automaton A is called *reachable* if e_A is surjective. A data language $L \subseteq \Sigma^*$ is *accepted* by A if there exists a finitely supported subset $F \subseteq Q$ with $L = e_A^{-1}[F]$. This corresponds to the usual notion of acceptance of an automaton with final states F: the language L consists of all words $w \in \Sigma^*$ such that A reaches a state in F after reading w. A data language $L \subseteq \Sigma^*$ is called *regular* if there exists an orbit-finite nominal automaton accepting it. In analogy to Proposition 3.1, we get

▶ **Proposition 5.2** (Unary presentation for nominal automata). *For every surjective equivariant map $e\colon \Sigma^* \twoheadrightarrow Q$, the following statements are equivalent:*

(1) *there exists a nominal automaton $A = (Q, \delta, q_0)$ with states Q such that $e = e_A$;*

(2) *the maps $\Sigma^* \xrightarrow{-\cdot w} \Sigma^*$ $(w \in \Sigma^*)$ lift along e, i.e. there exist (necessarily unique) maps $r_w\colon Q \to Q$ such that $e \cdot (- \cdot w) = r_w \cdot e$ for all $w \in \Sigma^*$.*

Define a *local pseudovariety of nominal Σ-automata* to be a class $\mathcal{V}_\Sigma$ of orbit-finite reachable nominal Σ-automata such that (1) $\mathcal{V}_\Sigma$ is closed under quotients (represented by surjective automata morphisms), and (2) for every pair $A, B \in \mathcal{V}_\Sigma$, the reachable part of the product $A \times B$ lies in $\mathcal{V}_\Sigma$. Here, the product of two nominal automata $A = (Q, \delta, q_0)$ and $B = (Q', \delta', q_0')$ is given by $A \times B = (Q \times Q', \delta, (q_0, q_0'))$ with $\delta((q, q'), a) = (\delta(q, a), \delta'(q', a))$ for $(q, q') \in Q \times Q'$ and $a \in \Sigma$, and the reachable part R of $A \times B$ is the coimage $e\colon \Sigma^* \twoheadrightarrow R$ of the unique

morphism $e_{A\times B}\colon \Sigma^* \to A \times B$. Note that a local pseudovariety corresponds precisely to a filter in the poset $\Sigma^*{\downarrow}\Sigma\text{-}\mathbf{nAut}_{\mathsf{of}}$ of orbit-finite reachable nominal Σ-automata. The dual version of this concept is the one of a *local variety of regular data languages over* Σ: an equivariant set $\mathcal{V}_\Sigma \subseteq \mathcal{P}\Sigma^*$ of such languages closed under the set-theoretic boolean operations, unions of S-orbits for every finite set $S \subseteq \mathbb{A}$ of atoms, and *right* derivatives. The following theorem, and its proof, are completely analogous to Theorem 4.11:

▶ **Theorem 5.3** (Local Variety Theorem for Regular Data Languages)**.** *For each* $\Sigma \in \mathbf{Nom}$*, the lattice of local varieties of regular data languages over* Σ *is dually isomorphic to the lattice of local pseudovarieties of nominal* Σ*-automata.*

6 Conclusions and Future Work

We have demonstrated that two cornerstones of the algebraic theory of regular languages, Eilenberg's variety theorem and Eilenberg and Schützenberger's axiomatic characterization of pseudovarieties, can be generalized to data languages recognizable by orbit-finite monoids. Our results are the first of this type for data languages, and thus the present work makes a contribution towards developing a fully fledged algebraic theory of such languages. In a broader sense, the approach taken in this paper can be seen as a further illustration of the power of duality in formal language theory: we believe that without the guidance given by nominal Stone duality, it would have been significantly harder to even come up with the suitable notion of a variety of data languages that makes the nominal Eilenberg correspondence work. The duality-based approach thus adds much conceptual clarity and simplicity. There remain several research questions and interesting directions for future work.

As indicated in section 5, the techniques used in our paper can be adapted without much effort to languages recognized by nominal algebraic structures other than monoids, including deterministic nominal automata. As a first step, we aim to extend the local variety theorem for regular data languages (Theorem 5.3) to a full Eilenberg correspondence. It remains an important goal to further extend our results to more powerful classes of data languages.

Our proof of the (local) Eilenberg correspondence rests on the observation that a local variety of data languages can be expressed as the directed union of its atom-finite subvarieties. From a category theoretic perspective, this suggests that local varieties are formed within the Ind-*completion* (i.e. the free completion under directed colimits) of the category of atom-finite nominal complete atomic boolean algebras. We conjecture that this completion can be described as a category of nominal boolean algebras with joins of S-orbits for each finite set S of atoms. On the dual side, we expect that the Pro-*completion* (i.e. the free completion under codirected limits) of the category of orbit-finite nominal sets consists of some form of nominal Stone spaces. The approach of working with free completions should lead to a topological version of nominal Stone duality similar to the one established by Gabbay, Litak, and Petrişan [8]. More importantly, it might pave the way to the introduction of pro-(orbit-)finite methods for the theory of data languages.

References

1 Jiří Adámek, Stefan Milius, Robert S.R. Myers, and Henning Urbat. Generalized Eilenberg Theorem I: Local Varieties of Languages. In Anca Muscholl, editor, *Proc. Foundations of Software Science and Computation Structures (FoSSaCS)*, volume 8412 of *Lecture Notes Comput. Sci.*, pages 366–380. Springer, 2014.

2 Jiří Adámek, Stefan Milius, Robert S.R. Myers, and Henning Urbat. Varieties of Languages in a Category. In Catuscia Palamidessi, editor, *Proc. 30th Annual Symposium on Logic in Computer Science (LICS'15)*, pages 414–425. IEEE Computer Society, 2015.

3 Jiří Adámek, Stefan Milius, Robert S.R. Myers, and Henning Urbat. Generalized Eilenberg Theorem: Varieties of Languages in a Category. *ACM Trans. Comput. Log.*, 20(1):3:1–3:47, 2019.

4 Mikołaj Bojańczyk. Nominal Monoids. *Theory of Computing Systems*, 53(2):194–222, 2013.

5 Mikołaj Bojańczyk, Bartek Klin, and Sławomir Lasota. Automata theory in nominal sets. *Log. Methods Comput. Sci.*, 10(3:4):44 pp., 2014.

6 Samuel Eilenberg. *Automata, Languages, and Machines Vol. B.* Academic Press, 1976.

7 Samuel Eilenberg and Marcel-Paul Schützenberger. On pseudovarieties. *Advances Math.*, 10:413–418, 1976.

8 Murdoch J. Gabbay, Tadeusz Litak, and Daniela Petrişan. Stone Duality for Nominal Boolean Algebras with New. In A. Corradini, B. Klin, and C. Cîrstea, editors, *Proc. CALCO'11*, volume 6859 of *LNCS*, pages 192–207. Springer, 2011.

9 Mai Gehrke, Serge Grigorieff, and Jean-Éric Pin. Duality and equational theory of regular languages. In *Proc. ICALP'08, Part II*, volume 5126 of *LNCS*, pages 246–257. Springer, 2008.

10 Michael Kaminski and Nissim Francez. Finite-memory automata. *Theoret. Comput. Sci.*, 134(2):329–363, 1994.

11 Alexander Kurz and Daniela Petrisan. On universal algebra over nominal sets. *Mathematical Structures in Computer Science*, 20(2):285–318, 2010.

12 Robert McNaughton and Seymour A. Papert. *Counter-Free Automata (M.I.T. Research Monograph No. 65)*. The MIT Press, 1971.

13 Stefan Milius and Henning Urbat. Equational Axiomatization of Algebras with Structure. Full version; available online at `arXiv:1812.02016`, 2018.

14 Stefan Milius and Henning Urbat. Equational axiomatization of algebras with structure. In *Proc. 22nd International Conference on Foundations of Software Science and Computation Structures*, 2019. To appear.

15 Daniela Petrişan. *Investigations into Algebra and Topology over Nominal Sets*. PhD thesis, University of Leicester, 2012.

16 Andrew M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*. Cambridge University Press, 2013.

17 Gabriele Puppis, Thomas Colcombet, and Clemens Ley. Logics with rigidly guarded data tests. *Log. Methods Comput. Sci.*, 11(3:10):56 pp., 2015.

18 Julian Salamanca. Unveiling Eilenberg-type Correspondences: Birkhoff's Theorem for (finite) Algebras + Duality, February 2017. `arXiv:1702.02822`.

19 Marcel-Paul Schützenberger. On finite monoids having only trivial subgroups. *Inform. and Control*, 8:190–194, 1965.

20 Thomas Schwentick. Automata for XML – A survey. *J. Comput. System Sci.*, 73(3):289–315, 2007.

21 Luc Segoufin. Automata and Logics for Words and Trees over an Infinite Alphabet. In *Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29, 2006, Proceedings*, pages 41–57, 2006.

22 Henning Urbat, Jiří Adámek, Liang-Ting Chen, and Stefan Milius. Eilenberg Theorems for Free. In Kim G. Larsen, Hans L. Bodlaender, and Jean-François Raskin, editors, *Proc. 42nd International Symposium on Mathematical Foundations of Computer Science (MFCS 2017)*, volume 83 of *LIPIcs*. Schloss Dagstuhl, 2017. EATCS Best Paper Award.

23 Henning Urbat and Stefan Milius. Varieties of Data Languages. *CoRR*, abs/1903.08053, 2019. `arXiv:1903.08053`.

How Fast Can We Reach a Target Vertex in Stochastic Temporal Graphs?

Eleni C. Akrida
Department of Computer Science, University of Liverpool, UK
eleni.akrida@liverpool.ac.uk

George B. Mertzios
Department of Computer Science, Durham University, UK
george.mertzios@durham.ac.uk

Sotiris Nikoletseas
Computer Engineering & Informatics Department, University of Patras, and CTI, Greece
nikole@cti.gr

Christoforos Raptopoulos
Computer Engineering & Informatics Department, University of Patras, and CTI, Greece
raptopox@ceid.upatras.gr

Paul G. Spirakis
Department of Computer Science, University of Liverpool, UK
Computer Engineering & Informatics Department, University of Patras, Greece
p.spirakis@liverpool.ac.uk

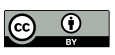
Viktor Zamaraev
Department of Computer Science, Durham University, UK
viktor.zamaraev@durham.ac.uk

Abstract

Temporal graphs are used to abstractly model real-life networks that are inherently dynamic in nature, in the sense that the network structure undergoes discrete changes over time. Given a static underlying graph $G = (V, E)$, a temporal graph on G is a sequence of *snapshots* $\{G_t = (V, E_t) \subseteq G : t \in \mathbb{N}\}$, one for each time step $t \geq 1$. In this paper we study *stochastic temporal graphs*, i.e. stochastic processes $\mathcal{G} = \{G_t \subseteq G : t \in \mathbb{N}\}$ whose random variables are the snapshots of a temporal graph on G. A natural feature of stochastic temporal graphs which can be observed in various real-life scenarios is a *memory effect* in the appearance probabilities of particular edges; that is, the probability an edge $e \in E$ appears at time step t depends on its appearance (or absence) at the previous k steps. In this paper we study the hierarchy of models *memory-k*, $k \geq 0$, which address this memory effect in an *edge-centric* network evolution: every edge of G has its own probability distribution for its appearance over time, *independently* of all other edges. Clearly, for every $k \geq 1$, memory-$(k-1)$ is a special case of memory-k. However, in this paper we make a clear distinction between the values $k = 0$ (*"no memory"*) and $k \geq 1$ (*"some memory"*), as in some cases these models exhibit a fundamentally different computational behavior for these values of k, as our results indicate. For every $k \geq 0$ we investigate the computational complexity of two naturally related, but fundamentally different, *temporal path* (or *journey*) problems: Minimum Arrival and Best Policy. In the first problem we are looking for the *expected arrival time* of a foremost journey between two designated vertices s, y. In the second one we are looking for the expected arrival time of the *best policy* for actually choosing a *particular* s-y journey. We present a detailed investigation of the computational landscape of both problems for the different values of memory k. Among other results we prove that, surprisingly, Minimum Arrival is *strictly harder* than Best Policy; in fact, for $k = 0$, Minimum Arrival is #P-hard while Best Policy is solvable in $O(n^2)$ time.

2012 ACM Subject Classification Mathematics of computing → Graph theory; Mathematics of computing → Graph algorithms; Mathematics of computing → Paths and connectivity problems

Keywords and phrases Temporal network, stochastic temporal graph, temporal path, #P-hard problem, polynomial-time approximation scheme



46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 131; pp. 131:1–131:14

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

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.131

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Related Version A full version of the paper is available at `https://arxiv.org/abs/1903.03636` [2].

Funding This work was supported by the NeST initiative of the EEE/CS School of the University of Liverpool and by the EPSRC grants EP/P020372/1 and EP/P02002X/1.

1 Introduction

Dynamic network analysis, i.e. analysis of networks that change over time, is currently one of the most active topics of research in network science and theory. A common task in this field is to use our prior knowledge of the network link dynamics to answer questions about the behavior of the network over time, e.g. how quickly information can flow through it. Many modern real-life networks are dynamic in nature, in the sense that the network structure undergoes discrete changes over time [31, 36]. Here we deal with the discrete-time dynamicity of the network links (edges) over a fixed set of nodes (vertices). That is, given an underlying static graph G, the network evolution over G is given by the successive appearance or absence of each edge of G at every time step $t = 1, 2, \ldots$. This concept of dynamic network evolution is given by *temporal graphs* [27, 29], which are also known by other names such as *evolving graphs* [6, 20], or *time-varying graphs* [1]. For a recent attempt to integrate existing models, concepts, and results from the distributed computing perspective, see the survey papers [12, 13] and the references therein.

▶ **Definition 1** (Temporal graph). *Given an underlying static graph $G = (V, E)$ on n vertices and m edges, a* temporal graph *on G is a sequence $\mathcal{G} = \{G_t = (V, E_t) : t \in \mathbb{N}\}$ of graphs such that $E_t \subseteq E$ for all $t \in \mathbb{N}$. Every G_t is the* snapshot *of $\mathcal{G}$ at time step t.*

Another way to think about temporal graphs is by assigning *time-labels* on the edges; for example, if an edge e appears in the snapshots G_3, G_5, and G_8, then we equivalently assign to e the set of labels $\lambda(e) = \{3, 5, 8\}$. Due to the vast applicability of temporal graphs, various structural and algorithmic properties of them have been studied extensively, both via theoretical/algorithmic analysis and via empirical simulation-based analysis. In many of these works, one of the central temporal notions is that of a temporal path. A path in the underlying (static) graph G is a *temporal path* (or *journey*) if there exists an increasing sequence of time-labels as one walks along the edges of the path [27, 29]. Motivated by the fact that, due to causality, information in temporal graphs can only flow along sequences of edges that appear in an increasing time order, many temporal graph parameters and optimization problems that have been studied so far are based on the notion of a temporal path and other related notions, e.g. temporal analogs of distance, diameter, connectivity, reachability, and exploration [4, 3, 23, 33, 10, 8, 14, 19, 21, 7, 28, 18]. In addition to temporal paths, recently also various temporal non-path problems have been introduced and algorithmically studied, such as temporal vertex cover [5], temporal coloring [30], and temporal Δ-cliques [38, 24].

Apart from the focus on the various algorithmic problems that one can study on temporal graphs, one can also view temporal graphs through several different levels of knowledge about the actual network evolution. On the one extreme, we may be given the whole temporal graph instance in advance, i.e. the times of appearance and absence of every edge at all times, as it typically happens e.g. when modeling transportation networks. On the other extreme, the temporal graph may be created by an adversary who reveals it to us snapshot-by-snapshot

at every time step. Here we focus on the intermediate knowledge settings, captured by *stochastic temporal graphs*, where the network evolution is given by a probability distribution that governs the appearance of each edge over time.

▶ **Definition 2** (Stochastic temporal graph). *A* stochastic temporal graph *is a stochastic process* $\mathcal{G} = \{G_t : t \in \mathbb{N}\}$ *whose random variables are snapshots* $G_t \subseteq G$ *of an underlying graph* G*. Every instantiation of* $\mathcal{G}$ *is a temporal graph.*

A natural feature of stochastic temporal graphs which can be observed in various real-life scenarios (and which we address in this paper) is that the appearance probability of a particular edge at a given time step t depends on the appearance (or absence) of the same edge at the previous $k \geq 1$ time steps. This "memory effect" can often be observed, among others, in faulty network communication and in mobile, social, and peer-to-peer networks [15, 37, 34]. Several other models of temporal networks which exhibit some sort of probabilistic behavior have been considered in the past, see e.g. [25].

In this paper, we study a hierarchy of models for stochastic temporal graphs which address an *edge-centric* network evolution, i.e. they assign to every edge of the underlying graph G a probability distribution for its appearance over time, independently of all the other edges. The first and most basic model (*memoryless* or *memory*-0) assigns independently to every edge e a probability p_e such that, at every time step, e appears with probability p_e. In the general model (*memory-k*), at every time step the appearance probability of every edge is a function of the history of its appearances/absences in the last $k \geq 1$ time steps. Clearly, for every $k \geq 1$, the memory-$(k-1)$ model is a special case of the memory-k model. However, in this paper we make a clear distinction between the values $k = 0$ (*"no memory"*) and $k \geq 1$ (*"some memory"*), as in some cases these models exhibit a fundamentally different computational behavior for these values of k, as our results indicate (see Section 4).

Our memory-k model, $k \geq 1$, is a direct generalization of the homogeneous version of the memory-1 model that was introduced in a seminal paper by Clementi et al. [16], in which all edges have the same probability distribution for their appearance, based on their own appearance/absence at the previous step. In this homogeneous memory-1 model, Clementi et al. gave upper bounds for the flooding time and they provided tight characterizations of the graphs on which the flooding time is constant [16]. It is worth noting here that Avin et al. [7] studied the completely opposite extreme of our edge-centric evolution; namely they considered a *graph-centric* evolution model where a global probability distribution assigns specific transition probabilities among different snapshots [7]. Between the two extremes of the edge-centric and the graph-centric network evolution models, there exists a whole hierarchy of locally interdependent probabilistic patterns, i.e. probability distributions where the appearance probability of one edge also depends on the appearance of *other edges* over time; such models remain mostly unexplored.

In both our memoryless and memory-k variations of stochastic temporal graphs, we study two fundamental temporal path (i.e. journey) problems that are defined on two designated vertices s and y. Consider a piece of information that is generated at s at time 1, which we would like to send to y via an s-y journey. The *arrival time* of an s-y journey in a realization of a stochastic temporal graph is the time the information reaches y using this journey. A *foremost* s-y journey is one with the smallest arrival time. In the first part of the paper we investigate the complexity of computing the *expected arrival time* of a *foremost* s-y journey. Basu et al. [9] and Nain et al. [32] studied a similar problem but their work is restricted to the simpler cases where the underlying graph is either a path or a grid.

In the second part of the paper we investigate the complexity of computing the arrival time of a *best policy* for actually choosing a particular s-y journey in the stochastic temporal graph. To illustrate this notion of a best policy, assume that some piece of information

is carried by an entity, say Alice. Alice is given as input the parameters of the stochastic temporal graph (i.e. the probabilistic rules on the edges) and, at every time step, she knows the current snapshot and her current location. Based on this information, Alice has to decide at every step for her next action, while her goal is to reach y as quickly as possible on expectation, starting at time 1. In a very inspiring paper, Basu et al. [8] consider this problem in the special case of the memoryless model where all edges have the same probability of appearance at every time, and give a Dijkstra-like polynomial-time algorithm. Special cases of the memory-1 model were considered in [11].

To illustrate the difference between the two problems we study, we make the following analogy. In the first problem (Minimum Arrival) we try to transfer information from s to y using an unbounded number of messages, i.e. we "flood" the stochastic temporal graph with information. Initially the information is stored at s at time 1 and then, at every step, every informed vertex informs all its neighbors as soon as the edge between them becomes available. In the second problem (Best Policy) we try to transfer a package with a tangible good from s to y. Now, at every step we need to decide for the actual route of the package through the network: when an edge appears, should we ship the package along it or rather wait where we currently are? Best Policy is more relevant to real-life applications than Minimum Arrival, where an actual *good* journey needs to be found in real time.

Our contribution. In the first part of the paper, in Section 3, we provide our results for the problem Minimum Arrival, i.e. for computing the expected arrival time of a foremost s-y journey in a stochastic temporal graph. First we prove in Section 3.1 that Minimum Arrival is #P-hard even for the memoryless model (and thus also for the memory-k model, for every $k \geq 1$). The reduction is done from the problem #PP2DNF which counts the number of satisfying assignments in a positive partitioned 2-DNF Boolean formula [35].

Second, we provide in Section 3.2 a non-trivial approximation scheme for Minimum Arrival, based on dynamic programming, for the memoryless model in the case where the underlying graph G is a series-parallel graph with s and y being its terminals. More specifically, it turns out that this is a *Fully Polynomial-Time Approximation Scheme (FPTAS)* whenever the probabilities p_e are lower bounded by $\frac{1}{n^c}$ for some $c \geq 1$. Let X be the random variable that expresses the arrival time of a foremost s-y journey. For every $\varepsilon \in (0, 1]$, our FPTAS gives an algorithm that produces a value μ where $\mathbb{E}(X) - \varepsilon \leq \mu \leq \mathbb{E}(X)$, and runs in polynomial time in both n and $\frac{1}{\varepsilon}$. Although our main result of Section 3.2 concerns series-parallel graphs, we actually present a more general FPTAS approach (see Theorem 11) which is of independent interest and could lead to FPTASs also for more general classes of underlying graphs G.

Third, we present in Section 3.3 a *Fully Polynomial Randomized Approximation Scheme (FPRAS)* for Minimum Arrival in the memory-k model, for every $k \geq 0$, under the assumption that every edge appearance probability is lower bounded by $\frac{1}{n^c}$ for some $c \geq 1$. Let X be the random variable that expresses the arrival time of a foremost s-y journey. For every $\varepsilon \in (0, 1)$, our FPRAS gives a randomized algorithm that produces an estimate $\widetilde{X}$ where $(1-\varepsilon)\mathbb{E}(X) \leq \widetilde{X} \leq (1+\varepsilon)\mathbb{E}(X)$ with probability tending to 1 as $n \to \infty$, and runs in polynomial time in both n and $\frac{1}{\varepsilon}$.

In the second part of the paper, in Section 4, we provide our results for the problem Best Policy, i.e. for computing the expected arrival time of a best policy for choosing a particular s-y journey. Initially we provide in Section 4.1 a dynamic programming algorithm for the memoryless model which runs in $O(n^2)$ time and space. In wide contrast, we prove in Section 4.2 that Best Policy becomes #P-hard for the memory-k model, where $k \geq 3$, again

by providing a reduction from the problem #PP2DNF. Finally, we provide in Section 4.3 a formulation of Best Policy in the memory-k model using the general *Markov Decision Process (MDP)* framework which allows us to devise in Section 4 an exact doubly exponential-time algorithm with running time $O(2^{(kmn+n\log n)\cdot 2^{km}})$. Due to lack of space, many proofs have been omitted; the full proofs of this paper can be found in our technical report [2].

2 Preliminaries

In this paper we consider temporal graphs (see Definition 1) in which the underlying (static) graph $G = (V, E)$ has n vertices and m edges . A subgraph $H = (V, E_H)$ of G, denoted by $H \subseteq G$, is a graph where $E_H \subseteq E$. For every vertex $u \in V$, the *neighborhood* $\Gamma_G(u)$ of u in G is the set of adjacent vertices of u in G. The *closed neighborhood* $\Gamma_G[u]$ also contains vertex u itself, i.e. $\Gamma_G[u] = \Gamma_G(u) \cup \{u\}$. For simplicity of notation we denote $[n] = \{1, 2, \ldots, n\}$ for every $n \in \mathbb{N}$. Furthermore, sometimes we refer to the discrete time steps $t = 1, 2, \ldots$ as *days*. Throughout the paper we consider stochastic temporal graphs that exhibit an edge-centric evolution, i.e. every edge e of G is assigned one probability distribution for its appearance over time, independently of all other edges. We investigate the case where there is a "memory effect" that governs the probability of appearance of every edge over time. We distinguish now the cases where the the memory is zero or non-zero.

Memoryless (or memory-0) model. Every edge $e \in E$ evolves stochastically and independently of other edges as follows: at every time step $t \in \mathbb{N}$, e appears in G_t with probability p_e and is absent with probability $1 - p_e$, independently of any other time step. The numbers $\{p_e : e \in E\}$ are given parameters of the model. We denote this (memoryless) stochastic temporal graph by $\mathcal{G}^{(0)} = (G, \{p_e : e \in E\})$ or simply $\mathcal{G}^{(0)} = (G, \{p_e\})$.

Memory-k model. This model of temporal graphs exhibits stochastic time-dependency of the edges: we assume an initial (arbitrary) sequence of k snapshots, $G_{-k+1}, \ldots, G_{-1}, G_0 \subseteq G$. At every time step $t \geq 1$, every edge e appears independently of all other edges with probability that depends only on (the edge and) the history of appearance of e in the k previous snapshots. At every time step t, this history is a k-bit binary vector, where a 0-entry (resp. 1-entry) on the i-th position denotes absence (resp. appearance) of e in $E_{t-k+i-1}$, for $i = 1, \ldots, k$. Therefore the snapshot G_t is the graph that appears at time $t \geq 1$ as the result of the following experiment: given the history $H_e^{(k)}$ of the appearance of edge $e \in E$ in the last k snapshots, e belongs to E_t independently with probability $p_e(H_e^{(k)})$. We denote the memory-k stochastic temporal graph by $\mathcal{G}^{(k)}$.

In the particular case where $k = 1$, the memory-1 stochastic temporal graph $\mathcal{G}^{(1)}$ is the sequence $\{G_t = (V, E_t) : t \in \mathbb{N}\}$ of snapshots such that $E_t = \{e \in E : X_t^e = 1\}$, where $\{X_t^e\}_{t\in\mathbb{N}}$ is a Markov chain for the edge $e \in E$ with states $\{0, 1\}$ (corresponding to non-appearance and appearance of e, respectively) and probability transition matrix:

$$M_e = \left(\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 1-p_e & p_e \\ 1 & q_e & 1-q_e \end{array}\right), \text{ where } 0 \leq p_e, q_e \leq 1.$$

Using this formalism, p_e (resp. q_e) is the probability that the edge e changes its current state from absence to appearance (resp. from appearance to absence) in the next snapshot. Note here that, setting $p_e = p$ and $q_e = q$ for every edge e, we obtain exactly the well-established *edge-Markovian evolving graph* model introduced by Clementi et al. [16].

2.1 The problems

This work studies two main problems, each under the models of stochastic temporal graphs defined above. To describe both of these problems, let us first recall that information in temporal graphs flows via journeys, i.e. temporal paths.

▶ **Definition 3** (Time-edge). *A time-edge in a temporal graph $\mathcal{G} = \{G_t : t \in \mathbb{N}\}$ is a pair (e, t) such that $e \in E_t$.*

▶ **Definition 4** (Journey / temporal path). *Let $\mathcal{G} = \{G_t : t \in \mathbb{N}\}$ be a temporal graph and s, y be two vertices of G. An s-y* journey *(or an s-y* temporal path*) in $\mathcal{G}$ is a sequence $\big((e_1, t_1), \ldots, (e_x, t_x)\big)$ of time-edges over a path $(e_1, \ldots, e_x)$ from s to y in G, where $t_1 < t_2 < \ldots < t_x$. The* arrival time *of the journey is the time t_x of appearance of its last edge.*

▶ **Definition 5** (Foremost Journey). *A foremost s-y journey in a temporal graph $\mathcal{G}$ is an s-y journey with the minimum arrival time amongst all s-y journeys in $\mathcal{G}$.*

Notice that the arrival time of a foremost s-y journey in a stochastic temporal graph is a random variable, which we henceforth denote by $X(s, y)$. The first problem that we study here is how to compute the expected value of the latter, namely $\mathbb{E}[X(s, y)]$.

▶ **Problem 1** (Minimum Arrival). *Given a stochastic temporal graph on an underlying graph $G = (V, E)$ and two distinct vertices $s, y \in V$, compute the expected value of the arrival time of a foremost s-y journey, i.e. $\mathbb{E}[X(s, y)]$.*

Now suppose that an individual (say Alice) is at day 0 at vertex s and would like to arrive at vertex y through a temporal path as quickly as possible. Denote by s_t the vertex where she is located at time t; then $s_0 = s$. Every day t Alice "wakes up" in the morning and looks at which edges are available in today's snapshot; by only knowing her current position, the history of the last k snapshots, and the input parameters of the stochastic temporal graph (i.e. the probabilistic rules of edge appearance), Alice needs to decide whether:

(a) to stay at the vertex s_t she currently is, or

(b) to use an edge of G_t to move to a neighboring vertex.

That is, s_{t+1} is either equal to s_t or equal to some vertex of $\Gamma_{G_t}(s_t)$.

A natural problem we can study here is to compute the expected arrival time of an s-y journey that Alice can follow, using a *best policy*[1] possible, i.e. a policy (sequence of actions) that minimizes her expected arrival time at y. Notice that the arrival time of the journey suggested to Alice by the best policy is a random variable $Y(s, y)$, whose distribution depends on the specific stochastic temporal graph. In particular, in the memoryless model, the expectation of $Y(s, y)$ depends only on the edges' probabilities of appearance. In the memory-k model, the expectation of $Y(s, y)$ also depends on the initial snapshots $G_{-k+1}, \ldots, G_{-1}, G_0$.

▶ **Problem 2** (Best Policy). *Given a stochastic temporal graph $\mathcal{G}^{(k)}$ on an underlying graph $G = (V, E)$ and two distinct vertices $s, y \in V$, compute $\mathbb{E}_{\mathcal{G}^{(k)}}[Y(s, y)]$.*

In particular, we will write $h(s, y) \stackrel{\text{def}}{=} \mathbb{E}_{\mathcal{G}^{(0)}}[Y(s, y)]$ and $h(s, y, G_0) \stackrel{\text{def}}{=} \mathbb{E}_{\mathcal{G}^{(1)}}[Y(s, y)]$.

[1] We use the term "policy" here (instead of "strategy") since, as we will see later, this problem can be formulated using a Markov Decision Process (MDP).

Difference between the two problems

Before we proceed further, we first give an example illustrating that the problems Minimum Arrival and Best Policy are different. In fact, the gap between the solution to Minimum Arrival and the solution to Best Policy can be arbitrarily large: Consider the graph consisting of vertices s and y and $n-2$ vertex disjoint paths of length 2 between s and y. Assume also that, under the memoryless model, every edge incident to s appears each day with probability 1 and every edge incident to y appears each day independently with probability $n^{-0.9}$. Similarly to the above example of the graph with $n-2$ vertex disjoint paths of length 2, here the expected arrival time of a best policy for Alice is $h(s,y) = 1 + n^{0.9}$. On the other hand, the arrival time of the foremost journey from s to y will be equal to the first day after day 1 on which some edge incident to y appears. But the time needed for the latter to happen follows the geometric distribution with success probability $1-(1-n^{-0.9})^{n-2} = 1 - o(1)$. Therefore, the expected arrival time of the foremost journey will be $\mathbb{E}[X(s,y)] = 2 + o(1)$, i.e. much smaller than $h(s,y) = 1 + n^{0.9}$.

As a final note, the expected arrival time $\mathbb{E}[X(s,y)]$ of the foremost s-y journey is always upper-bounded by the minimum among the expected values of the arrival times of all s-y journeys in the temporal graph. This is actually implied by a more general and well-known lemma in Probability Theory (Fatou's lemma [17, p. 29]) which establishes that the expected value of the minimum among n random variables is upper-bounded by the minimum among all the variables' expectations.

3 Computing the expected minimum arrival time

3.1 Hardness of exact computation in the memoryless model

In this section we show that, even in the memoryless model, Minimum Arrival is #P-hard in both undirected graphs and directed acyclic graphs (DAGs). In the proof of the following theorem, the edges can be treated either as oriented, in which case we obtain the result for DAGs, or as non-oriented, in which case we obtain the result for undirected graphs.

▶ **Theorem 6.** Minimum Arrival *in the memoryless model is #P-hard.*

▶ **Corollary 7.** *For every $k \geq 0$,* Minimum Arrival *in the memory-k model is #P-hard.*

3.2 The FPTAS for the memoryless model on series-parallel graphs

3.2.1 The case of paths

In this section we will consider a stochastic temporal graph $\mathcal{P}^{(0)} = (P = (V,E), \{p_e\})$ with the underlying graph being a path $P = (s = v_0, v_2, \ldots, v_n = y)$.

▶ **Lemma 8.** $\mathbb{E}[X_{\mathcal{P}^{(0)}}(s,y)] = \sum_{e \in E} \frac{1}{p_e}$.

Let us denote by μ the expectation $\mu \stackrel{\text{def}}{=} \mathbb{E}[X_{\mathcal{P}^{(0)}}(s,y)] = \sum_{e \in E} \frac{1}{p_e}$. Note that

$$\mu = \sum_{i=1}^{\infty} \Pr[X_{\mathcal{P}^{(0)}}(s,y) \geq i]. \tag{1}$$

In the remainder of this section we will show that the first $O(\mu \ln \mu)$ terms of sum (1) already give a very good approximation of μ. In our analysis we will use the following bound.

▶ **Theorem 9** ([26]). *Let $X = \sum_{i=1}^{n} X_i$, where $n \geq 1$ and X_i, $i = 1, \ldots, n$, are independent geometric random variables with parameters $p_1, p_2, \ldots, p_n \in (0, 1]$, respectively. Let $\mu = \mathbb{E}[X] = \sum_{i=1}^{n} \frac{1}{p_i}$. Then for any $\lambda \geq 1$, $\Pr[X \geq \lambda\mu] \leq e^{1-\lambda}$.*

▶ **Lemma 10.** *Let ε be a number such that $0 < \varepsilon \leq 1$. Then*

$$\mu - \sum_{i=1}^{\tau} \Pr[X_{\mathcal{P}^{(0)}}(s, y) \geq i] = \sum_{i=\tau+1}^{\infty} \Pr[X_{\mathcal{P}^{(0)}}(s, y) \geq i] < \varepsilon,$$

for every $\tau \geq \mu \left(\ln \frac{\mu}{\varepsilon} + 1\right)$, where $\mu = \mathbb{E}[X_{\mathcal{P}^{(0)}}(s, y)]$.

3.2.2 A general FPTAS approach

While deriving analytically and computing efficiently the exact solution of Minimum Arrival in a path is an easy task (cf. Lemma 8), it does not seem to be trivial for a slight generalization of paths, called *parallel compositions of paths*. A parallel composition of paths is the graph obtained from a collection of disjoint paths $P_1, P_2, \ldots, P_\ell$ with end vertices s_i, y_i, $i = 1, \ldots, \ell$, respectively, by identifying the vertices $s_1, s_2, \ldots, s_\ell$ in a single vertex s, and by identifying the vertices $y_1, y_2, \ldots, y_\ell$ in a single vertex y.

It is not clear whether there exists an efficient procedure for computing the expected arrival time from s to y in a parallel composition of paths, even if the parallel paths are of equal length and all the probabilities of edge appearance are the same. In this section we present a general approach for developing *ε-additive approximation algorithms*[2] for computing the expected arrival time of a foremost journey in special classes of stochastic temporal graphs. In Section 3.2.3 we apply this approach to develop an efficient ε-additive approximation algorithm for the problem on the class of stochastic temporal graphs with underlying graphs being series-parallel graphs, which generalize parallel compositions of paths and graphs in which all simple s-y paths are of the same length.

Throughout the section we denote by $\mathcal{G}^{(0)} = (G = (V, E), \{p_e\})$ a memoryless stochastic temporal graph with n vertices and m edges, and by $s, y \in V$ two distinct vertices in G. Furthermore, we denote by $H = (V, E, w)$ the weighted graph obtained from the underlying graph G by assigning to every edge $e \in E$ the weight $w(e) = \frac{1}{p_e}$.

▶ **Theorem 11.** *Let $c \in \mathbb{N}$ and $\varepsilon \in (0, 1]$. Let $p_e \geq \frac{1}{n^c}$ for every $e \in E$ and suppose that there exists an algorithm A that computes in time $O(f(\ell, n, m))$ the probabilities $\Pr[X_{\mathcal{G}^{(0)}}(s, y) \geq i]$, for all $i = 1, \ldots, \ell$. Then there exists an algorithm B that approximates $\mathbb{E}[X_{\mathcal{G}^{(0)}}(s, y)]$ within the additive factor of ε in time*

$$O\left(f\left(n^{c+1} \ln \frac{n}{\varepsilon}, n, m\right) + n \ln n + m\right).$$

Consequently, if $f(\ell, n, m)$ is a polynomial in variables ℓ, n, and m, then B is an FPTAS on the instance $(\mathcal{G}^{(0)}, s, y)$.

Proof. Let $P = (s = v_0, v_1, \ldots, v_r = y)$ be a minimum weight s-y path in H, and let $\mathcal{P}^{(0)}$ be the stochastic temporal subgraph of $\mathcal{G}^{(0)}$ restricted to the edges of P. For convenience, let us denote $e_i = v_{i-1}v_i$ for every $i = 1, \ldots, r$. Then, by definition and Lemma 8, the weight w^*

[2] A feasible solution is *ε-additive approximate* if it is within ε additive factor from the optimal value. An algorithm is called an *ε-additive approximation algorithm* if it returns an ε-additive approximate solution for any instance.

of P is equal to $\sum_{i=1}^{r} \frac{1}{p_{e_i}} = \mathbb{E}[X_{\mathcal{P}^{(0)}}(s,y)]$. Let $\tau := w^* \left(\ln \frac{w^*}{\epsilon} + 1\right)$. Then, by Lemma 10, we have that

$$\sum_{i=\tau+1}^{\infty} \Pr[X_{\mathcal{G}^{(0)}}(s,y) \geq i] \leq \sum_{i=\tau+1}^{\infty} \Pr[X_{\mathcal{P}^{(0)}}(s,y) \geq i] < \varepsilon,$$

and hence

$$\begin{aligned} \sum_{i=1}^{\tau} \Pr[X_{\mathcal{G}^{(0)}}(s,y) \geq i] \quad &\leq \quad \mathbb{E}[X_{\mathcal{G}^{(0)}}(s,y)] \;=\; \sum_{i=1}^{\infty} \Pr[X_{\mathcal{G}^{(0)}}(s,y) \geq i] \\ &< \quad \sum_{i=1}^{\tau} \Pr[X_{\mathcal{G}^{(0)}}(s,y) \geq i] + \varepsilon, \end{aligned}$$

that is, $\sum_{i=1}^{\tau} \Pr[X_{\mathcal{G}^{(0)}}(s,y) \geq i]$ approximates $\mathbb{E}[X_{\mathcal{G}^{(0)}}(s,y)]$ within the additive factor of ε.

Now we define the desired algorithm B as follows:

1. Construct the graph H and compute the minimum weight w^* of an s-y path in H using Dijkstra's algorithm.
2. Using algorithm A, compute the probabilities $\Pr[X_{\mathcal{G}^{(0)}}(s,y) \geq i], i = 1, \ldots, \tau$, where $\tau = w^* \left(\ln \frac{w^*}{\epsilon} + 1\right)$.
3. Output $\sum_{i=1}^{\tau} \Pr[X_{\mathcal{G}^{(0)}}(s,y) \geq i]$.

The above discussion implies that algorithm B correctly computes the declared approximation of $\mathbb{E}[X_{\mathcal{G}^{(0)}}(s,y)]$. It remains to justify the time complexity. First, Dijkstra's algorithm can be implemented to work in time $O(n \ln n + m)$ [22]. Second, the assumption on p_e's implies that $w^* = O(n^{c+1})$, and hence $\tau = w^* \left(\ln \frac{w^*}{\epsilon} + 1\right) = O\left(n^{c+1} \ln \frac{n}{\epsilon}\right)$. Therefore the assumption of the theorem implies that the last two steps of the algorithm run in time $O\left(f\left(n^{c+1} \ln \frac{n}{\varepsilon}, n, m\right)\right)$, which in turn implies the complexity bound and completes the proof. ◀

3.2.3 The FPTAS for stochastic temporal series-parallel graphs

In the present section we use the approach from Section 3.2.2 to derive a polynomial-time approximation scheme for stochastic temporal series-parallel graphs.

▶ **Theorem 12.** *Let $\varepsilon \in (0,1]$ and let $\mathcal{G}^{(0)} = \{G = (V,E), \{p_e\}\}$ be a stochastic temporal series-parallel graph, where s and y are the terminals of G and $p_e \geq \frac{1}{n^c}$ for every $e \in E$. Then* Minimum Arrival *on $\mathcal{G}^{(0)}$ admits an FPTAS with running time $O\left(m \cdot n^{2c+2} \ln^2 \frac{n}{\varepsilon}\right)$, where $|V| = n$ and $|E| = m$.*

3.3 The FPRAS for general graphs in the memory-k model, $k \geq 0$

In this section, we present our FPRAS for Minimum Arrival in the memory-k model, for every $k \geq 0$, under the assumption that the appearance probability of every edge e is lower bounded by $\frac{1}{n^c}$ for some $c \geq 1$ regardless of the history $H_e^{(k)}$, i.e. $p_e(x) \geq \frac{1}{n^c}$ holds for all $x \in \{0,1\}^k$.

▶ **Theorem 13.** *Let $\varepsilon \in (0,1)$ and let $\mathcal{G}^{(k)}$ be a memory-k stochastic temporal graph with two designated vertices s, y. Furthermore let every edge appearance probability be at least $\frac{1}{n^c}$ for some $c \geq 1$, regardless of the history $H_e^{(k)}$ of e. Then* Minimum Arrival *admits an FPRAS which runs in $O\left(m \frac{n^{5c+8}}{\varepsilon^4} \cdot \log(\frac{n}{\varepsilon})\right)$ time with probability of success at least $1 - \frac{2}{n}$.*

4 Computing the expected arrival time of a best policy

In this section we investigate the computational complexity of our second problem, namely Best Policy.

4.1 A polynomial-time algorithm for the memoryless model

In this section we focus on the memoryless model and we derive a polynomial-time dynamic-programming algorithm for Best Policy. We define for every vertex v the expected arrival time $h(v,y) \stackrel{\text{def}}{=} \mathbb{E}_{\mathcal{G}^{(0)}}[Y(v,y)]$ of the v-y journey suggested to Alice by a best policy (i.e. when Alice starts her journey at vertex v). For simplicity of presentation, throughout Section 4.1 we write $h(v) \stackrel{\text{def}}{=} h(v,y)$.

Assume for now that for all $v \in V$, the value $h(v)$ is given; let $v_1 = y, v_2, \ldots, v_n$ be an ordering of vertices of V in non-decreasing values of h (ties broken arbitrarily), namely $h(v_1) \leq h(v_2) \leq \cdots \leq h(v_n)$. Clearly, $v_1 = y$ and $h(v_1) = h(y) = 0$.

Let s_t be the vertex that Alice occupied at time t and recall that $\Gamma_{G_t}(v)$ is the neighborhood of vertex v in the snapshot G_t, for all $v \in V$ and all $t \in \mathbb{N}$. Notice that, the best strategy of Alice at time $t+1$ is to look at all neighboring vertices of s_t in G_{t+1} and find one with minimum h-value, namely a vertex $u \in \arg\min\{h(v) : v \in \Gamma_{G_{t+1}}(s_t)\}$. If $h(u) \geq h(s_t)$, then Alice has no incentive to change vertex and thus $s_{t+1} = s_t$. Otherwise, if $h(u) < h(s_t)$, then $s_{t+1} = u$.

Therefore, to find the best choice for Alice, it suffices to find the values $h(v), v \in V$. In view of the above, if Alice is on vertex v_i at time 0 (i.e. she is on the i-th best vertex in terms of closeness to y), she will move to the j-th best (with $j < i$) only if an edge appears between v_i and v_j in the next step, and no edge to a vertex better than v_j appears (i.e. no edge between v_i and v_ℓ, $1 \leq \ell \leq j-1$). This happens with probability $Q_{i,j} = p_{\{v_i,v_j\}} \prod_{\ell=1}^{j-1}(1 - p_{\{v_i,v_\ell\}})$, where $\{v_i, v_\ell\}$ denotes the (undirected) edge between v_i and v_ℓ. Additionally, with probability $Q_i = \prod_{\ell=1}^{i-1}(1 - p_{\{v_i,v_\ell\}})$ no edge to a vertex better than v_i will appear, in which case Alice will stay on v_i. Therefore $h(v_i)$ can be recursively computed by $h(v_i) = \sum_{j=1}^{i-1} Q_{i,j} h(v_j) + Q_i h(v_i) + 1$, or equivalently $h(v_i) = \dfrac{\sum_{j=1}^{i-1} Q_{i,j} h(v_j) + 1}{1 - Q_i}$, with initial condition $h(v_1) = 0$. Indeed, the above equation follows by observing that the expected length of the foremost journey to y when Alice is on v_i is equal to $1 + h(v_1)$ with probability $Q_{i,1}$ (which is the probability that an edge between v_i and $v_1 = y$ exists), plus $1 + h(v_2)$ with probability $Q_{i,2}$ (which is the probability that an edge between v_i and the second best vertex v_2 exists, but there is no edge between v_i and v_1), and so on. In general, the above recurrence states that there is no incentive to visit vertices with larger index and also Alice will visit the smallest index vertex v_j for which the edge $\{v_i, v_j\}$ is present (otherwise, if no such edge exists, she will stay on v_i). Using the above recurrence, we can compute all values of $h(v_i)$ by a bottom-up dynamic programming algorithm.

▶ **Theorem 14.** Best Policy *can be optimally computed in the memoryless model in* $O(n^2)$ *time and space.*

4.2 Hardness of computation for the memory-k model, $k \geq 3$

We now show that Best Policy is #P-hard for memory-3 stochastic temporal graphs on directed acyclic graphs, and consequently also for memory $k \geq 3$.

▶ **Theorem 15.** *When the underlying graph is a Directed Acyclic Graph (DAG), it is #P-hard to compute the expected arrival time of the best policy journey in the memory-3 model.*

Proof. We will provide a reduction from the counting problem #PP2DNF which is known to be #P-hard [35]. This problem takes as input a DNF formula $\Phi = \bigvee_{(i,j)\in E} x_i y_j$ on the sets of variables $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$, for some $E \subseteq [n] \times [m]$, and the task is to compute the number ψ of truth assignments that satisfy Φ. We create a directed acyclic graph (DAG) H as follows. First, H has one vertex for each of the variables in $X \cup Y$; then we add two distinct vertices s, y and one other vertex v. For every vertex $x_i \in X$ and every vertex $y_i \in Y$ we add the directed edges (s, x_i) and (y_j, y). Furthermore we add the edge (x_i, y_j) whenever $x_i y_j$ is a clause in Φ. Finally we add the edges (s, v) and (v, y). The construction of H is illustrated in Figure 1.

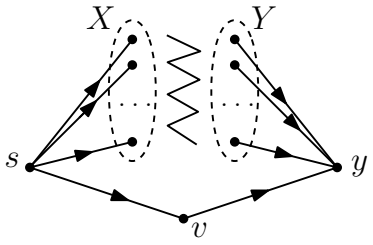


Figure 1 The construction of the DAG H.

Denote by $M = 5 \cdot 2^{n+m}$, and assume that $2^{n+m} \geq 3$ in order to avoid trivialities. All edges (x_i, y_j) appear constantly in H, i.e. they appear at every time step $i \geq 1$ in a memoryless fashion with probability 1. Both edges (s, v) and (v, y) also appear in a memoryless fashion, each of them with probability $\frac{2}{M}$ at every step $i \geq 1$. Moreover, each of the edges (s, x_i) and (y_j, y) appears at each step $i \geq 1$ according to the following table of memory 3. This table has four columns and eight rows. Each column is labeled with the sequence of consecutive time steps $i-3, i-2, i-1$, and i. Each row corresponds to a different triple of appearances of each of the edges in $\{(s, x_i), (y_j, y) : x \in X, y \in Y\}$ at the time steps $i-3, i-2, i-1$ (here 1 means "edge exists" and 0 means "edge does not exist"). At the end of each row there is a pair of numbers $(p, 1-p)$ which denotes that, with the particular history of memory 3, at time step i the edge appears with probability p and it does not appear with probability $1-p$. For simplicity of notation, in the column of time step i, we write "0" and "1" to denote the entries $(0, 1)$ and $(1, 0)$, respectively.

$i-3$	$i-2$	$i-1$	i
0	0	1	0
0	1	0	$(\frac{1}{2}, \frac{1}{2})$
1	0	0	0
0	0	0	0
1	0	1	1
0	1	1	1
1	1	1	1
1	1	0	1

To complete the description of our memory-3 instance, we specify that, in the fictitious initialization snapshots G_{-2}, G_{-1}, G_0, each of the edges (s, x_i) and (y_j, y) appears with probability 0, 0, and 1, respectively, i.e. according to the first row of the above table.

The intuition of this table for the edges (s, x_i) and (y_j, y) is as follows. In the snapshot G_1, none of these edges appears (see the first line of the table). Then, to determine whether each of these edges appears at time step 2 (see the second row of the table), we need to toss an unbiased coin which with probability $\frac{1}{2}$ outputs "appear" and with probability $\frac{1}{2}$ outputs "does not appear". Once this coin has been tossed at time step 2, the status of the edge does not change any more in any subsequent time step $i \geq 3$. That is, if one of the edges (s, x_i) and (y_j, y) appears (resp. does not appear) at time 2, then it appears (resp. does not appear) at all times $i \geq 3$ too. This is easy to be verified by observing the rows 3-7 of the table. Note that the last row of the table is included only for the sake of completeness, as it does not affect the appearance of any edge of H at any time step i.

Let ℓ be the expected s-y arrival time of the best policy in the memory-3 model. Note that, from the above construction of the temporal graph instance, each of the edges (s, x_i) and (y_j, y) appears with probability $\frac{1}{2}$ at all steps $i \geq 2$, while it does not appear at any step $i \geq 2$ with probability $\frac{1}{2}$. Therefore, the probability that there exists a directed temporal path (s, x_i, y_j, y) is equal to $g = \frac{\psi}{2^{n+m}}$, where ψ is the number of satisfying truth assignments of the DNF formula Φ. That is, with probability $1 - g$, there exists no such temporal path from s to y with 3 edges through some vertices x_i and y_j. Furthermore, the expected s-y arrival time through the edges (s, v) and (v, y) is equal to $\frac{M}{2} + \frac{M}{2} = M$. Therefore, since with probability $1 - g$ any policy (also the best one) needs to travel from s to y through vertex v, it follows that $\ell \geq M(1 - g)$.

We now define the following policy: at time step 1 do nothing and just wait for the outcome of the random coin tosses which occur at time step 2. Subsequently, at time step 2 do the following: if there exists a directed temporal path (s, x_i, y_j, y) then follow it, starting at time step 2; otherwise follow the temporal path (s, v, y) which has an expected travel time $\frac{M}{2} + \frac{M}{2} = M$. The expected arrival time of this particular policy is equal to $1 + 3g + M(1 - g)$, and thus it follows that $\ell \leq 1 + 3g + M(1 - g)$. Summarizing, we have:

$$\begin{aligned} M(1-g) &\leq \ell \leq 1 + 3g + M(1-g) \Leftrightarrow \\ 5 \cdot 2^{n+m} - 5\psi &\leq \ell \leq 5 \cdot 2^{n+m} - 5\psi + 3\frac{\psi}{2^{n+m}} + 1. \end{aligned}$$

The first inequality can be written as $2^{n+m} - \frac{\ell}{5} \leq \psi$, while the second one can be written as $\left(1 - \frac{3}{5 \cdot 2^{n+m}}\right)\psi \leq 2^{n+m} - \frac{\ell}{5} + \frac{1}{5}$. Therefore:

$$2^{n+m} - \frac{\ell}{5} \leq \psi \leq \left(1 + \frac{3}{5 \cdot 2^{n+m} - 3}\right)\left(2^{n+m} - \frac{\ell}{5} + \frac{1}{5}\right) \leq 2^{n+m} - \frac{\ell}{5} + \frac{1}{5} + \frac{3}{4},$$

and thus $2^{n+m} - \frac{\ell}{5} \leq \psi \leq 0.95 + 2^{n+m} - \frac{\ell}{5}$. Therefore, knowing the expected value ℓ for the best policy we can derive the exact integer value for ψ in the counting problem #PP2DNF. This completes the #P-hardness reduction. ◀

4.3 An exact algorithm for the memory-k model, $k \geq 1$

In this section we present a doubly exponential-time exact algorithm for computing the best policy for Alice in the memory-k model, where $k \geq 1$. Our results in this section are derived using a Markov Decision Process (MDP) formulation of our problem under the memory-k model.

▶ **Theorem 16.** *Let $k \geq 1$ and $\mathcal{G}^{(k)}$ be a stochastic temporal graph, where the underlying graph G has n vertices and m edges. Then* Best Policy *can be solved on $\mathcal{G}^{(k)}$ in $O(2^{(kmn + n\log n)\cdot 2^{km}})$ time.*

References

1 E. Aaron, D. Krizanc, and E. Meyerson. DMVP: foremost waypoint coverage of time-varying graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, pages 29–41, 2014.

2 E. Akrida, G.B. Mertzios, S. Nikoletseas, C. Raptopoulos, P.G. Spirakis, and V. Zamaraev. *How fast can we reach a target vertex in stochastic temporal graphs?*, 2019. Technical Report available at `arXiv:1903.03636`.

3 E. C. Akrida, L. Gasieniec, G. B. Mertzios, and P. G. Spirakis. Ephemeral networks with random availability of links: The case of fast networks. *Journal of Parallel and Distributed Computing*, 87:109–120, 2016.

4 E. C. Akrida, L. Gasieniec, G. B. Mertzios, and P. G. Spirakis. The complexity of optimal design of temporally connected graphs. *Theory of Computing Systems*, 61(3):907–944, 2017.

5 E.C. Akrida, G.B. Mertzios, P.G. Spirakis, and V. Zamaraev. Temporal vertex cover with a sliding time window. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 148:1–148:14, 2018.

6 A. Anagnostopoulos, J. Lacki, S. Lattanzi, S. Leonardi, and M. Mahdian. Community Detection on Evolving Graphs. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems*, pages 3522–3530, 2016.

7 C. Avin, M. Koucký, and Z. Lotker. How to Explore a Fast-Changing World (Cover Time of a Simple Random Walk on Evolving Graphs). In *International Colloquium on Automata, Languages and Programming, ICALP*, pages 121–132, 2008.

8 P. Basu, A. Bar-Noy, R. Ramanathan, and M. P. Johnson. Modeling and Analysis of Time-Varying Graphs. *CoRR*, abs/1012.0260, 2010.

9 P. Basu, S. Guha, A. Swami, and D. Towsley. Percolation phenomena in networks under random dynamics. In *International Conference on Communication Systems and Networks, COMSNETS*, pages 1–10, 2012.

10 P. Basu, F. Yu, A. Bar-Noy, and D. Rawitz. To Sample or To Smash? Estimating reachability in large time-varying graphs. In *SIAM International Conference on Data Mining*, pages 983–991, 2014.

11 P. Basu, F. Yu, M. P. Johnson, and A. Bar-Noy. Low expected latency routing in dynamic networks. In *IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS*, pages 267–271, 2014.

12 A. Casteigts and P. Flocchini. Deterministic Algorithms in Dynamic Networks: Formal Models and Metrics. Technical report, Defence R&D Canada, April 2013. URL: `https://hal.archives-ouvertes.fr/hal-00865762`.

13 A. Casteigts and P. Flocchini. Deterministic Algorithms in Dynamic Networks: Problems, Analysis, and Algorithmic Tools. Technical report, Defence R&D Canada, April 2013. URL: `https://hal.archives-ouvertes.fr/hal-00865764`.

14 A. Casteigts, P. Flocchini, E. Godard, N. Santoro, and M. Yamashita. On the expressivity of time-varying graphs. *Theoretical Computer Science*, 590:27–37, 2015.

15 A. Clementi, A. Monti, F. Pasquale, and R. Silvestri. Information Spreading in Stationary Markovian Evolving Graphs. *IEEE Transactions on Parallel and Distributed Systems*, 22(9):1425–1432, 2011.

16 A.E.F. Clementi, C. Macci, A. Monti, F. Pasquale, and R. Silvestri. Flooding Time of Edge-Markovian Evolving Graphs. *SIAM Journal on Discrete Mathematics (SIDMA)*, 24(4):1694–1712, 2010.

17 R. Durrett. Probability: Theory and examples, 2011.

18 J. Enright, K. Meeks, G.B. Mertzios, and V. Zamaraev. *Deleting edges to restrict the size of an epidemic in temporal networks*, 2018. Technical Report available at `arXiv:1805.06836`.

19 T. Erlebach, M. Hoffmann, and F. Kammer. On Temporal Graph Exploration. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 444–455, 2015.

20 A. Ferreira. Building a Reference Combinatorial Model for MANETs. *IEEE Network*, 18(5):24–29, 2004.

21 P. Flocchini, B. Mans, and N. Santoro. On the exploration of time-varying networks. *Theoretical Computer Science*, 469:53–68, 2013.

22 M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3):596–615, 1987.

23 S. Henri, S. Shneer, and P. Thiran. On the Delays in Time-Varying Networks: Does Larger Service-Rate Variance Imply Larger Delays? In *ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc*, pages 201–210, 2018.

24 A.-S. Himmel, H. Molter, R. Niedermeier, and M. Sorge. Adapting the Bron-Kerbosch algorithm for enumerating maximal cliques in temporal graphs. *Social Network Analysis and Mining*, 7(1):35:1–35:16, 2017.

25 P. Holme and J. Saramäki, editors. *Temporal Networks*. Springer, 2013.

26 S. Janson. Tail bounds for sums of geometric and exponential variables. *Statistics & Probability Letters*, 135:1–6, 2018.

27 D. Kempe, J. M. Kleinberg, and A. Kumar. Connectivity and inference problems for temporal networks. In *ACM Symp. on Theory of Comp. (STOC)*, pages 504–513, 2000.

28 I. Lamprou, R. Martin, and P. G. Spirakis. Cover Time in Edge-Uniform Stochastically-Evolving Graphs. *Algorithms*, 11(10):149, 2018.

29 G.B. Mertzios, O. Michail, I. Chatzigiannakis, and P.G. Spirakis. Temporal network optimization subject to connectivity constraints. In *International Colloquium on Automata, Languages and Programming (ICALP), Part II*, pages 657–668, 2013.

30 G.B. Mertzios, H. Molter, and V. Zamaraev. Sliding window temporal graph coloring. In *Proceedings of the 33st AAAI Conference on Artificial Intelligence (AAAI)*, 2019. To appear.

31 O. Michail and P.G. Spirakis. Elements of the Theory of Dynamic Networks. *Communications of the ACM*, 61(2):72–72, January 2018.

32 P. Nain, D. Towsley, M. P. Johnson, P. Basu, A. Bar-Noy, and F. Yu. *Computing Traversal Times on Dynamic Markovian Paths*, 2013. Technical Report available at `arXiv:1303.3660`.

33 A. Orda and R. Rom. Distributed Shortest-Path Protocols for Time-Dependent Networks. *Distributed Computing*, 10(1):49–62, 1996.

34 B. Pittel. On Spreading a Rumor. *SIAM Journal on Applied Mathematics*, 47(1):213–223, 1987.

35 J.S. Provan and M.O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12(4):777–788, 1983.

36 N. Santoro. Computing in Time-Varying Networks. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems SSS*, page 4, 2011.

37 C. Scheideler. Models and Techniques for Communication in Dynamic Networks. In *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 27–49, 2002.

38 T. Viard, M. Latapy, and C. Magnien. Computing maximal cliques in link streams. *Theoretical Computer Science*, 609:245–252, 2016.

Distributed Detection of Cliques in Dynamic Networks

Matthias Bonne
Department of Computer Science, Technion, Haifa, Israel
matt.b@cs.technion.ac.il

Keren Censor-Hillel
Department of Computer Science, Technion, Haifa, Israel
ckeren@cs.technion.ac.il

Abstract

This paper provides an in-depth study of the fundamental problems of finding small subgraphs in distributed dynamic networks.

While some problems are trivially easy to handle, such as detecting a triangle that emerges after an edge insertion, we show that, perhaps somewhat surprisingly, other problems exhibit a wide range of complexities in terms of the trade-offs between their round and bandwidth complexities.

In the case of triangles, which are only affected by the topology of the immediate neighborhood, some end results are:

- The bandwidth complexity of 1-round dynamic triangle detection or listing is $\Theta(1)$.
- The bandwidth complexity of 1-round dynamic triangle membership listing is $\Theta(1)$ for node/edge deletions, $\Theta(n^{1/2})$ for edge insertions, and $\Theta(n)$ for node insertions.
- The bandwidth complexity of 1-round dynamic triangle membership detection is $\Theta(1)$ for node/edge deletions, $O(\log n)$ for edge insertions, and $\Theta(n)$ for node insertions.

Most of our upper and lower bounds are *tight*. Additionally, we provide almost always tight upper and lower bounds for larger cliques.

2012 ACM Subject Classification Theory of computation → Distributed computing models; Theory of computation → Distributed algorithms

Keywords and phrases distributed computing, subgraph detection, dynamic graphs

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.132

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Related Version A full version of the paper is available at [5], `http://arxiv.org/abs/1904.11440`.

Acknowledgements This project has received funding from the European Union's Horizon 2020 Research And Innovation Program under grant agreement no. 755839.

1 Introduction

A fundamental problem in many computational settings is to find small subgraphs. In distributed networks it is particularly vital for various reasons, among which is the ability to perform some tasks much faster if, say, triangles do not occur in the underlying network graph (see, eg., [24, 30]).

Finding cliques is a *local* task that trivially requires only a single communication round if the message size is unbounded. However, its complexity may dramatically increase when the bandwidth is restricted to the standard $O(\log n)$ bits, for an n-node network. For example, the complexity of detecting 4-cliques is at least $\Omega(n^{1/2})$ [11]. For triangles, the complexity is yet a fascinating riddle, where only recently the first non-trivial complexities of $\tilde{O}(n^{2/3})$ and $\tilde{O}(n^{3/4})$ have been given for detection and listing, respectively [25], and the current

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 132; pp. 132:1–132:15


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

state-of-the-art is the $\tilde{O}(n^{1/2})$-round algorithm of [10]. For listing, there is an $\tilde{\Omega}(n^{1/3})$ lower bound [25, 28]. The only non-trivial lower bounds for detection say that a single round is insufficient, as given in [1] and extended for randomized algorithms in [18]. In [1] it was also shown that 1-bit algorithms require $\Omega(\log^* n)$ rounds, improved in [18] to $\Omega(\log n)$.

In this paper, we address the question of detecting small cliques in *dynamic* networks of limited bandwidth. We consider a model that captures real-world behavior in networks that undergo changes, such as nodes joining or leaving the network, or communication links between nodes that appear or disappear. Various problems have been studied in many variants of such a setting (see Section 1.3).

The task of finding cliques is unique, in that it is trivial if the bandwidth is not restricted, and it can be easily guaranteed that at the end of each round all nodes have the correct output. This implies that one does not have to wait for *stabilization* and does not have to assume that the network is *quiet* for any positive number of rounds. If, however, the bandwidth is restricted, the solution may not be as simple, although some problems can still be solved even with very small bandwidth.

As a toy example, consider the case of triangle listing when a new edge is inserted to the graph. The endpoints of the inserted edge simply broadcast a single bit that indicates this change to all of their neighbors, and hence if a new triangle is created then its third endpoint detects this by receiving two such indications.

Nevertheless, we show that this simplified case is far from reflecting the general complexities of clique problems in such a dynamic setting. For example, the above algorithm does not solve the problem of *membership-listing* of triangles, in which *each* node should list all triangles that contain it. Indeed, we prove that this stronger variant *cannot* be solved with constant bandwidth, and, in fact, every solution must use at least $\Omega(\sqrt{n})$ bits.

Our contributions provide an in-depth study of various detection and listing problems of small cliques in this dynamic setting, as we formally define and summarize next.

1.1 Our contributions

For a subgraph H we categorize four types of tasks: Detecting an appearance of H in the network, for which it is sufficient that a single node does so, listing all appearances of H, such that every appearance is listed at least by a single node, and their two *membership* variants, membership-detection and membership-listing, for which each node has to detect whether it is a member of a copy of H, or list all copies of H to which it belongs, respectively.

The model is explicitly defined in Section 1.4. In a nutshell, there can be one topology change in every round, followed by a standard communication round among neighboring nodes, of B bits per message, where B is the bandwidth. An algorithm takes r rounds if the outputs of the nodes are consistent with a correct solution after r communication rounds that follow the last change. In particular, a 1-round algorithm is an algorithm in which the outputs are correct already at the end of the round in which the topology change occurred. Hence, 1-round algorithms are very strong, in the sense that they work even in settings in which no round is free of changes. We note that in what follows, all of our algorithms are deterministic and all of our lower bounds hold also for randomized algorithms.

Triangles. Our upper and lower bounds for triangles ($H = K_3$) are displayed in Table 1.

Most of the complexities in this table are shown to be tight, by designing algorithms and proving matching lower bounds. The one exception is for membership detection with edge insertions, for which we show an algorithm that uses $O(\log n)$ bits of bandwidth, but we do not know whether this is tight. However, we also show a 1-round algorithm for this problem

Table 1 The bandwidth complexities of 1-round algorithms for dynamic triangle problems.

	Node deletions	Edge deletions	Edge insertions	Node insertions
Detection/Listing	0	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$
Membership detection	0	$\Theta(1)$	$O(\log n)$	$\Theta(n)$
Membership listing	0	$\Theta(1)$	$\Theta(\sqrt{n})$	$\Theta(n)$

that works with a bandwidth of $O((\Delta \log n)^{1/2})$, where Δ is the maximum degree in the graph, implying that if our logarithmic algorithm is optimal, then a proof for its optimality must exhibit a worst case in which the maximum degree is $\Omega(\log n)$.

A single round is sufficient for solving all clique problems, given enough (linear) bandwidth. Nevertheless, for the sake of comparison, we show that with just one additional communication round, all of the bandwidth complexities in Table 1 drop to $\Theta(1)$, apart from membership listing of triangles, whose bandwidth complexity for r-round algorithms is $\Theta(n/r)$.

Larger Cliques. We also study the bandwidth complexities for finding cliques on $s > 3$ nodes. Here, too, a single round is sufficient for all problems, and the goal is to find the bandwidth complexity of each problem. Some of the algorithms and lower bounds that we show for triangles carry over to larger cliques, but others do not. Yet, with additional techniques, we prove that for cliques of constant size, almost all of the 1-round bandwidth complexities are the same as their triangle counterparts. Due to lack of space, the study of s-cliques is deferred to the full version [5].

Combining types of changes. In most cases, one can obtain an algorithm that handles several types of changes, by a simple combination of the corresponding algorithms. However, intriguingly, sometimes this is not the case. A prime example is when trying to combine edge insertions and node insertions for triangle detection: a node obtaining 1-bit indications of a change from two neighbors cannot tell whether this is due to an edge inserted between them, or due to an insertion of a node that is a neighbor of both. In some of these cases we provide techniques to overcome these difficulties, and use them to adjust our algorithms to cope with more than one type of change.

1.2 Challenges and techniques

Algorithms. The main challenge for designing algorithms is how to convey enough information about the topology changes that occur, despite non-trivial (in particular, sublinear) bandwidth. Consider, e.g., edge insertions. As described, while listing triangles is trivial with a single indication bit, this fails for membership detection or membership listing.

For membership detection we can still provide a very simple algorithm for which a logarithmic bandwidth suffices, by sending the identity of the new edge, and by *helping* neighbors in the triangle to know that they are such. For membership listing even this is insufficient. To overcome this challenge, we introduce a technique for sending and collecting *digests* of neighborhood information. When all digests from a given neighbor have been collected, one can determine the entire neighborhood of this neighbor. The caveat in using such an approach in a straightforward manner is that a node needs to list its triangles with a newly connected neighbor already at the same round in which they connected, and cannot wait to receive all of its neighbor's digests. By our *specific choice* of a digests, we ensure that a newly-connected neighbor has enough information to give a correct output after a single communication round, despite receiving only a single digest from its latest neighbor.

Lower bounds. For the lower bounds, our goal is to argue that in order to guarantee that all nodes give a correct output, each node must receive enough information about the rest of the graph. To do this, we identify sequences of topology changes that exhibit a worst-case behavior, in the sense that a node cannot give a correct output if it receives too little information.

One approach for doing this is to look at a particular node x, and define as many sequences as possible such that the correct output of x is different for each sequence. At the same time, we ensure that the number of messages that x receives from the other nodes is as small as possible. These two requirements are conflicting – if x can have many different outputs, it must have many different neighbors, which implies that it receives many messages with information. Still, we are able to find such a family of sequences for each problem, and we wrap-up our constructions by using counting arguments to prove the desired lower bounds. In some cases, e.g., membership-listing under edge insertions, even this is insufficient, and we construct the sequences such that one *critical* edge affects the output of x, but it is *added last*, so that it conveys as little information as possible.

The above can be seen as another step in the spirit of the *fooling views* framework, which was introduced in [1] for obtaining the first lower bounds for triangle detection under limited bandwidth. After being beautifully extended by [18], our paper essentially gives another indication of the power of the fooling views framework in proving lower bounds for bandwidth-restricted settings.

On the way to constructing our worst-case graph sequences, we prove combinatorial lemmas that show the existence of graphs with certain desired properties. To make our lower bounds apply also for randomized algorithms, we rely on Yao's lemma and on additional machinery that we develop.

1.3 Additional related work

Dynamic distributed algorithms. Dynamic networks have been the subject of much research. A central paradigm is that of *self-stabilization* [12], in which the system undergoes various changes, and after some quiet time needs to converge to a stable state. The model addressed in this paper can be considered as such, but our focus is on algorithms that do not require any quiet rounds (though we also address the gain in bandwidth complexity if the system does allow them, for the sake of comparison). Yet, we assume a single topology change in each round. A single change in each round and enough time to recover is assumed in recent algorithms for maintaining a maximal independent set [2, 3, 9, 14, 23] and matchings [31], and for analyzing amortized complexity [29]. Highly-dynamic networks, in which multiple topology changes can occur in each round are analyzed in [4, 7], and it is an intriguing open question how efficient can subgraph detection be in such a setting. Various other dynamic settings in which the topology changes can be almost arbitrary have been proposed, but the questions that arise in such networks address the flow of information and agreement problems, as they are highly non-structured. A significant paper that relates to our work is [26], which differs from our setting in the bandwidth assumption. Clique detection in the latter model is trivial, and indeed their paper addresses other problems.

Distributed subgraph detection. In the CONGEST model and in related models, where the nodes synchronously communicate with $O(\log n)$ bits, much research is devoted to subgraph detection, e.g., [1, 10, 13, 20, 22, 25, 27, 28].

A related problem is property-testing of subgraphs, where the nodes need to determine, for a given subgraph H, whether the graph is H-free or *far* from being H-free [6, 8, 16, 17, 19, 21].

1.4 Preliminaries

In the dynamic setting, the network is a sequence of graphs. The initial graph, whose topology is known to the nodes, represents the state of the network at some starting point in time, and every other graph in the sequence is either identical to its preceding graph or obtained from it by a single topology change. The network is synchronized, and in each round, each node can send to each one of its neighbors a message of B bits, where B is the bandwidth of the network. Each node has a unique ID and knows the IDs of all its neighbors. We denote by n the number of possible IDs. Hence, n is an upper bound on the number of nodes that can participate in the network at all times.

Note that the nodes do not receive any indication when a topology change occurs. A node can deduce that a change has occurred by comparing the list of its neighbors in the current round and in the previous round. This implies that a node u cannot distinguish between the insertion of an edge uv and the insertion of a node v which is connected to u, as the list of neighbors of u is the same in both cases.

An algorithm can be designed to handle edge insertions or deletions or node insertions or deletions, or any combination of these. We say that an algorithm is an r-round algorithm if the outputs of the nodes after r rounds of communication starting from the last topology change are correct. The (deterministic or randomized) r-round bandwidth complexity of a problem is the minimum bandwidth for which an r-round (deterministic or randomized) algorithm exists. We denote by $\mathsf{MemList}(H)$, $\mathsf{MemDetect}(H)$, $\mathsf{List}(H)$, and $\mathsf{Detect}(H)$, respectively, the problems of membership listing, membership detection, listing and detection of H. The following is a simple observation.

▶ **Observation 1.** *Let r be an integer and let H be a graph. Denote by B_{MemList}, $B_{\mathsf{MemDetect}}$, B_{List} and B_{Detect} the (deterministic or randomized) r-round bandwidth complexities of $\mathsf{MemList}(H)$, $\mathsf{MemDetect}(H)$, $\mathsf{List}(H)$, and $\mathsf{Detect}(H)$, respectively, under some type of topology changes. Then: $B_{\mathsf{Detect}} \leq B_{\mathsf{List}} \leq B_{\mathsf{MemList}}$ and $B_{\mathsf{Detect}} \leq B_{\mathsf{MemDetect}} \leq B_{\mathsf{MemList}}$.*

Due to this observation, our exposition starts with the more challenging task of membership-listing, then addresses membership-detection, and concludes with listing and detection. Within each case, we first provide algorithms for each type of topology change, and then prove lower bounds. Missing proofs are deferred to the full version [5].

2 Triangle problems

2.1 Membership listing

Table 2 summarizes our results for membership listing of triangles.

Table 2 Bandwidth complexities of $\mathsf{MemList}(K_3)$.

	Node deletions	Edge deletions	Edge insertions	Node insertions
$r = 1$	0	$\Theta(1)$	$\Theta(\sqrt{n})$	$\Theta(n)$
$r \geq 2$	0	$\Theta(1)$	$\Theta(1)$	$\Theta(n/r)$

2.1.1 Upper bounds

Our algorithms for node and edge deletions appear in the full version [5]. Note that combining these algorithms in a direct manner *does not* handle both types of changes. We provide a more subtle combination of the two algorithms for doing so within $O(1)$ rounds. Handling edge insertions and node insertions is much more complicated. We start by showing an algorithm that handles edge insertions.

▶ **Theorem 2.1.** *The deterministic* 1*-round bandwidth complexity of* MemList(K_3) *under edge insertions is* $O(\sqrt{n})$.

Proof. Let $N_u(r)$ be the set of neighbours of u on round r. Note that $N_u(r)$ can be encoded as an n-bit string, which indicates, for every node x, whether or not x is a neighbour of u on round r.

The algorithm is as follows. When a new edge uv is inserted on round r, both u and v send to all their neighbors the identity of their new neighbor, denoted by `NEWID`.

In addition, u sends a bitmask of $\lceil\sqrt{n}\rceil$ bits to v, indicating, for every one of the previous $\lceil\sqrt{n}\rceil$ rounds, whether or not one of u's neighbors has sent a `NEWID` to u. We denote this information by $\texttt{LAST}_u$. Node v sends to u the same information.

Finally, u encodes $N_u(r)$ as an n-bit string, denoted $\texttt{ALL}_u(r)$, and starts sending it to v in chunks of $\lceil\sqrt{n}\rceil$ bits per round. This process begins on round r and ends on round $r + \lfloor\sqrt{n}\rfloor$, when the entire string has been sent. During these rounds, u keeps sending `NEWID` to v, and to all its other neighbors, as described above. Node v does the same. It should be noted that this continuous communication between u and v is not intended for allowing them to detect triangles that appear by the insertion of the edge uv, as these are detected immediately due to previous information. Rather, communicating `ALL` allows u and v to detect triangles that appear by other topology changes that may occur in subsequent rounds, as we show below.

Overall, `NEWID` requires $O(\log n)$ bits, while `LAST` and `ALL` require $O(\sqrt{n})$ bits. Thus, the required bandwidth is $O(\sqrt{n})$.

We show that at the end of each round, every node u has enough information to determine, for every two of its neighbors v and w, whether or not the edge vw exists.

First, since only edge insertions are considered, if the edge vw exists in the initial graph, it exists throughout. Also, if vw is inserted when at least one of the edges uv or uw already exists, then u receives this information through `NEWID`. The only other case is when vw does not exist in the initial graph, and is inserted when u is not yet connected to either v or w. That is, the initial graph does not contain any of these three edges, and vw is inserted before the other two. W.l.o.g. assume that uv is inserted before uw. Now, let t_v be the round in which uv is inserted, and let t_w be the round in which uw is inserted.

If $t_w - t_v \leq \lceil\sqrt{n}\rceil$, then when uv is inserted, v sends a `NEWID` to w. Therefore, when uw is inserted, u can determine from $\texttt{LAST}_w$ that in round t_v a neighbor of w has sent a `NEWID` to w. Since the only edge inserted in that round is uv, u determines that v is a neighbor of w. If $t_w - t_v > \lceil\sqrt{n}\rceil$, then, in round t_w, v has already sent the entire string $\texttt{ALL}_v(t_v)$ to u, which indicates that w is a neighbor of v. Therefore, in all cases, u determines whether or not the edge vw exists, as claimed. ◀

The algorithm given for Theorem 2.1 can be extended to handle edge deletions and node deletions as well. Also, if we are promised a quiet round, the problem can be solved with $O(1)$ bits of bandwidth (see full version [5]). Node insertions are harder to handle than the other types of changes. If we are promised $r - 1$ quiet rounds, a simple algorithm exists that uses only $O\left(\frac{n}{r}\right)$ bits of bandwidth. As we show in Section 2.1.2, this is tight.

▶ **Theorem 2.2.** *For every r, the deterministic r-round bandwidth complexity of* MemList(K_3) *under any type of change is $O(n/r)$.*

Proof. The algorithm is as follows: on every round, every node u prepares an n-bit message that specifies its current list of neighbors. Then, it breaks this message into B-bit blocks, where $B = \lceil \frac{n}{r} \rceil$, and sends it to all its neighbors, one block on every round.

Additionally, on every round, u sends to all its neighbors one bit indicating whether or not its list of neighbors has changed on the current round. Whenever the list of neighbors changes, u builds its new list of neighbors, breaks it into blocks, and starts sending it again.

After at most $r-1$ quiet rounds, all nodes are guaranteed to finish sending their list to all their neighbors, thus every node can determine whether any pair of its neighbors are connected to each other or not, as required. ◀

2.1.2 Lower bounds

The 1-round bandwidth complexities for node and edge deletions are clearly tight. Next, we show that handling edge insertions in 1 round requires at least $\Omega(\sqrt{n})$ bits of bandwidth. By Theorem 2.1, this bound is tight. We first prove the following lemma, which shows that a sufficiently dense bipartite graph includes a large enough complete bipartite subgraph. This has the same spirit as the results of Erdös [15], used in [1, 18] for bounding the number of single-bit rounds for detecting triangles, but here the sides can be of different sizes.

▶ **Lemma 2.3.** *For every $\varepsilon \in (0,1)$ there exist $\alpha, \beta, \gamma \in (0,1)$, such that for every bipartite graph $G = (L \cup R, E)$ having at least $(1-\varepsilon)|L||R|$ edges, there are subsets $A \subseteq L$ and $B \subseteq R$, whose sizes are $|A| \geq \alpha \cdot |L|$ and $|B| \geq \beta \cdot \gamma^{|L|} \cdot |R|$, such that $uv \in E$ for every $u \in A$ and $v \in B$ (i.e., the induced subgraph on $A \cup B$ is a complete bipartite graph).*

Proof. Let $\alpha = \frac{1-\varepsilon}{6}$, and let $\mathcal{A}$ be the set of all subsets of L whose size is exactly $\lceil \alpha \cdot |L| \rceil$. For every $A \in \mathcal{A}$, we denote by N_A the set of all vertices in R which are connected to every vertex in A. Consider the sum $S = \sum_{A \in \mathcal{A}} |N_A|$. Let $M = \max\{|N_A| : A \in \mathcal{A}\}$. Then:

$$S \leq \sum_{A \in \mathcal{A}} M = \binom{|L|}{\lceil \alpha \cdot |L| \rceil} \cdot M \tag{1}$$

On the other hand, the sum S can be computed by counting, for every $v \in R$, the number of sets $A \in \mathcal{A}$ such that $v \in N_A$:

$$S = \sum_{A \in \mathcal{A}} |N_A| = \sum_{A \in \mathcal{A}} \sum_{v \in N_A} 1 = \sum_{v \in R} \sum_{\substack{A \in \mathcal{A}: \\ v \in N_A}} 1 = \sum_{v \in R} |\{A \in \mathcal{A} : v \in N_A\}|$$

For $p \in (0,1)$, let k_p be the number of vertices in R whose degree is at least $p \cdot |L|$. For every $v \in R$ whose degree is $d(v)$ we have $|\{A \in \mathcal{A} : v \in N_A\}| = \binom{d(v)}{\lceil \alpha \cdot |L| \rceil}$. Therefore we can bound the above sum

$$S = \sum_{v \in R} \binom{d(v)}{\lceil \alpha \cdot |L| \rceil} \geq \sum_{\substack{v \in R: \\ d(v) \geq p \cdot |L|}} \binom{d(v)}{\lceil \alpha \cdot |L| \rceil} \geq \sum_{\substack{v \in R: \\ d(v) \geq p \cdot |L|}} \binom{\lceil p \cdot |L| \rceil}{\lceil \alpha \cdot |L| \rceil} = k_p \cdot \binom{\lceil p \cdot |L| \rceil}{\lceil \alpha \cdot |L| \rceil}.$$

Combining this with (1) gives $\binom{|L|}{\lceil \alpha \cdot |L| \rceil} \cdot M \geq k_p \cdot \binom{\lceil p \cdot |L| \rceil}{\lceil \alpha \cdot |L| \rceil}$, which implies:

$$M \geq k_p \cdot \binom{\lceil p \cdot |L| \rceil}{\lceil \alpha \cdot |L| \rceil} \Big/ \binom{|L|}{\lceil \alpha \cdot |L| \rceil} \tag{2}$$

We can bound k_p as follows:

$$|E| = \sum_{v\in R} d(v) = \sum_{\substack{v\in R:\\ d(v)\geq p\cdot|L|}} d(v) + \sum_{\substack{v\in R:\\ d(v)<p\cdot|L|}} d(v) \leq \sum_{\substack{v\in R:\\ d(v)\geq p\cdot|L|}} |L| + \sum_{\substack{v\in R:\\ d(v)<p\cdot|L|}} p\cdot|L|$$
$$= k_p\cdot|L| + (|R|-k_p)\cdot p\cdot|L| = (1-p)\cdot k_p\cdot|L| + p\cdot|L|\cdot|R|$$

On the other hand we have $|E| \geq (1-\varepsilon)\cdot|L|\cdot|R|$, and therefore $(1-p)\cdot k_p\cdot|L| + p\cdot|L|\cdot|R| \geq (1-\varepsilon)\cdot|L|\cdot|R|$, which implies $k_p \geq \frac{1-\varepsilon-p}{1-p}\cdot|R|$. Setting $p=\frac{1-\varepsilon}{2}$ gives $k_p \geq \frac{1-\varepsilon}{1+\varepsilon}\cdot|R|$, and substituting this into (2) gives:

$$M \geq \frac{1-\varepsilon}{1+\varepsilon}\cdot\frac{\binom{\lceil p\cdot|L|\rceil}{\lceil\alpha\cdot|L|\rceil}}{\binom{|L|}{\lceil\alpha\cdot|L|\rceil}}\cdot|R|$$

Finally, we bound the binomial fraction on the right-hand size as follows. For simplicity, define $a=|L|, b=\lceil p\cdot|L|\rceil, c=\lceil\alpha\cdot|L|\rceil$. Note that $a\geq b\geq c$. Now:

$$\frac{\binom{b}{c}}{\binom{a}{c}} = \frac{\prod_{i=1}^{c}(i+b-c)}{\prod_{i=1}^{c}(i+a-c)} = \prod_{i=1}^{c}\frac{i+b-c}{i+a-c} = \prod_{i=1}^{c}(1-\frac{a-b}{i+a-c})$$
$$\geq \prod_{i=1}^{c}(1-\frac{a-b}{a-c}) = \prod_{i=1}^{c}\frac{b-c}{a-c} = \left(\frac{b-c}{a-c}\right)^c$$

Therefore:

$$\frac{\binom{\lceil p\cdot|L|\rceil}{\lceil\alpha\cdot|L|\rceil}}{\binom{|L|}{\lceil\alpha\cdot|L|\rceil}} \geq \left(\frac{\lceil p\cdot|L|\rceil - \lceil\alpha\cdot|L|\rceil}{|L|-\lceil\alpha\cdot|L|\rceil}\right)^{\lceil\alpha\cdot|L|\rceil} \tag{3}$$

In order to choose appropriate values for β and γ, we now distinguish between two cases for the graph G. If $|L|\leq\frac{1}{\alpha}$, then $\alpha|L|\leq 1$. Since there must be a vertex in $|L|$ having at least $(1-\varepsilon)|R|$ neighbors in R, the claim holds for every $\beta\leq 1$ and $\gamma\leq 1-\varepsilon$.

If $|L|>\frac{1}{\alpha}$, we can further develop the right hand side of (3) as follows:

$$\left(\frac{\lceil p|L|\rceil - \lceil\alpha|L|\rceil}{|L|-\lceil\alpha|L|\rceil}\right)^{\lceil\alpha|L|\rceil} \geq \left(\frac{p|L|-(\alpha|L|+1)}{|L|-\alpha|L|}\right)^{\alpha|L|+1} = \left(\frac{p-\alpha}{1-\alpha} - \frac{1}{(1-\alpha)|L|}\right)^{\alpha|L|+1}$$
$$> \left(\frac{p-\alpha}{1-\alpha} - \frac{1}{(1-\alpha)\frac{1}{\alpha}}\right)^{\alpha|L|+1} = \left(\frac{p-2\alpha}{1-\alpha}\right)^{\alpha|L|+1}$$
$$= \left(\frac{1-\varepsilon}{5+\varepsilon}\right)^{\alpha|L|+1}$$

To sum it all up, we now have $M > \left(\frac{1-\varepsilon}{5+\varepsilon}\right)^{\alpha\cdot|L|+1}$. Recalling the definition of M, this inequality implies that there exists $A\subseteq L$ whose size is exactly $\lceil\alpha\cdot|L|\rceil$, such that $|N_A| > \left(\frac{1-\varepsilon}{5+\varepsilon}\right)^{\alpha\cdot|L|+1}$, i.e., there are more than $\left(\frac{1-\varepsilon}{5+\varepsilon}\right)^{\alpha\cdot|L|+1}$ vertices in R which are connected to every vertex in A. Therefore, the claim holds for every β and γ such that: $\beta\leq\frac{1-\varepsilon}{5+\varepsilon}, \gamma\leq \left(\frac{1-\varepsilon}{5+\varepsilon}\right)^{\alpha} = \left(\frac{1-\varepsilon}{5+\varepsilon}\right)^{\frac{1-\varepsilon}{6}}$. Thus, given $\varepsilon\in(0,1)$, the following values of α,β,γ satisfy the claim for every G:

$$\alpha = \frac{1-\varepsilon}{6}, \beta = \frac{1-\varepsilon}{5+\varepsilon}, \gamma = \min\left\{1-\varepsilon, \left(\frac{1-\varepsilon}{5+\varepsilon}\right)^{\frac{1-\varepsilon}{6}}\right\}.$$

◀

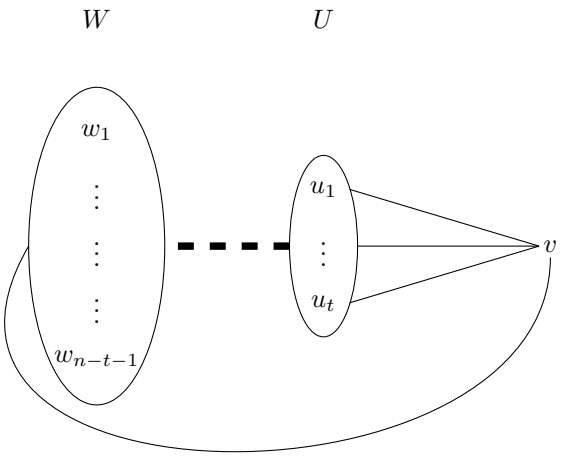


Figure 1 The lower bound sequence for MemList(K_3) with edge insertions. The connections between W and U are chosen. Then, v is connected to all of U. The single edge from v to W is added last.

▶ **Theorem 2.4.** *The randomized* 1*-round bandwidth complexity of* MemList(K_3) *under edge insertions is* $\Omega(\sqrt{n})$.

Proof. Let A be a (randomized) 1-round algorithm that solves MemList(K_3) under edge insertions using bandwidth B with error probability ε. Let t be a parameter to be defined later, and consider a tripartite graph with n nodes as in Figure 1. Let $\mathcal{C}$ be the set of all possible bipartite graphs with vertex sets W and U. Note that $|\mathcal{C}| = 2^{t(n-t-1)}$. For every $C \in \mathcal{C}$ and every $w \in W$, we define a sequence of changes $S_{C,w}$ as follows. We start with no edges. Then, we insert edges between U and W to get the bipartite graph C. During the next t rounds, we connect v to every $u \in U$, one by one. Finally, we insert the edge vw. In the end of $S_{C,w}$, after 1 additional round of communication, node v must output a list of all triangles containing v. By construction, this list is uniquely determined by the set of neighbors of w in U. Thus, node v needs to know the set of all neighbors of w after 1 additional round of communication.

We assume that the output of v is correct with probability at least $1 - \varepsilon$. Therefore, by Yao's lemma, there exists a deterministic algorithm A' that solves the same problem correctly for at least $1 - \varepsilon$ of all inputs. We define a bipartite graph with node sets $\mathcal{C}$ and W, such that the edge $w - C$ exists if and only if the output of v is correct for the sequence $S_{C,w}$.

We have $|W| = n - t - 1$ and $|\mathcal{C}| = 2^{t(n-t-1)}$. Also, by assumption, this graph contains at least $(1 - \varepsilon)$ of all possible edges. Therefore, by Lemma 2.3, there exists $W_1 \subseteq W$ of size at least $\alpha \cdot (n - t - 1)$ and $\mathcal{C}_1 \subseteq \mathcal{C}$ of size at least $\beta \cdot \gamma^{n-t-1} \cdot 2^{t(n-t-1)}$, for some $\alpha, \beta, \gamma \in (0, 1)$ that depend only on ε, such that the output of v is correct for the sequence $S_{C,w}$ for every $C \in \mathcal{C}_1$ and every $w \in W_1$.

Now, consider the input of node v during any sequence $S_{C,w}$. During the first stage of building the bipartite graph C, v is isolated and receives no input. Then, it receives a set of messages from the nodes in U, and finally one additional message from w. Note that the messages v receives from the nodes in U depend only on C, and not on w, since the nodes in U cannot know the identity of w until the final round of communication.

Now, on every round, every node in U can send to v any of 2^B possible messages. Altogether, during the entire sequence, the nodes of U send to v a set of $\frac{t(t+3)}{2}$ messages. Hence, the number of possible inputs from the nodes of U to v is $2^{B\cdot\frac{t(t+3)}{2}}$. Therefore, there exists $\mathcal{C}_2 \subseteq \mathcal{C}_1$ of size at least $2^{-B\cdot\frac{t(t+3)}{2}} \cdot |\mathcal{C}_1|$, such that v receives the same input from all nodes in U for every sequence $S_{C,w}$ for $C \in \mathcal{C}_2$ and every $w \in W_1$. Thus, we have:

$$|\mathcal{C}_2| \geq 2^{-B\frac{t(t+3)}{2}}|\mathcal{C}_1| \geq 2^{-B\frac{t(t+3)}{2}}\beta\gamma^{n-t-1}2^{t(n-t-1)} = \beta\gamma^{n-t-1}2^{t(n-t-1)-B\frac{t(t+3)}{2}} \tag{4}$$

On the other hand, we can bound the size of $\mathcal{C}_2$ by considering the number of possible neighbors of w in any $C \in \mathcal{C}_2$, for every $w \in W$. Recall that during the sequence $S_{C,w}$, v receives only one message from w. Also, for every $w \in W_1$, v must determine the set of all neighbors of w in U. Since there are only 2^B possible inputs v can receive from w, every $w \in W_1$ can have at most 2^B possible sets of neighbors in any $C \in \mathcal{C}_2$.

Every $w \in W \setminus W_1$ can have any subset of U as its set of neighbors, hence it can have at most 2^t possible sets of neighbors. Now, since every $C \in \mathcal{C}_2$ is uniquely determined by set of neighbors of every $w \in W$, we have:

$$|\mathcal{C}_2| \leq \left(\prod_{w\in W_1} 2^B\right)\left(\prod_{w\in W\setminus W_1} 2^t\right) = 2^{B\cdot|W_1|}\cdot 2^{t\cdot(|W|-|W_1|)} = 2^{(B-t)\cdot|W_1|+t(n-t-1)} \tag{5}$$

Combining (4) and (5) gives: $\beta\cdot\gamma^{n-t-1}\cdot 2^{t(n-t-1)-B\cdot\frac{t(t+3)}{2}} \leq 2^{(B-t)\cdot|W_1|+t(n-t-1)}$, and setting $t = \lceil\sqrt{n}\rceil$ gives, with some algebraic manipulations, $B \geq \Omega(\sqrt{n})$. ◀

Finally, for node insertions, we show that every r-round algorithm must use at least $\Omega(n/r)$ bits of bandwidth, which is tight by Theorem 2.2.

▶ **Theorem 2.5.** *For every r, the randomized r-round bandwidth complexity of $\mathsf{MemList}(K_3)$ under node insertions is $\Omega\left(\frac{n}{r}\right)$.*

Proof. Let A be a (randomized) r-round algorithm that solves $\mathsf{MemList}(K_3)$ under node insertions using bandwidth B with error probability ε. We show that $r \cdot B = \Omega(n)$.

Let u be any node, and let $\mathcal{C}$ be the set of all possible graphs on the other $n-1$ nodes. For every $C \in \mathcal{C}$ we define the sequence S_C as follows:

- Start with an empty graph (no nodes and no edges).
- During $n-1$ rounds, insert one node on each round and connect it to the nodes which have been already inserted, according to the edges in C. After $n-1$ rounds, the graph is identical to the graph C.
- On round n insert u and connect to all the other nodes.

After $r-1$ quiet rounds u needs to output the list of all triangles that contain u. Since u is connected to all the other nodes, this implies that u needs to know the graph C. For every $C \in \mathcal{C}$, the output of u is guaranteed to be correct for the sequence S_C with probability at least $(1-\varepsilon)$. Therefore, by Yao's lemma, there exists a deterministic algorithm, A', that guarantees that the output of u is correct for at least $(1-\varepsilon)$ of all sequences. That is, there exists a subset $\mathcal{C}_1 \subseteq \mathcal{C}$, whose size is at least $(1-\varepsilon)\cdot|\mathcal{C}|$, such that A' guarantees the correct output of u for sequences S_C for all $C \in \mathcal{C}_1$.

Now, the number of possible graphs on $n-1$ nodes is $2^{\binom{n-1}{2}}$, hence the size of $\mathcal{C}_1$ is at least $(1-\varepsilon)\cdot 2^{\binom{n-1}{2}}$. Since A' is deterministic, and u needs to distinguish correctly between all possible $C \in \mathcal{C}_1$, this implies that the input u receives from its neighbors must have at

least $(1-\varepsilon)\cdot 2^{\binom{n-1}{2}}$ possible values. Every neighbor of u can send any of 2^B messages on every round, thus the number of possible inputs u can receive on a single round is $2^{B\cdot(n-1)}$. Therefore, during r rounds of communication, the number of possible inputs to u is $2^{r\cdot B\cdot(n-1)}$.

Combining the above we get that $2^{r\cdot B\cdot(n-1)} \geq (1-\varepsilon)\cdot 2^{\binom{n-1}{2}}$, which can be simplified to $r\cdot B \geq \frac{n}{2} + \frac{\log(1-\varepsilon)}{n-1}$, and it follows that $r\cdot B = \Omega(n)$. ◀

2.2 Membership detection

Table 3 summarizes the results of this subsection.

Table 3 Bandwidth complexities of MemDetect(K_3).

	Node deletions	Edge deletions	Edge insertions	Node insertions
$r=1$	0	$\Theta(1)$	$O(\log n)$	$\Theta(n)$
$r\geq 2$	0	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$

2.2.1 Upper bounds

The upper bounds for node deletions and edge deletions follow from Observation 1. The following shows that edge insertions can be handled with $O(\log n)$ bits of bandwidth, which can be extended to handle node/edge deletions as well (see full version [5]).

▶ **Theorem 2.6.** *The deterministic* 1*-round bandwidth complexity of* MemDetect(K_3) *under edge insertions is* $O(\log n)$.

Proof. The algorithm works as follows. On every round, every node sends to all its neighbors an indication whether or not it got a new neighbor on the current round, along with the ID of the new neighbor (if any). We denote this information by `NEWID`. Note that with just this information, for every pair of adjacent edges $u-v-w$, at least one of u and w know that these two edges exist (specifically, the first one connected to v knows that the other one is also connected to v).

Additionally, every node u sends to every neighbor v one bit, indicating whether u knows that v is part of some triangle. We denote this bit by `ACCEPT`.

Suppose the edge uv is inserted and creates at least one triangle uvw. As explained above, in this case at least one of u and v knows that the triangle exists, therefore it sends `ACCEPT` $= 1$ to the two other nodes. Thus all three nodes know that they are part of a triangle. It follows that on each round all nodes can determine whether they are in a triangle. ◀

If we only consider sequences of graphs with bounded degree Δ, and only allow edge insertions, the complexity of Theorem 2.6 can be improved to $O(\sqrt{\Delta\log n})$, using an algorithm similar to that of Theorem 2.1, but with $O(\sqrt{\Delta\log n})$ bits instead of $O(\sqrt{n})$, as follows. On every round, every node sends to all its neighbors an indication whether or not it has a new neighbor. We denote this information by `NEW`. Additionally, whenever a new edge uv is inserted, the following happens:

- u sends to v an indication whether or not it knows about a triangle that contains v.
- For every round i within the last $\left\lceil\sqrt{\Delta\log n}\right\rceil$ rounds, u sends to v an indication whether or not it has had a new neighbor on round i.
- For every round i within the last $\left\lceil\sqrt{\Delta\log n}\right\rceil$ rounds, u sends to v an indication whether or not any of its neighbors has sent `NEW` $= 1$ on round i.

- u computes the list of IDs of all its current neighbors, and starts sending it to v, $\left\lceil \sqrt{\Delta \log n} \right\rceil$ bits on every round. Since u has at most Δ neighbors, after $O(\sqrt{\Delta \log n})$ rounds v has the complete list. Note that by the time this process completes, the list may not be up-to-date.

All of this requires a $O(\sqrt{\Delta \log n})$ bandwidth, and it can be shown, similarly to Theorem 2.1, that this allows every node to give the correct output in all cases. The discussion of what happens when a quiet round is promised, is deferred to the full version [5].

2.2.2 Lower bounds

For node deletions and edge deletions, we have trivial constant lower bounds, which are tight, as shown above. For edge insertions, we have shown a general algorithm that uses $O(\log n)$ bits, and also an algorithm that uses $O(\sqrt{\Delta \log n})$ bits for graphs with bounded degree Δ. The latter algorithm implies that showing a lower bound of B bits for the bandwidth complexity of this problem would require looking at sequences of graphs with degree at least $\Omega(\frac{B^2}{\log n})$. In particular, in order to show that the algorithm of Theorem 2.6 is optimal, one has to consider sequences of graphs with degree at least $\log n$. Other than that, it remains an open question whether or not the algorithm of Theorem 2.6 can be improved.

▶ **Open Question 1.** *What is the* 1*-round bandwidth complexity of* MemDetect(K_3) *under edge insertions?*

Finally, for node insertions, the following theorem shows a lower bound of $\Omega(n)$ bits.

▶ **Theorem 2.7.** *The randomized* 1*-round bandwidth complexity of* MemDetect(K_3) *under node insertions is* $\Omega(n)$.

Proof. Let A be a (randomized) 1-round algorithm that solves MemDetect(K_3) under node insertions using bandwidth B with error probability ε. Fix $x \in V$, let $U = V \setminus \{x\}$, and let $\mathcal{C}$ be the set of all possible graphs on the nodes of U. For every $C \in \mathcal{C}$ and every $u, v \in U$, define the sequence $S_{C,u,v}$ as follows:

- Start with an empty graph (no nodes and no edges).
- During $n-1$ rounds, insert one node of U on each round and connect it to the nodes which have been already inserted, according to the edges in C. After $n-1$ rounds, the graph is identical to the graph C.
- On round n insert x and connect it to u and v.

Then, x should output 1 iff the edge uv exists in C. By our assumption, for every C and every $u, v \in U$, the output is correct with probability at least $1 - \varepsilon$. Note that during the final round, x receives only one message from each of u and v. Also, during this final round, u and v do not know the identity of the other neighbor of x. Hence, the message received from u depends only on the identity of u and the graph C, and not on the identity of v, and the message received from v depends only on the identity of v and the graph C.

Now, consider the following experiment for a given graph $C \in \mathcal{C}$. First, we run the sequence $S_{C,u,v}$ for some $u, v \in U$, and stop just before the last round, in which x is connected to u and v. Then, every node $w \in U$ generates a message to be sent to x, as if it has been connected to x on the final round. For every $w \in U$, let m_w be the message generated by w (note that m_w is a random variable).

Note again, that the messages generated by the nodes of U depend only on the graph C, and not on the other nodes that may have been connected to x on the last round. Therefore, for every $u, v \in U$, given the messages generated by u and v, x should be able to determine, with probability at least $1 - \varepsilon$, whether or not the edge uv exists in C.

Next, for every pair of nodes $u, v \in U$, let I_{uv} be the output of x given the two messages m_u and m_v. Note that for nodes u, v, w, the variables I_{uv} and I_{uw} are not necessarily independent. Now, let C' be the graph on the nodes of U, in which the edge uv exists if and only if $I_{uv} = 1$. We consider C' to be the result of the experiment.

Let p_C denote the probability that at the end of the experiment we have $C' = C$. Since, for every $u, v \in U$, the value of I_{uv} corresponds to the edge uv in the graph C with probability at least $1 - \varepsilon$, we have $p_C \geq (1-\varepsilon)^{\binom{n-1}{2}}$. Summing the above for every $C \in \mathcal{C}$ we get:

$$\sum_C p_C \geq |\mathcal{C}| \cdot (1-\varepsilon)^{\binom{n-1}{2}} = (2-2\varepsilon)^{\binom{n-1}{2}} \tag{6}$$

On the other hand, consider the set of messages generated by the nodes of U at the end of the first stage of the experiment. For every possible set of messages M generated by the nodes of U during the first stage of the experiment, denote by $\phi_C(M)$ the probability for generating exactly the messages of M. Also, for every $C' \in \mathcal{C}$, let $\Psi_M(C')$ denote the probability for the result of the experiment to be equal to C', given the set of generated messages M. We have:

$$\sum_C p_C = \sum_C \sum_M \phi_C(M) \cdot \Psi_M(C) \leq \sum_C \sum_M \Psi_M(C) = \sum_M \sum_C \Psi_M(C) = \sum_M 1 = |\mathcal{M}|$$

where $\mathcal{M}$ is the set of all possible values of M. Since every message has exactly 2^B possible values, the number of possible sets of $(n-1)$ messages is $|\mathcal{M}| = 2^{B(n-1)}$, and hence $\sum_C p_C \leq 2^{B(n-1)}$. Combining this with (6) gives $2^{B(n-1)} \geq (2-2\varepsilon)^{\binom{n-1}{2}}$. After some simplifications we get the desired bound:

$$B \geq \log(2-2\varepsilon) \cdot \frac{n-2}{2} = \Omega(n)$$ ◀

▶ Remark 2.8. The discussion of $\mathsf{List}(K_3)$ and $\mathsf{Detect}(K_3)$, as well as the treatment of s-cliques, appear in the full version [5].

References

1. Amir Abboud, Keren Censor-Hillel, Seri Khoury, and Christoph Lenzen. Fooling Views: A New Lower Bound Technique for Distributed Computations under Congestion. *CoRR*, abs/1711.01623, 2017. arXiv:1711.01623.
2. Sepehr Assadi, Krzysztof Onak, Baruch Schieber, and Shay Solomon. Fully dynamic maximal independent set with sublinear update time. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 815–826, 2018. doi:10.1145/3188745.3188922.
3. Sepehr Assadi, Krzysztof Onak, Baruch Schieber, and Shay Solomon. Fully Dynamic Maximal Independent Set with Sublinear in n Update Time. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1919–1936, 2019. doi:10.1137/1.9781611975482.116.
4. Philipp Bamberger, Fabian Kuhn, and Yannic Maus. Local Distributed Algorithms in Highly Dynamic Networks. *CoRR*, abs/1802.10199, 2018. arXiv:1802.10199.
5. Matthias Bonne and Keren Censor-Hillel. Distributed Detection of Cliques in Dynamic Networks. *CoRR*, abs/1904.11440, 2019. arXiv:1904.11440.
6. Zvika Brakerski and Boaz Patt-Shamir. Distributed discovery of large near-cliques. *Distributed Computing*, 24(2):79–89, 2011. doi:10.1007/s00446-011-0132-x.
7. Keren Censor-Hillel, Neta Dafni, Victor I. Kolobov, Ami Paz, and Gregory Schwartzman. Fast and Simple Deterministic Algorithms for Highly-Dynamic Networks. *CoRR*, abs/1901.04008, 2019.

8 Keren Censor-Hillel, Eldar Fischer, Gregory Schwartzman, and Yadu Vasudev. Fast distributed algorithms for testing graph properties. *Distributed Computing*, 32(1):41–57, 2019. doi:10.1007/s00446-018-0324-8.

9 Keren Censor-Hillel, Elad Haramaty, and Zohar S. Karnin. Optimal Dynamic Distributed MIS. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*, pages 217–226, 2016. doi:10.1145/2933057.2933083.

10 Yi-Jun Chang, Seth Pettie, and Hengjie Zhang. Distributed Triangle Detection via Expander Decomposition. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 821–840, 2019. doi:10.1137/1.9781611975482.51.

11 Artur Czumaj and Christian Konrad. Detecting Cliques in CONGEST Networks. In *32nd International Symposium on Distributed Computing, DISC 2018, New Orleans, LA, USA, October 15-19, 2018*, pages 16:1–16:15, 2018. doi:10.4230/LIPIcs.DISC.2018.16.

12 Shlomi Dolev. *Self-Stabilization*. MIT Press, 2000.

13 Andrew Drucker, Fabian Kuhn, and Rotem Oshman. On the power of the congested clique model. In *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, pages 367–376, 2014. doi:10.1145/2611462.2611493.

14 Yuhao Du and Hengjie Zhang. Improved Algorithms for Fully Dynamic Maximal Independent Set. *CoRR*, abs/1804.08908, 2018. arXiv:1804.08908.

15 P. Erdös. On extremal problems of graphs and generalized graphs. *Israel Journal of Mathematics*, 2(3):183–190, September 1964. doi:10.1007/BF02759942.

16 Guy Even, Orr Fischer, Pierre Fraigniaud, Tzlil Gonen, Reut Levi, Moti Medina, Pedro Montealegre, Dennis Olivetti, Rotem Oshman, Ivan Rapaport, and Ioan Todinca. Three Notes on Distributed Property Testing. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, pages 15:1–15:30, 2017. doi:10.4230/LIPIcs.DISC.2017.15.

17 Guy Even, Reut Levi, and Moti Medina. Faster and Simpler Distributed Algorithms for Testing and Correcting Graph Properties in the CONGEST-Model. *CoRR*, abs/1705.04898, 2017. arXiv:1705.04898.

18 Orr Fischer, Tzlil Gonen, Fabian Kuhn, and Rotem Oshman. Possibilities and Impossibilities for Distributed Subgraph Detection. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures, SPAA 2018, Vienna, Austria, July 16-18, 2018*, pages 153–162, 2018. doi:10.1145/3210377.3210401.

19 Orr Fischer, Tzlil Gonen, and Rotem Oshman. Distributed Property Testing for Subgraph-Freeness Revisited. *CoRR*, abs/1705.04033, 2017. arXiv:1705.04033.

20 Pierre Fraigniaud, Pedro Montealegre, Dennis Olivetti, Ivan Rapaport, and Ioan Todinca. Distributed Subgraph Detection. *CoRR*, abs/1706.03996, 2017. arXiv:1706.03996.

21 Pierre Fraigniaud, Ivan Rapaport, Ville Salo, and Ioan Todinca. Distributed Testing of Excluded Subgraphs. In *Distributed Computing - 30th International Symposium, DISC 2016, Paris, France, September 27-29, 2016. Proceedings*, pages 342–356, 2016. doi:10.1007/978-3-662-53426-7_25.

22 Tzlil Gonen and Rotem Oshman. Lower Bounds for Subgraph Detection in the CONGEST Model. In *21st International Conference on Principles of Distributed Systems, OPODIS 2017, Lisbon, Portugal, December 18-20, 2017*, pages 6:1–6:16, 2017. doi:10.4230/LIPIcs.OPODIS.2017.6.

23 Manoj Gupta and Shahbaz Khan. Simple dynamic algorithms for Maximal Independent Set and other problems. *CoRR*, abs/1804.01823, 2018. arXiv:1804.01823.

24 Juho Hirvonen, Joel Rybicki, Stefan Schmid, and Jukka Suomela. Large Cuts with Local Algorithms on Triangle-Free Graphs. *Electr. J. Comb.*, 24(4):P4.21, 2017. URL: http://www.combinatorics.org/ojs/index.php/eljc/article/view/v24i4p21.

25 Taisuke Izumi and François Le Gall. Triangle Finding and Listing in CONGEST Networks. In *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*, pages 381–389, 2017. doi:10.1145/3087801.3087811.

26 Michael König and Roger Wattenhofer. On Local Fixing. In *OPODIS*, volume 8304 of *Lecture Notes in Computer Science*, pages 191–205. Springer, 2013.

27 Janne H. Korhonen and Joel Rybicki. Deterministic Subgraph Detection in Broadcast CONGEST. In *21st International Conference on Principles of Distributed Systems, OPODIS 2017, Lisbon, Portugal, December 18-20, 2017*, pages 4:1–4:16, 2017. doi:10.4230/LIPIcs.OPODIS.2017.4.

28 Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. On the Distributed Complexity of Large-Scale Graph Computations. In *SPAA*, pages 405–414. ACM, 2018.

29 Merav Parter, David Peleg, and Shay Solomon. Local-on-Average Distributed Tasks. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 220–239, 2016. doi:10.1137/1.9781611974331.ch17.

30 Seth Pettie and Hsin-Hao Su. Distributed coloring algorithms for triangle-free graphs. *Inf. Comput.*, 243:263–280, 2015. doi:10.1016/j.ic.2014.12.018.

31 Shay Solomon. Fully Dynamic Maximal Matching in Constant Update Time. In *Proceedings of the IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 325–334, 2016. doi:10.1109/FOCS.2016.43.

On Approximate Pure Nash Equilibria in Weighted Congestion Games with Polynomial Latencies

Ioannis Caragiannis
University of Patras & CTI "Diophantus", Patras, Greece
caragian@ceid.upatras.gr

Angelo Fanelli
CNRS (UMR-6211), Caen, France
angelo.fanelli@unicaen.fr

Abstract

We consider the problem of the existence of natural improvement dynamics leading to approximate pure Nash equilibria, with a reasonable small approximation, and the problem of bounding the efficiency of such equilibria in the fundamental framework of weighted congestion game with polynomial latencies of degree at most $d \geq 1$.
In this work, by exploiting a simple technique, we firstly show that the game always admits a d-approximate potential function. This implies that every sequence of d-approximate improvement moves by the players always leads the game to a d-approximate pure Nash equilibrium. As a corollary, we also obtain that, under mild assumptions on the structure of the players' strategies, the game always admits a constant approximate potential function. Secondly, by using a simple potential function argument, we are able to show that in the game there always exists a $(d + \delta)$-approximate pure Nash equilibrium, with $\delta \in [0, 1]$, whose cost is $2/(1 + \delta)$ times the cost of an optimal state.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory; Theory of computation → Convergence and learning in games

Keywords and phrases Congestion games, approximate pure Nash equilibrium, potential functions, approximate price of stability

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.133

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Funding *Angelo Fanelli*: ANR-14-CE24-0007-01 *"CoCoRICo-CoDec"*

1 Introduction

Among other solution concepts, the notion of pure Nash equilibrium plays a central role in Game Theory. Pure Nash equilibria in a game characterize situations in which no player has an incentive to unilaterally deviate from the current situation in order to achieve a higher payoff. Unfortunately, it is well known that there are games that do not have pure Nash equilibria. Furthermore, even in games where the existence of pure Nash equilibria is guaranteed, these equilibria could be very inefficient compared to solutions dictated by a central authority. Such negative results significantly question the importance of pure Nash equilibria as solution concepts that characterize the behavior of rational players.

One way to overcome the limitations of the non-existence and inefficiency of pure Nash equilibria is to consider a relaxation of the stability constraints. This relaxation leads to the concept of *approximate* pure Nash equilibrium. This concept characterizes situations where no player can *significantly improve* her payoff by unilaterally deviating from her current strategy. Approximate pure Nash equilibria can accommodate small modeling inaccuracies due to uncertainty (e.g., see the arguments in [5]), therefore they may be more desirable as

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 133; pp. 133:1–133:12


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

solution concepts in practical decision-making settings. Beside mere existence and efficiency, approximate pure Nash equilibria are also an appealing alternative solution concept from a computational point of view [2, 4].

In this work, we investigate the existence and efficiency of approximate pure Nash equilibria in the context of the weighted congestion game. This game is a general framework which models situations in which a group of agents compete for the use of a set of shared resources. In the following, we introduce weighted congestion games and give a formal statement of the problems we address. We conclude this section with a discussion about the current literature and a detailed presentation of our contribution.

Weighted congestion games. In a weighted congestion game, players compete over a set of resources. Each player has a positive weight. Each resource incurs a latency to all players using it; this latency depends on the total weight (congestion) of the players that use the resource according to a resource-specific, non-negative, and non-decreasing latency function. Among a given set of strategies (over sets of resources), each player aims to select one selfishly, trying to minimize her individual total cost, i.e., the sum of the latencies on the resources in her strategy. Typical examples include weighted congestion games in networks, where the network links correspond to the resources and each player has alternative paths that connect two nodes as strategies. Now, let us describe the game more formally.

The WEIGHTED CONGESTION GAME with polynomial latencies of degree at most $d \in \mathbb{Z}^{\geq 1}$ is a collection of instances, denoted by $\texttt{WCG}(d)$, of the form $G = \langle N, E, (w_i)_{i\in N}, (S_i)_{i\in N}, (a_e, k_e)_{e\in E}\rangle$, where $N = \{1, 2, \ldots, |N|\}$ is the set of *players*, $E = \{1, 2, \ldots, |E|\}$ is the set of *resources*, $w_i \in \mathbb{R}^{>0}$ is the *weight* of player i, $S_i \subseteq 2^E$ is the set of *strategies* of player i and $(a_e, k_e) \in \mathbb{R}^{>0} \times \{1, 2, \ldots, d\}$ are the *coefficient* and the *degree* of resource $e \in E$ respectively, which encode the *latency function* $\ell_e : 2^N \mapsto \mathbb{R}^{\geq 0}$ associated with e, mapping every subset of players $P \subseteq N$ to the non-negative real $a_e \left(\sum_{j\in P} w_j\right)^{k_e}$.
The set of *states* of G is denoted by $\mathcal{S}(G) = S_1 \times S_2 \times \ldots \times S_{|N|}$. For every state $\mathbf{s}$ we refer to its i-th component, that is the strategy played by player i in $\mathbf{s}$, by $\mathbf{s}(i)$. For every state $\mathbf{s}$ and resource $e \in E$, we denote by $L_e(\mathbf{s})$ the set of players using resource e in $\mathbf{s}$, i.e., $L_e(\mathbf{s}) = \{j \in N : e \in \mathbf{s}(i)\}$. We refer to the sum of the weights of all the players in $L_e(\mathbf{s})$ as the *congestion* of e in $\mathbf{s}$. For every state $\mathbf{s} \in \mathcal{S}(G)$, the *cost* incurred by player $i \in N$ in $\mathbf{s}$ is $c_i(\mathbf{s}) = \sum_{e\in\mathbf{s}(i)} \ell_e(L_e(\mathbf{s}))$, while the *social cost* of $\mathbf{s}$ is the weighted sum of the players' costs, i.e., $C(\mathbf{s}) = \sum_{i\in N} w_i c_i(\mathbf{s})$. Notice that, by summing over the resources instead of the players, $C(\mathbf{s})$ can be rewritten as $C(\mathbf{s}) = \sum_{e\in E} a_e \left(\sum_{j\in L_e(\mathbf{s})} w_j\right)^{k_e+1}$. Let $w_{\max}$ be the greatest weight in G, we say that G is *mildly congested* if $\ell_e(L_e(\mathbf{s})) \geq (k_e + 1)w_{\max}$, for every resource $e \in E$ and state $\mathbf{s} \in \mathcal{S}(G)$.

Preliminary definitions. We now introduce concepts that are necessary to formally state our problems and present our results.
Let us consider an instance $G = \langle N, E, (w_i)_{i\in N}, (S_i)_{i\in N}, (a_e, k_e)_{e\in E}\rangle$ of $\texttt{WCG}(d)$. For every state $\mathbf{s} \in \mathcal{S}(G)$ and every $s \in S_i$, we denote by $[\mathbf{s}_{-i}, s]$ the new state obtained from $\mathbf{s}$ by setting the i-th component, that is the strategy of i, to s and keeping all the remaining components unchanged, i.e., $[\mathbf{s}_{-i}, s](i) = s$. The transition from $\mathbf{s}$ to $[\mathbf{s}_{-i}, s]$ is called a *move* of player i from state $\mathbf{s}$. For $\alpha \geq 1$, we say that a transition from $\mathbf{s}$ to $[\mathbf{s}_{-i}, s]$ is an *α-improvement move* for i if $\alpha c_i([\mathbf{s}_{-i}, s]) \leq c_i(\mathbf{s})$ (it is a *strictly* α-improvement if $\alpha c_i([\mathbf{s}_{-i}, s]) < c_i(\mathbf{s})$). For $\alpha \geq 1$,

we say that a state-value function $\Gamma : \mathcal{S}(G) \mapsto \mathbb{R}^{\geq 0}$ is an *α-approximate potential function* for G if it strictly decreases at every strictly α-improvement move; formally, $\Gamma([\mathbf{s}_{-i}, s]) < \Gamma(\mathbf{s})$ whenever $\alpha c_i([\mathbf{s}_{-i}, s]) < c_i(\mathbf{s})$. If G admits an α-approximate potential function Γ then every sequence of strictly α-improvement moves leads to a *local optimum* of Γ, that is a state in which no further strictly improvement move can be performed; such a state is called α-approximate pure Nash equilibrium. Formally, for $\alpha \geq 1$, we say that a state $\mathbf{s} \in \mathcal{S}(G)$ is an *α-approximate pure Nash equilibrium* if, for every player $i \in N$ and every strategy $s \in S_i$, we have $c_i(\mathbf{s}) \leq \alpha c_i([\mathbf{s}_{-i}, s])$. If $\alpha = 1$ we simply refer to $\mathbf{s}$ as a *pure Nash equilibrium* rather than a 1-approximate pure Nash equilibrium. For $\alpha \geq 1$, we denote by $\mathcal{E}_\alpha(G) \subseteq \mathcal{S}(G)$ the set of all α-approximate pure Nash equilibria of G. Every state $\mathbf{s} \in \mathcal{S}(G)$ minimizing the social cost is called a *social optimum*. We denote by $\mathcal{OPT}(G)$ the set of social optima of G, i.e., $\mathcal{OPT}(G) = \arg\min_{\mathbf{s}\in\mathcal{S}(G)} C(\mathbf{s})$. Let $\mathbf{o} \in \mathcal{OPT}(G)$ be any social optimum of G, we define the *α-approximate price of stability* of G as $\mathrm{PoS}_\alpha(G) = \min_{\mathbf{e}\in\mathcal{E}_\alpha(G)} \frac{C(\mathbf{e})}{C(\mathbf{o})}$.

Problem statement. In this work we consider the problem of the existence of natural improvement dynamics leading to approximate pure Nash equilibria, with a reasonable small approximation, and the problem of bounding the efficiency of such equilibria in the fundamental framework of weighted congestion game with polynomial latencies of degree at most $d \geq 1$. We formally state such problems as follows.

(i) Existence of Convergent sequences of α-improvement moves. In this problem, given any instance G of $\texttt{WCG}(d)$, we seek for a reasonable small $\alpha \geq 1$ for which any sequence of α-improvement moves in G converges to an α-approximate pure Nash equilibrium. This would be equivalent to say that G admits an α-approximate potential function, whose value decreases at every α-improvement move and whose local optima coincide with α-approximate pure Nash equilibria.

(ii) Bounding the approximate price of stability. In this problem, given any instance G of $\texttt{WCG}(d)$, which admits an α-approximate pure Nash equilibrium, we aim at bounding the α-approximate price of stability of G.

Related work. The unweighted congestion game (i.e., when all players have unit weight) has been widely studied in the literature. Rosenthal [16] proved that this game admits a 1-approximate potential function. This immediately implies that every sequence of 1-improvement moves by the players leads the game to a pure Nash equilibrium. For the weighted congestion game, a 1-approximate potential function exists only when the latencies are linear or exponential [10, 13, 15]. For polynomial latencies (of constant maximum degree strictly higher than 1), pure Nash equilibria may not exist [10, 11, 14]. In general, for arbitrary latencies, the problem of deciding whether a given instance of weighted congestion game has a pure Nash equilibrium is NP-hard [9]. Caragiannis et al. [2] proved that every instance of weighted congestion game with polynomial latencies of degree at most d admits a $d!$-approximate potential function. This results has been subsequently improved by Hansknecht et al. [12]; they showed that every instance of weighted congestion game with polynomial latencies admits a $(d+1)$-approximate potential function. The potential function they proposed is a Rosenthal-like potential function. Roughly speaking, they obtained an approximate potential as follows. For each resource, they chose an appropriate fixed ordering of the players. Then, for each resource separately, they computed a discrete integral. Specifically, they sum up the latency of the resource after introducing the first player multiplied

with the weight of the first player, the latency after introducing the first two players multiplied with the weight of the second player, and so on, i.e., $w_1\ell_e(\{w_1\}) + w_2\ell_e(\{w_1 + w_2\}) + \ldots$. The potential obtained depends on the way the players have been initially ordered. The authors showed that, the potential function providing the best approximation for polynomial latencies, that is $d+1$, is the one obtained by ordering the players in non-decreasing order in terms of their weights, i.e., $w_1 \leq w_2 \leq \ldots$.

For the 1-approximate price of stability, for the unweighted game, there exists a bound of 1.577 for linear latencies [8, 3] and a bound of $\Theta(d)$ for polynomial latencies [6]. The 1-approximate price of stability for the weighted game with polynomial latencies has been recently investigated in [7]; they provided a lower bound of $\Omega(d/\log d)^{d+1}$, matching the upper bound in [1]. The authors also showed bounds to the α-approximate price of stability. Specifically, they proved that for the weighted congestion game with polynomial latencies and weights ranging in $[1, W]$, there exists an α-approximate pure Nash equilibrium, for any α in the range $[\frac{2(d+1)W}{2W+d+1}, d+1]$, whose cost is $1 + (\frac{d+1}{\alpha} - 1)W$ the cost of any optimal state. Their proof exploits a potential function called Faulhaber's potential.

Our contribution. Concerning the first problem, we show (Theorem 3) that every instance of $\texttt{WCG}(d)$ admits in general d-approximate potential functions. This implies that every sequence of d-approximate improvement moves by the players always leads the game to a d-approximate pure Nash equilibrium. This result is achieved by using the technique formalised in Theorem 2 and the class of state-value functions Φ_γ defined in Definition 1. Essentially, while Definition 1 provides a simple interesting class of candidate potential functions, Theorem 2 gives a local condition to each resource to determine the approximation guarantee achieved by a given state-value function. So, by exploiting Theorem 2, in Theorem 3 we are able to show that the class Φ_γ contains d-approximate potential functions and, more generally, $(d+\delta)$-approximate potential functions, for every $\delta \geq 0$. We remark that, our potential functions are substantially different from the potential function proposed in [12]. In fact, while the potential in [12] is obtained in a Rosenthal-like fashion, by ordering the players and summing their costs, by assuming that each player is affected only by the congestion caused by preceding players in the ordering, our potential, more simply, is obtained by a suitable scaling of the coefficients of the polynomials. As a matter of fact, our potentials, despite their simplicity, provide an approximation of d instead of $d+1$; although it is worth noticing that, for very small degrees, the two approaches provide the same approximation guarantee. As a corollary of Theorem 3, we also show (Corollary 4) that the social optimum of an instance of $\texttt{WCG}(d)$ is always a $(d+1)$-approximate pure Nash equilibrium, as it has already been observed in [7]. More importantly, Theorem 3 implies that, as state by Corollary 5, every mildly congested instance of $\texttt{WCG}(d)$ always admits a $\frac{e}{e-1}$-approximate potential function, where $\boldsymbol{e}$ is the Euler's number.

We also show that, the class of functions Φ_γ also serves as an essential tool to give a constant bound on the approximate price of stability. In fact, by exploiting Φ_γ, we are able to show (Theorem 8) an upper bound of $2/(1+\delta)$ for the $(d+\delta)$-approximate price of stability, for every $\delta \in [0, 1]$. To prove this bound, we use the standard potential function argument. Specifically, we first give bounds (Lemma 6 and Lemma 7) relating the value of the $(d+\delta)$-approximate potential function for a given state to the social cost of that state; if we then perform a sequence of $(d+\delta)$-improvement moves starting from an optimal state, the potential does not increase, and hence we can bound the cost of any $(d+\delta)$-approximate pure Nash equilibrium that we reach. Notice that our bound does not depend on the range of the players' weights and significantly improves the bound provided in [7], by making use of a different and simpler potential function.

Roadmap. We begin with the definition of a class of state-value functions in Section 2. In Section 3 we first present a simple technique to bound the approximation guarantee of a given state-value function, subsequently we show that some elements of the class introduced in Section 2 provide a good approximation. In Section 4 we present the bound on the approximate price of stability. Finally, in section 5 we present a couple of technical lemmas.

2 A class of state-value functions

In this section we define a class of functions mapping every state of the game to a non-negative real number. This class of functions will be exploited in the subsequent sections.

▶ **Definition 1.** *Let $G = \langle N, E, (w_i)_{i\in N}, (S_i)_{i\in N}, (a_e, k_e)_{e\in E}\rangle$ be an instance of* WCG(d)*. For every resource $e \in E$ and every $\gamma = (\gamma_e)_{e\in E}$, we define*

$$\Phi_\gamma(\mathbf{s}) = \sum_{e\in E} a_e \Psi_e^{\gamma_e}\big(L_e(\mathbf{s})\big),$$

where, for every nonempty subset of players $P \subseteq N$, we have

$$\Psi_e^{\gamma_e}(P) = \frac{\gamma_e}{k_e+1}\left(\sum_{j\in P} w_j\right)^{k_e+1} + \left(1 - \frac{\gamma_e}{k_e+1}\right)\sum_{j\in P} w_j^{k_e+1}.$$

3 Approximate potential functions

The main result of this section is stated by Theorem 3, where we show the existence of good approximate potential functions. Before showing this result, in Theorem 2 we illustrate our tool to define an approximate potential function. Such tool gives a local condition to each resource to determine the approximation guarantee of a given state-value function. We conclude this section with two corollaries, the first (Corollary 4) showing that the social optimum of an instance of WCG(d) is always a $(d+1)$-approximate pure Nash equilibrium, the second (Corollary 5) showing that, under mild conditions, the game always admits a constant approximate potential function.

▶ **Theorem 2.** *Let $G = \langle N, E, (w_i)_{i\in N}, (S_i)_{i\in N}, (a_e, k_e)_{e\in E}\rangle$ be an instance of* WCG(d)*. Let $\Gamma : \mathcal{S}(G) \mapsto \mathbb{R}^{>0}$ be a state-value function such that $\Gamma(\mathbf{s}) = \sum_{e\in E} a_e \Gamma_e\big(L_e(\mathbf{s})\big)$, where $\Gamma_e : 2^N \mapsto \mathbb{R}^{>0}$. If, for every resource $e \in E$, every non-empty subset of players $P \subseteq N$ and every player $i \in P$, there exist $\lambda_e, \upsilon_e \in \mathbb{R}^{>0}$, with $\lambda_e \le \upsilon_e$, such that*

$$\frac{w_i \ell_e(P)}{a_e\Big(\Gamma_e(P) - \Gamma_e(P\setminus\{i\})\Big)} \in [\lambda_e, \upsilon_e] \tag{1}$$

then Γ is a $\left(\frac{\upsilon}{\lambda}\right)$-approximate potential function for G, where $\upsilon = \max_{e\in E}\upsilon_e$ and $\lambda = \min_{e\in E}\lambda_e$.

Proof. Let us consider a state $\mathbf{s} \in \mathcal{S}(G)$ and a player i. Let us assume that i can perform an $\frac{\upsilon}{\lambda}$-improvement move by replacing strategy $\mathbf{s}(i)$ with $s \neq \mathbf{s}(i)$, i.e., $\frac{\upsilon}{\lambda} c_i([\mathbf{s}_{-i}, s]) < c_i(\mathbf{s})$. In order to prove the claim we need to show that $\Gamma([\mathbf{s}_{-i}, s]) < \Gamma(\mathbf{s})$. To this aim, let us bound the expression $\Gamma([\mathbf{s}_{-i}, s]) - \Gamma(\mathbf{s})$. We have,

$$
\begin{aligned}
\Gamma([\mathbf{s}_{-i}, s]) - \Gamma(\mathbf{s}) &= \sum_{e \in E} a_e \Gamma_e\big(L_e([\mathbf{s}_{-i}, s])\big) - \sum_{e \in E} a_e \Gamma_e\big(L_e(\mathbf{s})\big) \\
&= \sum_{e \in E} a_e \Big[\Gamma_e\big(L_e([\mathbf{s}_{-i}, s])\big) - \Gamma_e\big(L_e(\mathbf{s})\big)\Big] \\
&= \sum_{e \in s \setminus \mathbf{s}(i)} a_e \Big[\Gamma_e\big(L_e([\mathbf{s}_{-i}, s])\big) - \Gamma_e\big(L_e(\mathbf{s})\big)\Big] \\
&\qquad - \sum_{e \in \mathbf{s}(i) \setminus s} a_e \Big[\Gamma_e\big(L_e(\mathbf{s})\big) - \Gamma_e\big(L_e([\mathbf{s}_{-i}, s])\big)\Big] \\
&= \sum_{e \in s} a_e \Big[\Gamma_e\big(L_e([\mathbf{s}_{-i}, s])\big) - \Gamma_e\big(L_e(\mathbf{s})\big)\Big] \\
&\qquad - \sum_{e \in \mathbf{s}(i)} a_e \Big[\Gamma_e\big(L_e(\mathbf{s})\big) - \Gamma_e\big(L_e([\mathbf{s}_{-i}, s])\big)\Big] && (2) \\
&\leq \sum_{e \in s} \frac{1}{\lambda_e} w_i \ell_e\big(L_e([\mathbf{s}_{-i}, s])\big) - \sum_{e \in \mathbf{s}(i)} \frac{1}{\upsilon_e} w_i \ell_e\big(L_e(\mathbf{s})\big) && (3) \\
&\leq \frac{1}{\lambda} \sum_{e \in s} w_i \ell_e\big(L_e([\mathbf{s}_{-i}, s])\big) - \frac{1}{\upsilon} \sum_{e \in \mathbf{s}(i)} w_i \ell_e\big(L_e(\mathbf{s})\big) && (4) \\
&= \frac{w_i}{\upsilon} \Big(\frac{\upsilon}{\lambda} c_i([\mathbf{s}_{-i}, s]) - c_i(\mathbf{s})\Big) && (5)
\end{aligned}
$$

where (2) follows from the fact that for every $e \in \mathbf{s}(i) \cap s$ the variation of Γ_e is null since $L_e([\mathbf{s}_{-i}, s]) = L_e(\mathbf{s})$, (3) from (1), (4) from the definition of λ and υ and (5) from the definition of cost.

From (5) we obtain that, if $\frac{\upsilon}{\lambda} c_i([\mathbf{s}_{-i}, s]) < c_i(\mathbf{s})$ then $\Gamma([\mathbf{s}_{-i}, s]) < \Gamma(\mathbf{s})$, from which the claim follows. ◀

We are ready to present the main result of this section.

▶ **Theorem 3.** *Let $G = \langle N, E, (w_i)_{i \in N}, (S_i)_{i \in N}, (a_e, k_e)_{e \in E} \rangle$ be an instance of* WCG(d)*. It holds that*

(a) *if $\gamma_e = 1$, for every $e \in E$, then Φ_γ is a $\rho(G)$-approximate potential function for G, where*

$$
\rho(G) = \max_{e \in E} \sup_{x > 0} \frac{\big(1 + x\big)^{k_e}}{\frac{1}{k_e+1}\big(1 + x\big)^{k_e+1} + \frac{k_e}{k_e+1} - \frac{1}{k_e+1} x^{k_e+1}} \leq d; \tag{6}
$$

(b) *if $\gamma_e = k_e$, for every $e \in E$, then Φ_γ is a d-approximate potential function for G;*

(c) *if $\gamma_e = k_e + \delta$, for every $e \in E$ and $\delta \geq 0$, then Φ_γ is a $(d + \delta)$-approximate potential function for G.*

Proof. We prove the claim using Theorem 2. Therefore, for every resource $e \in E$, every non-empty subset of players $P \subseteq N$ and every player $i \in P$, we bound the ratio

$$
\frac{w_i \ell_e(P)}{a_e\Big(\Psi_e^{\gamma_e}(P) - \Psi_e^{\gamma_e}(P \setminus \{i\})\Big)}. \tag{7}
$$

Let us explicitly rewrite the numerator and denominator of the previous expression. We distinguish between the cases $|P| = \{i\}$ and $|P| \geq 2$.

Let us fist assume $|P| = \{i\}$. In this case, for $w_i \ell_e(P)$ we get

$$w_i \ell_e(P) = w_i a_e (w_i)^{k_e} = a_e w_i^{k_e+1}. \tag{8}$$

On the other hand, for the expression $\Psi_e^{\gamma_e}(P) - \Psi_e^{\gamma_e}(P \setminus \{i\})$, using Definition 1, we have

$$\Psi_e^{\gamma_e}(P) - \Psi_e^{\gamma_e}(P \setminus \{i\}) = \left[\frac{\gamma_e}{k_e+1} (w_j)^{k_e+1} + \left(1 - \frac{\gamma_e}{k_e+1}\right) w_j^{k_e+1} \right] - 0 = w_j^{k_e+1}. \tag{9}$$

Therefore, by combining (8) and (9), we conclude that, for $|P| = \{i\}$, for the ratio (7) we have

$$\frac{w_i \ell_e(P)}{a_e \Big(\Psi_e^{\gamma_e}(P) - \Psi_e^{\gamma_e}(P \setminus \{i\}) \Big)} = \frac{a_e w_i^{k_e+1}}{a_e w_i^{k_e+1}} = 1. \tag{10}$$

Not let us assume that $|P| \geq 2$. In this case, for $w_i \ell_e(P)$ we get

$$\begin{aligned} w_i \ell_e(P) &= w_i a_e \left(\sum_{j \in P} w_j \right)^{k_e} = w_i a_e \left(w_i + \sum_{j \in P \setminus \{i\}} w_j \right)^{k_e} \\ &= a_e w_i^{k_e+1} \big(1 + \lambda_i(P)\big)^{k_e}. \end{aligned} \tag{11}$$

Now, let us focus on the expression $\Psi_e^{\gamma_e}(P) - \Psi_e^{\gamma_e}(P \setminus \{i\})$. For every $i \in P$, let $\lambda_i(P) = \frac{1}{w_i} \sum_{j \in P \setminus \{i\}} w_j$. Using Definition 1, we have

$$\begin{aligned} &\Psi_e^{\gamma_e}(P) - \Psi_e^{\gamma_e}(P \setminus \{i\}) \\ &= \frac{\gamma_e}{k_e+1} \left(\sum_{j \in P} w_j \right)^{k_e+1} + \left(1 - \frac{\gamma_e}{k_e+1}\right) \sum_{j \in P} w_j^{k_e+1} \\ &\quad - \frac{\gamma_e}{k_e+1} \left(\sum_{j \in P \setminus \{i\}} w_j \right)^{k_e+1} - \left(1 - \frac{\gamma_e}{k_e+1}\right) \sum_{j \in P \setminus \{i\}} w_j^{k_e+1} \\ &= \frac{\gamma_e}{k_e+1} \left(\sum_{j \in P} w_j \right)^{k_e+1} + \left(1 - \frac{\gamma_e}{k_e+1}\right) w_i^{k_e+1} - \frac{\gamma_e}{k_e+1} \left(\sum_{j \in P \setminus \{i\}} w_j \right)^{k_e+1} \\ &= \frac{\gamma_e}{k_e+1} \left(w_i + \sum_{j \in P \setminus \{i\}} w_j \right)^{k_e+1} + \left(1 - \frac{\gamma_e}{k_e+1}\right) w_i^{k_e+1} - \frac{\gamma_e}{k_e+1} \left(\sum_{j \in P \setminus \{i\}} w_j \right)^{k_e+1} \\ &= \frac{\gamma_e}{k_e+1} w_i^{k_e+1} \big(1 + \lambda_i(P)\big)^{k_e+1} + \left(1 - \frac{\gamma_e}{k_e+1}\right) w_i^{k_e+1} - \frac{\gamma_e}{k_e+1} w_i^{k_e+1} \lambda_i(P)^{k_e+1} \\ &= w_i^{k_e+1} \left[\frac{\gamma_e}{k_e+1} \big(1 + \lambda_i(P)\big)^{k_e+1} + \left(1 - \frac{\gamma_e}{k_e+1}\right) - \frac{\gamma_e}{k_e+1} \lambda_i(P)^{k_e+1} \right] \end{aligned} \tag{12}$$

Therefore, by combining (11) and (12), we conclude that, for $|P| \geq 2$, for the expression (7) we have

$$\begin{aligned}
&\frac{w_i \ell_e(P)}{a_e\Big(\Psi_e^{\gamma_e}(P) - \Psi_e^{\gamma_e}(P \setminus \{i\})\Big)} \\
&= \frac{a_e w_i^{k_e+1}\big(1+\lambda_i(P)\big)^{k_e}}{a_e w_i^{k_e+1}\left[\frac{\gamma_e}{k_e+1}\big(1+\lambda_i(P)\big)^{k_e+1} + \Big(1-\frac{\gamma_e}{k_e+1}\Big) - \frac{\gamma_e}{k_e+1}\lambda_i(P)^{k_e+1}\right]} \\
&= \frac{\big(1+\lambda_i(P)\big)^{k_e}}{\frac{\gamma_e}{k_e+1}\big(1+\lambda_i(P)\big)^{k_e+1} + \Big(1-\frac{\gamma_e}{k_e+1}\Big) - \frac{\gamma_e}{k_e+1}\lambda_i(P)^{k_e+1}}. \qquad (13)
\end{aligned}$$

Now we apply Lemma 9 to (13), by setting $x = \lambda_i(P)$, $h = k_e$ and $\beta = \gamma_e$.
If $\gamma_e = 1$, we obtain

$$\frac{w_i \ell_e(P)}{a_e\Big(\Psi_e^{\gamma_e}(P) - \Psi_e^{\gamma_e}(P \setminus \{i\})\Big)} \in \big[1, k_e\big]. \qquad (14)$$

If $\gamma_e = k_e$, we have

$$\frac{w_i \ell_e(P)}{a_e\Big(\Psi_e^{\gamma_e}(P) - \Psi_e^{\gamma_e}(P \setminus \{i\})\Big)} \in \Big[\frac{1}{k_e}, 1\Big]. \qquad (15)$$

Finally, if $\gamma_e = k_e + \delta$, with $\delta \geq 0$, we have

$$\frac{w_i \ell_e(P)}{a_e\Big(\Psi_e^{\gamma_e}(P) - \Psi_e^{\gamma_e}(P \setminus \{i\})\Big)} \in \Big[\frac{1}{k_e+\delta}, 1\Big]. \qquad (16)$$

Claim (a) follows by combining (10), (13) and (14), and by applying Theorem 2. Claims (b) and (c) follow by combining (10), (15) and (16), and by applying Theorem 2. ◀

▶ **Corollary 4.** *Let $G = \langle N, E, (w_i)_{i\in N}, (S_i)_{i\in N}, (a_e, k_e)_{e\in E}\rangle$ be an instance of* WCG(d)*. Any social optimum of G is a $(d+1)$-approximate pure Nash equilibrium.*

Proof. Let us consider the function $\Phi_\gamma(\mathbf{s}) = \sum_{e\in E} \Psi_e^{\gamma_e}(L_e(\mathbf{s}))$ (defined in Definition 1), with $\gamma_e = k_e + 1$. The claim follows from observing that $\Phi_\gamma(\mathbf{s}) = C(\mathbf{s})$ and from the fact that, by Theorem 3(c), Φ_γ is also a $(d+1)$-approximate potential function. ◀

▶ **Corollary 5.** *Let $G = \langle N, E, (w_i)_{i\in N}, (S_i)_{i\in N}, (a_e, k_e)_{e\in E}\rangle$ be a mildly congested instance of* WCG(d)*. If $\gamma_e = 1$, for every $e \in E$, then Φ_γ is a $\frac{\boldsymbol{e}}{\boldsymbol{e}-1}$-approximate potential function for G, where $\boldsymbol{e}$ is the Euler's number.*

Proof Sketch. Since G is mildly congested, we have $\ell_e(L_e(\mathbf{s})) \geq (k_e+1)w_{\max}$, for every resource $e \in E$ and state $\mathbf{s} \in \mathcal{S}(G)$, where $w_{\max}$ denotes the greatest weight in G. Under this condition, we can restrict the proof of Theorem 3 to the case in which the generic subset of player P is such that $\sum_{j\in P} w_j \geq (k_e+1)w_{\max}$. With this condition in place, we have that, for every $i \in P$, $\lambda_i(P)$ is at least k_e. In fact,

$$\lambda_i(P) = \frac{1}{w_i}\left(\sum_{j\in P} w_j - w_i\right) \geq \frac{1}{w_i}\Big((k_e+1)w_{\max} - w_i\Big) \geq \frac{w_{\max}}{w_i}k_e \geq k_e.$$

By using the previous inequality, it is easy to prove that the expression (13) gets always values in the range $[1, \frac{\boldsymbol{e}}{\boldsymbol{e}-1}]$, for every $k_e \geq 1$. The claim follows by applying Theorem 2. ◀

4 Approximate price of stability

In this section we show an upper bound on the α-approximate price of stability, for $\alpha \in [d, d+1]$. This bound is stated by Theorem 8, whose proof is based on Lemma 6 and Lemma 7 presented below.

▶ **Lemma 6.** *Let $G = \langle N, E, (w_i)_{i\in N}, (S_i)_{i\in N}, (a_e, k_e)_{e\in E}\rangle$ be an instance of* WCG(d). *Let $\gamma_e = k_e + \delta$ and $\delta \in [0,1]$, for every resource $e \in E$. For every resource $e \in E$ and every non-empty subset of players $P \subseteq N$, we have*

$$\Psi_e^{\gamma_e}(P) \leq \left(\sum_{j\in P} w_j\right)^{k_e+1} \leq \frac{k_e+1}{k_e+\delta}\Psi_e^{\gamma_e}(P).$$

Proof. We have

$$\frac{\left(\sum_{j\in P} w_j\right)^{k_e+1}}{\Psi_e^{\gamma_e}(P)} = \frac{\left(\sum_{j\in P} w_j\right)^{k_e+1}}{\frac{k_e+\delta}{k_e+1}\left(\sum_{j\in P} w_j\right)^{k_e+1} + \left(1-\frac{k_e+\delta}{k_e+1}\right)\sum_{j\in P} w_j^{k_e+1}} \tag{17}$$

$$\in \left[1, \frac{k_e+1}{k_e+\delta}\right). \tag{18}$$

where (17) follows from Definition 1 and the definition of γ_e, and (18) follows by applying Lemma 10, where we set $x = \left(\sum_{j\in P} w_j\right)^{k_e+1}$, $y = \sum_{j\in P} w_j^{k_e+1}$ and $\beta = \frac{k_e+1}{k_e+\delta}$. ◀

In the following lemma, we give bounds relating the value of the approximate potential function for a given state to the social cost of that state.

▶ **Lemma 7.** *Let $G = \langle N, E, (w_i)_{i\in N}, (S_i)_{i\in N}, (a_e, k_e)_{e\in E}\rangle$ be an instance of* WCG(d). *Let $\gamma = (\gamma_e)_{e\in E}$, where $\gamma_e = k_e + \delta$ and $\delta \in [0,1]$. For every state $\mathbf{s} \in \mathcal{S}(G)$ we have*

$$\Phi_\gamma(\mathbf{s}) \leq C(\mathbf{s}) \leq \frac{2}{1+\delta}\Phi_\gamma(\mathbf{s}). \tag{19}$$

Proof. Let $E = \{e_1, e_2, \ldots, e_m\}$. Let us bound the ratio

$$\frac{C(\mathbf{s})}{\Phi_\gamma(\mathbf{s})} = \frac{\sum_{t=1}^m a_{e_t}\left(\sum_{j\in L_{e_t}(\mathbf{s})} w_j\right)^{k_{e_t}+1}}{\sum_{t=1}^m a_{e_t}\Psi_{e_t}^{\gamma_{e_t}}\left(L_{e_t}(\mathbf{s})\right)}.$$

In order to bound $C(\mathbf{s})/\Phi_\gamma(\mathbf{s})$, we consider the ratio between the t-th term in the numerator and the t-th term in the denominator, for every $t \in [1,m]$, that is

$$\frac{\left(\sum_{j\in L_{e_t}(\mathbf{s})} w_j\right)^{k_{e_t}+1}}{\Psi_{e_t}^{\gamma_{e_t}}\left(L_{e_t}(\mathbf{s})\right)}. \tag{20}$$

From Lemma 6, (20) gets values in the interval $\left[1, \frac{k_{e_t}+1}{k_{e_t}+\delta}\right]$. We get that the smallest ratio is 1 while the greatest one is $\max_t \frac{k_{e_t}+1}{k_{e_t}+\delta} \leq \frac{2}{1+\delta}$. It follows that, $C(\mathbf{s})/\Phi_\gamma(\mathbf{s})$ is at least 1 and at most $\frac{2}{1+\delta}$, from which the claim follows. ◀

▶ **Theorem 8.** *Let $G = \langle N, E, (w_i)_{i\in N}, (S_i)_{i\in N}, (a_e, k_e)_{e\in E}\rangle$ be an instance of* WCG(d). *Then* $\mathrm{PoS}_{d+\delta}(G) \leq \frac{2}{1+\delta}$, *for every $\delta \in [0,1]$.*

Proof. Let Φ_γ be the function with $\gamma = (\gamma_e)_{e\in E}$, where $\gamma_e = k_e + \delta$. Let $\mathbf{o} \in \mathcal{OPT}(G)$ be a social optimum. Let us consider any sequence of $(d+\delta)$-improvement moves starting from $\mathbf{o}$. From Theorem 3(c) we know that this sequence converges to a state which is a $(d+\delta)$-approximate pure Nash equilibrium which we denote by $\mathbf{e}$. Moreover, along this sequence of moves, Φ_γ is not increasing. Hence,

$$\Phi_\gamma(\mathbf{e}) \leq \Phi_\gamma(\mathbf{o}). \tag{21}$$

By applying Lemma 7 repeatedly to both $\mathbf{o}$ and $\mathbf{e}$ we obtain

$$C(\mathbf{e}) \leq \frac{2}{1+\delta}\Phi_\gamma(\mathbf{e}) \leq \frac{2}{1+\delta}\Phi_\gamma(\mathbf{o}) \leq \frac{2}{1+\delta}C(\mathbf{o}),$$

where the second inequality follows from (21). From which the claim follows. ◀

5 Technical lemmas

In this section we present two technical lemmas. Lemma 9 is used in the proof of Theorem 3, while Lemma 10 is used in the proof of Lemma 6.

▶ **Lemma 9.** *For every $x \in \mathbb{R}^{>0}$, $h \in \mathbb{Z}^{\geq 1}$ and $\beta \in \mathbb{R}^{\geq 1}$, we have*

$$\frac{(1+x)^h}{\beta\frac{1}{h+1}(1+x)^{h+1} + (1-\beta\frac{1}{h+1}) - \beta\frac{1}{h+1}x^{h+1}} \in \begin{cases} [\frac{1}{\beta}, \frac{h}{\beta}] & \text{if } \beta \in [1,h] \\ [\frac{1}{\beta}, 1] & \text{if } \beta \geq h. \end{cases}$$

Proof. We have

$$\begin{aligned}
&\frac{(1+x)^h}{\beta\frac{1}{h+1}(1+x)^{h+1} + (1-\beta\frac{1}{h+1}) - \beta\frac{1}{h+1}x^{h+1}} \\
&= \frac{\sum_{t=0}^{h}\binom{h}{t}x^t}{\beta\frac{1}{h+1}\sum_{t=0}^{h+1}\binom{h+1}{t}x^t + (1-\beta\frac{1}{h+1}) - \beta\frac{1}{h+1}x^{h+1}} \\
&= \frac{1+\sum_{t=1}^{h}\binom{h}{t}x^t}{1+\beta\frac{1}{h+1}\sum_{t=1}^{h+1}\binom{h+1}{t}x^t - \beta\frac{1}{h+1}x^{h+1}} \\
&= \frac{1+\sum_{t=1}^{h}\binom{h}{t}x^t}{1+\sum_{t=1}^{h}\beta\frac{1}{h+1}\binom{h+1}{t}x^t} \\
&= \frac{1+\sum_{t=1}^{h}\binom{h}{t}x^t}{1+\sum_{t=1}^{h}\beta\frac{1}{h+1}\frac{h+1}{h+1-t}\binom{h}{t}x^t} = \frac{1+\sum_{t=1}^{h}\binom{h}{t}x^t}{1+\sum_{t=1}^{h}\beta\frac{1}{h+1-t}\binom{h}{t}x^t} && (22) \\
&= \frac{1\cdot x^0 + \binom{h}{1}x^1 + \binom{h}{2}x^2 + \ldots + \binom{h}{h}x^h}{1\cdot x^0 + \beta\frac{1}{h}\binom{h}{1}x^1 + \beta\frac{1}{h-1}\binom{h}{2}x^2 + \ldots + \beta\binom{h}{h}x^h}, && (23)
\end{aligned}$$

where (22) holds because

$$\binom{h+1}{t} = \frac{\frac{(h+1)!}{t!(h+1-t)!}}{\frac{h!}{t!(h-t)!}}\binom{h}{t} = \frac{(h+1)!}{t!(h+1-t)!}\frac{t!(h-t)!}{h!}\binom{h}{t} = \frac{h+1}{h+1-t}\binom{h}{t}.$$

In order to bound (23), for every $t \in [0,h]$ we consider the ratio between the coefficient of the term x^t in the numerator and the coefficient of the same term in the denominator. For $t=0$ the ratio is 1, while for $t \in [1,h]$ the ratio is $\frac{h+1-t}{\beta}$. For the case $\beta \in [1,h]$, we get that the smallest ratio is $1/\beta$ while the greatest is h/β. It follows that, when $\beta \in [1,h]$, the expression in (23) is at least $1/\beta$ and at most h/β. For the case $\beta \geq h$, we obtain that the smallest is $1/\beta$ while the greatest ratio is 1df. Therefore, when $\beta \geq h$, the expression in (23) is at least $1/\beta$ and at most 1. From which the claim follows. ◀

▶ **Lemma 10.** *For every $x, y, \beta \in \mathbb{R}^{>0}$ such that $\beta \geq 1$ and $y \leq x$, we have*

$$\frac{x}{\frac{1}{\beta}x + (1-\frac{1}{\beta})y} \in \big[1, \beta\big].$$

Proof. We have

$$\frac{x}{\frac{1}{\beta}x + (1-\frac{1}{\beta})y} = \frac{1}{\frac{1}{\beta} + (1-\frac{1}{\beta})\frac{y}{x}} \geq \frac{1}{\frac{1}{\beta} + (1-\frac{1}{\beta})} = 1,$$

where the last inequality is due to the fact that $y/x \leq 1$.

$$\frac{x}{\frac{1}{\beta}x + (1-\frac{1}{\beta})y} = \frac{1}{\frac{1}{\beta} + (1-\frac{1}{\beta})\frac{y}{x}} < \frac{1}{\frac{1}{\beta}} = \beta,$$

where the last inequality is due to the fact that $y/x > 0$. ◀

References

1. Sebastian Aland, Dominic Dumrauf, Martin Gairing, Burkhard Monien, and Florian Schoppmann. Exact price of anarchy for polynomial congestion games. *SIAM Journal on Computing*, 40(5):1211–1233, 2011.
2. Ioannis Caragiannis, Angelo Fanelli, Nick Gravin, and Alexander Skopalik. Approximate pure Nash equilibria in weighted congestion games: Existence, efficient computation, and structure. *ACM Transactions on Economics and Computation*, 3(1):2:1–2:32, 2015.
3. Ioannis Caragiannis, Michele Flammini, Christos Kaklamanis, Panagiotis Kanellopoulos, and Luca Moscardelli. Tight bounds for selfish and greedy load balancing. *Algorithmica*, 61(3):606–637, 2011.
4. Steve Chien and Alistair Sinclair. Convergence to approximate Nash equilibria in congestion games. *Games and Economic Behavior*, 71(2):315–327, 2011.
5. Nicolas Christin, Jens Grossklags, and John Chuang. Near Rationality and Competitive Equilibria in Networked Systems. In *Proceedings of the ACM SIGCOMM Workshop on Practice and Theory of Incentives in Networked Systems (PINS)*, pages 213–219, 2004.
6. George Christodoulou and Martin Gairing. Price of stability in polynomial congestion games. *ACM Transactions on Economics and Computation*, 4(2):10:1–10:17, 2016.
7. George Christodoulou, Martin Gairing, Yiannis Giannakopoulos, and Paul G. Spirakis. The Price of Stability of Weighted Congestion Games. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 150:1–150:16, 2018.
8. George Christodoulou and Elias Koutsoupias. On the price of anarchy and stability of correlated equilibria of linear congestion games. In *Proceedings of the 13th Annual European Symposium on Algorithms (ESA)*, pages 59–70, 2005.
9. Juliane Dunkel and Andreas S. Schulz. On the complexity of pure-strategy Nash equilibria in congestion and local-effect games. *Mathematics of Operations Research*, 33(4):851–868, 2008.
10. Dimitris Fotakis, Spyros C. Kontogiannis, and Paul G. Spirakis. Selfish unsplittable flows. *Theoretical Computer Science*, 348(2-3):226–239, 2005.
11. Michel X. Goemans, Vahab S. Mirrokni, and Adrian Vetta. Sink equilibria and convergence. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 142–154, 2005.
12. Christoph Hansknecht, Max Klimm, and Alexander Skopalik. Approximate pure Nash equilibria in weighted congestion games. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 242–257, 2014.
13. Tobias Harks and Max Klimm. On the existence of pure Nash equilibria in weighted congestion games. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP), Part I*, pages 79–89, 2010.

14 Lavy Libman and Ariel Orda. Atomic resource sharing in noncooperative networks. *Telecommunication Systems*, 17(4):385–409, 2001.

15 Panagiota N. Panagopoulou and Paul G. Spirakis. Algorithms for pure Nash equilibria in weighted congestion games. *ACM Journal of Experimental Algorithmics*, 11, 2006.

16 Robert W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

Temporal Cliques Admit Sparse Spanners

Arnaud Casteigts
LaBRI, Université de Bordeaux, CNRS, Bordeaux INP, France
arnaud.casteigts@labri.fr

Joseph G. Peters
School of Computing Science, Simon Fraser University, Canada
peters@cs.sfu.ca

Jason Schoeters
LaBRI, Université de Bordeaux, CNRS, Bordeaux INP, France
jason.schoeters@labri.fr

Abstract

Let $G = (V, E)$ be an undirected graph on n vertices and $\lambda : E \to 2^{\mathbb{N}}$ a mapping that assigns to every edge a non-empty set of positive integer labels. These labels can be seen as discrete times when the edge is present. Such a labeled graph $\mathcal{G} = (G, \lambda)$ is said to be *temporally connected* if a path exists with non-decreasing times from every vertex to every other vertex. In a seminal paper, Kempe, Kleinberg, and Kumar (STOC 2000) asked whether, given such a temporal graph, a *sparse* subset of edges can always be found whose labels suffice to preserve temporal connectivity – a *temporal spanner*. Axiotis and Fotakis (ICALP 2016) answered negatively by exhibiting a family of $\Theta(n^2)$-dense temporal graphs which admit no temporal spanner of density $o(n^2)$. The natural question is then whether sparse temporal spanners always exist in *some* classes of dense graphs.

In this paper, we answer this question affirmatively, by showing that if the underlying graph G is a complete graph, then one can always find temporal spanners of density $O(n \log n)$. The best known result for complete graphs so far was that spanners of density $\binom{n}{2} - \lfloor n/4 \rfloor = O(n^2)$ always exist. Our result is the first *positive* answer as to the existence of $o(n^2)$ sparse spanners in adversarial instances of temporal graphs since the original question by Kempe et al., focusing here on complete graphs. The proofs are constructive and directly adaptable as an algorithm.

2012 ACM Subject Classification Theory of computation → Sparsification and spanners; Theory of computation → Dynamic graph algorithms

Keywords and phrases Dynamic networks, Temporal graphs, Temporal connectivity, Sparse spanners

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.134

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Related Version A full version of this paper is available at `https://arxiv.org/abs/1810.00104` [7].

Funding This research was supported by ANR project ESTATE (ANR-16-CE25-0009-03) and NSERC of Canada.

Acknowledgements We thank Cyril Gavoille for advice on the presentation of these results.

1 Introduction

The study of highly dynamic networks has gained interest lately, motivated by emerging technological contexts (e.g. vehicular networks, wireless sensors, robots, and drones) where the entities move and communicate with each other. The communication links in these networks vary with time, leading to the definition of temporal graph models (also called time-varying graphs or evolving graphs) where temporality plays a central role. In these graphs, the properties of interest are often defined over the time rather than at a given instant.

For example, the graph may never be connected, and yet offer a form of connectivity over time. In [6], a dozen temporal properties were identified that have been effectively exploited in the distributed computing and networking literature. Perhaps the most basic property is that of *temporal connectivity*, which requires that every vertex can reach every other vertex through a temporal path (also called *journey* [5]), that is, a path whose edges are used over a non-decreasing sequence of times. The times may also be required to be strictly increasing (strict journey), both cases being carefully discussed in this paper. Temporal connectivity was considered in an early paper by Awerbuch and Even [3] (1984), and systematically studied from a graph-theoretical point of view in the early 2000's in a number of seminal works including Kempe, Kleinberg, and Kumar [13], and Bui-Xuan, Ferreira, and Jarry [5] (see also [16] for an early study of graphs with time-dependent delays on the edges). More recently, temporal connectivity has been the subject of several algorithmic studies, such as [1] and [4] (discussed below), and [18] and [19], which consider algorithms for computing structures related to temporal connectivity. Broad surveys on these topics can be found, e.g. in [6, 11, 14], although the list is non-exhaustive and the literature is rapidly evolving.

1.1 Sparse Temporal Spanners and Related Work

In the last section of [13] (conference version [12]), Kempe, Kleinberg, and Kumar ask

> "Given a temporally connected network $G = (V, E)$ on n nodes, is there a set $E' \subseteq E$ consisting of $O(n)$ edges so that the temporal network on the subgraph (V, E') is also temporally connected? In other words, do all temporal networks have sparse subgraphs preserving this basic connectivity property?"

Here, Kempe et al. consider a model where each edge has a single label, thus the edges are identified with their labels, but the discussion is more general. What they are asking, essentially, is whether an analogue of spanning tree exists for temporal networks when the labels are *already fixed*. They answer immediately (and negatively) for the particular case of $O(n)$ density, by showing that hypercubes labeled in a certain way need all of their edges to achieve temporal connectivity, thus some temporal graphs of density $\Theta(n \log n)$ cannot be sparsified. The more general question, asking whether $o(n^2)$-sparse spanners always exist in dense temporal graphs remained open for more than a decade, and was eventually settled, again negatively, by Axiotis and Fotakis [4]. The proof in [4] exhibits an infinite family of temporally connected graphs with $\Theta(n^2)$ edges that do not admit $o(n^2)$-sparse spanners. Their construction can be adapted for strict and non-strict journeys.

On the positive side, Akrida et al. [1] show that, if the underlying graph G is complete and every edge is assigned a single globally-unique label, then it is always possible to find a temporal spanner of density $\binom{n}{2} - \lfloor n/4 \rfloor$ edges (leaving the asymptotic density unchanged). Akrida et al. [1] also prove that if the label of every edge in G is chosen uniformly at random (from an appropriate interval), then almost surely the graph admits a temporal spanner with $O(n \log n)$ edges. Both [4] and [1] include further results related to the (in-)approximability of finding a *minimum* temporal spanner, which is out of the scope of this paper.

By its nature, the problem of finding a temporal spanner in a temporal network seems to be significantly different from its classical (i.e., non-temporal) version, whether this version considers a static graph (see e.g. [8, 15, 17]) or the current network topology of an updated dynamic graph (see e.g. [2, 9, 10]). The essential difference is that spanning trees always exist in standard (connected) graphs, thus one typically focuses on the tradeoff between the density of a solution and a quality parameter like the *stretch factor*, rather than to the very existence of a sparse spanner.

1.2 Contributions

In this paper, we establish that temporal graphs built on top of complete graphs *unconditionally* admit $O(n \log n)$-sparse temporal spanners when *non-strict* journeys are allowed. Furthermore, such spanners can be computed in polynomial time. The case of strict journeys requires more discussion. Kempe et al. observed in [13] that if every edge of a complete graph is given the same label, then this graph is temporally connected, but no multi-hop *strict* journey can exist, thus none of the edges can be removed, and the problem is trivially unsolvable. To make the problem interesting when only strict journeys are allowed, one should constrain the labeling to avoid this pathological situation. Thus, in this case, we require that a subset of one label per edge exists in which any two adjacent edges have different labels. This formulation slightly generalizes the single-label global-unicity assumption made in [1] (although essentially equivalent) and eliminates the distinction between strict and non-strict journeys. Under this restriction, we establish that all temporal graphs whose underlying graph is complete admit an $O(n \log n)$-sparse temporal spanner. Moreover, if the restricted labeling is given, then the spanner can be computed in polynomial time. (The problem of deciding whether a general labeling admits such a sub-labeling is not discussed here; it might be an interesting problem on its own, possibly computationally hard.)

Our proofs are constructive. We start by observing that the above two settings one-way reduce to the setting where every edge has a single label and two adjacent edges have different labels. The reduction is "one-way" in the sense that the transformed instance may have less feasible journeys than the original instance, but all of these journeys correspond to valid journeys in the original instance, so that a temporal spanner computed in the transformed instance is valid in the original instance. As a result, the main algorithm takes as input a complete graph $\mathcal{G}$ with single, locally distinct labels, and computes an $O(n \log n)$-sparse spanner of $\mathcal{G}$ in polynomial time. This algorithm is based on several original techniques, which we think may be of interest for other problems related to temporal connectivity.

In summary, our results give the first positive answer to the question of whether sparse temporal spanners always exist in a class of dense graphs, focusing here on the case of complete graphs. This answer complements the negative answer by Axiotis and Fotakis [4] and motivates more investigation to understand where the transition occurs between their negative result (no sparse spanners exist in some dense temporal graphs) and our positive result (they essentially always exist in complete graphs).

The paper is organized as follows. In Section 2, we define the model and notations, and describe the one-way reductions that allows us to concentrate subsequently on single (and distinct) labels. We also mention a technique from [1] and we introduce a basic technique called *pivoting* which is a natural analogue of Kosaraju's algorithm for temporal graphs. In Section 3 we introduce the main concepts used in the rest of the paper, namely *delegation*, *dismountability*, and *k-hop dismountability*, whose purpose is to recursively self-reduce the problem to smaller graph instances. While these technique alone fail in some cases, we extend them and combine them into a more sophisticated algorithm that successfully computes a temporal spanner of $O(n \log n)$ edges. The first step, presented in Section 4, is called *fireworks* and results in a spanner of density (essentially) $\binom{n}{2}/2$. It is sparsified further down to $O(n \log n)$ by exploiting a particular dichotomy in the structure of the residual instance. Due to space limitation, several proofs are omitted. They can be found in the full version [7].

2 Definitions and Basic Results

2.1 Model and definitions

Let $G = (V, E)$ be an undirected graph and $\lambda : E \to 2^{\mathbb{N}}$ a mapping that assigns to every edge of E a non-empty set of integer labels. These labels can be seen as discrete times when the edge is present. In this paper, we refer to the resulting graph $\mathcal{G} = (G, \lambda)$ as a *temporal graph* (other models and terminologies exist, all being equivalent for the considered problem). If λ is single-valued and locally injective (i.e., adjacent edges have different labels), then we say that λ is *simple*, and by extension, a temporal graph is simple if its labeling is simple.

A temporal path in $\mathcal{G}$ (also called *journey*), is a finite sequence of k triplets $\mathcal{J} = \{(u_i, u_{i+1}, t_i)\}$ such that $(u_1, \ldots, u_{k+1})$ is a path in G and for all $1 \leq i < k$, $\{u_i, u_{i+1}\} \in E$, $t_i \in \lambda(\{u_i, u_{i+1}\})$ and $t_{i+1} \geq t_i$. Strict temporal path (strict journeys) are defined analogously by requiring that $t_{i+1} > t_i$. We say that a vertex u can *reach* a vertex v iff a journey exists from u to v (strict or non-strict, depending on the context). If every vertex can reach every other vertex, then $\mathcal{G}$ is *temporally connected*. Interestingly, the distinction between strict and non-strict journeys does not exist in simple temporal graphs, as all the journeys are strict.

In general, one can define a *temporal spanner* of $\mathcal{G} = ((V, E), \lambda)$ as a temporal graph $\mathcal{G}' = ((V', E'), \lambda')$ such that $V = V'$, $E' \subseteq E$ and $\lambda' : E' \to 2^{\mathbb{N}}$ with $\lambda'(e) \subseteq \lambda(e)$ for all $e \in E'$. We call $\mathcal{G}'$ a *valid* spanner if it is temporally connected. Observe that, if $\mathcal{G}$ is simple, then spanners are fully determined by the chosen subset of edges $E' \subseteq E$ (as in the above citation from [13]). Thus, in such cases, we say that E' itself *is* the spanner. Many of these notions are analogous to the ones considered in [1, 4, 13], although they are not referred to as "spanners" in these works.

Finally, when the underlying graph G is a complete graph, we call $\mathcal{G}$ a temporal clique. An example of a (valid) temporal spanner of a *simple temporal clique* is shown in Figure 1.

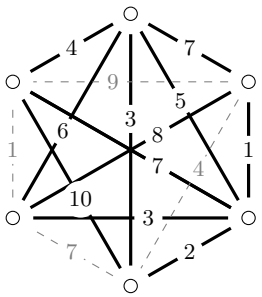


Figure 1 Example of a simple temporal clique and one of its temporal spanners (edges in bold). This spanner is not minimum (nor even minimal) and the reader may try to remove further edges.

2.2 Generality of simple labelings

We claimed in Section 1.2 that if non-strict journeys are allowed, then one can transform a temporal clique $\mathcal{G} = (G, \lambda)$ with *unrestricted* labeling λ into a clique $\mathcal{H} = (G, \lambda_H)$ with *simple* labeling such that any valid temporal spanner of $\mathcal{H}$ induces a valid temporal spanner of $\mathcal{G}$. (As explained, the converse is false, but this is fine because our result is *positive* on $\mathcal{H}$.) The reduction proceeds in two steps: (1) For every edge e, restrict $\lambda(e)$ to a single label chosen arbitrarily; and (2) Whenever k adjacent edges have identical label l, then all labels $l' > l$ are shifted to $l' + k$ and the k labels are assigned a unique value in the interval $[l, l + k]$

(again, arbitrarily). It should not be difficult to see that this reduction produces a simple clique $\mathcal{H}$, such that if a journey exists in $\mathcal{H}$, then the same sequence of edges allows for a (possibly non-strict) journey in $\mathcal{G}$.

As for the case that strict journeys are the only ones allowed, as explained above, we consider that a *simple* sub-labeling of λ exists and is identified prior to the computation. Here, it is even more direct that any journey based on the sub-labeling only is a fortiori available in the complete instance. Based on these arguments, the rest of the paper focuses on simple temporal cliques, sometimes dropping the adjective "simple".

2.3 Preliminary techniques

The best approach so far for sparsifying simple temporal cliques is that of Akrida et al. [1], who prove that one can always remove $\lfloor n/4 \rfloor$ edges without breaking temporal connectivity as follows. (Their paper has other significant contributions.) First, they show that if $n = 4$, then it is always possible to remove at least *one* edge. Then, as $n \to \infty$, one can arbitrarily partition the input clique into (essentially) $n/4$ subcliques of 4 vertices each, and remove an edge from each subclique. The edges *between* subcliques are kept, thus the impact of removals is confined to each subclique. In the full version of the present paper [7], we improve this technique to remove a constant fraction of $\lfloor n^2/12 \rfloor$ edges. However, we consider as unlikely that such purely *structural* techniques could sparsify a graph to $o(n^2)$ edges. The techniques that we develop here are completely different.

Another natural approach that one might think of is discussed in the full version of this paper, inspired by Kosaraju's principle for testing strong connectivity in a directed graph. This principle relies on finding a vertex that all of the other vertices can reach (through directed paths) and that can reach all these vertices in return. This condition is sufficient in standard graphs because paths are transitive. In the temporal setting, transitivity does not hold, but we can define a temporal analogue as follows. A *pivot vertex* p is a vertex such that all other vertices can reach p by some time t (through journeys) and p can reach all other vertices back *after* time t. The union of the tree of (incoming) journeys towards p and the tree of (outgoing) journeys from p is a temporal spanner with at most $2(n-1)$ edges. Unfortunately, pivot vertices may not exist, even in temporal cliques. Both possibilities (positive and negative) are shown in Figure 2. A generic construction to build arbitrarily large non-pivotable graphs is proposed in the full version of this paper.

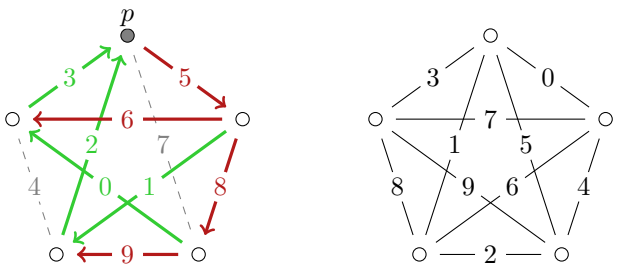


Figure 2 Examples of pivotable graph (left) and non-pivotable graph (right). The (light) green edges in the pivotable graph belong to the tree of incoming journeys to pivot vertex p (with $t = 4$); the (darker) red edges belong to the tree of outgoing journeys; the dashed edges belong to neither.

3 Delegation and Dismountability

This section introduces a number of basic techniques which are subsequently refined and adapted in Sections 4 and 5 in the main algorithm. Given a vertex v, write $e^-(v)$ for the edge with smallest label incident with v, and $e^+(v)$ analogously for the largest label.

▶ **Fact 1.** *Given a temporal clique $\mathcal{G}$, if $\{u,v\} = e^-(v)$, then u can reach all vertices through v. Similarly, if $\{u,w\} = e^+(w)$, then all vertices can reach u through w.*

Fact 1 applies because the underlying graph G is complete. This fact makes it possible for a vertex u to *delegate* its emissions to a vertex v, i.e., exploit the fact that v can still reach all the other vertices (if need be, by a direct edge) *after* interacting with u, thus none of u's other edges are required for reaching the other vertices. By a symmetrical argument, u can delegate its receptions (collections) to a vertex w if w can be reached by all the other vertices (if need be, by a direct edge) *before* interacting with u, so u does not need its other edges to be reached by the other vertices.

This type of delegation suggests an interesting technique to obtain temporal spanners. We say that a vertex u in a temporal clique $\mathcal{G}$ is *dismountable* if there exist two other vertices v and w such that $\{u,v\} = e^-(v)$ and $\{u,w\} = e^+(w)$. The idea of dismountability is to select $e^-(v)$ and $e^+(w)$ for inclusion in a temporal spanner for $\mathcal{G}$ and then reduce the computation to finding a temporal spanner in the smaller clique $\mathcal{G}[V \setminus u]$. We state this more formally in the following theorem.

▶ **Theorem 1** (Dismountability). *Let $\mathcal{G}$ be a temporal clique, and let u, v, w be three vertices in $\mathcal{G}$ such that $\{u,v\} = e^-(v)$ and $\{u,w\} = e^+(w)$. Let S' be a temporal spanner of $\mathcal{G}[V \setminus u]$. Then $S = S' \cup \{\{u,v\},\{u,w\}\}$ is a temporal spanner of $\mathcal{G}$.*

Proof. Since $\{u,v\} = e^-(v)$, all edges incident with v in S' have a larger label than $\{u,v\}$, thus u can reach all the vertices in $\mathcal{G}$ through v and the edges of S'. A symmetrical argument implies that all vertices in $\mathcal{G}$ can reach u through w using only $\{u,w\}$ and the edges of S'. ◀

We call a graph *dismountable* if it contains a dismountable vertex. It is said to be *fully dismountable* if one can find an ordering of V that allows for a recursive dismounting of the graph until the residual instance is a two-vertex graph with a single edge. An example of such dismountable graph is given in Figure 3.

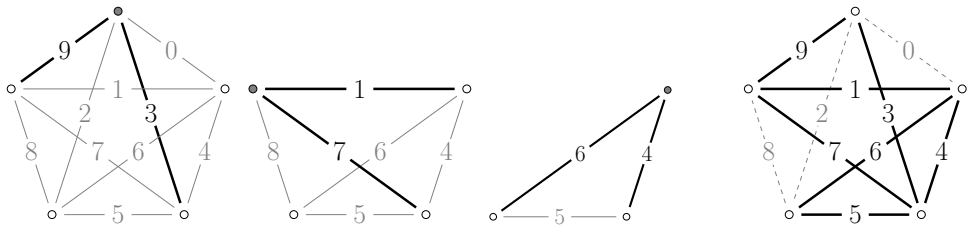


Figure 3 Example of a fully dismountable graph and the resulting spanner.

▶ **Fact 2** (Spanners based on dismountability). *If a graph can be fully dismounted, then the union of the pairs of edges involved in all steps of the recursion, plus the last edge forms a temporal spanner. There are $n-2$ steps, so this spanner has $2(n-2)+1 = 2n-3$ edges.*

Unfortunately, one can design arbitrarily large temporal cliques which are not fully dismountable (we give a generic construction in the full version of this paper). Yet, techniques derived from dismountability are at the core of our algorithm. To start, the concept of dismountability can be generalized to *multi-hop* journeys. The key observation is that a multi-hop journey may exist from a vertex u to another vertex v, say through vertices $u = u_0, u_1, \ldots, u_k = v$ such that $\{u_{k-1}, u_k\} = e^-(v)$, despite the fact that $\{u_{i-1}, u_i\} \neq e^-(u_i)$ for some i. Indeed, it is sufficient that the *last* edge of a journey from u to v is $e^-(v)$ in order to delegate u's emissions to v. Symmetrically, it is sufficient that the *first* edge of a journey from w to u is $e^+(w)$ in order to delegate u's receptions to w. Thus, a vertex u is called *k-hop dismountable* if one can find two other vertices v and w (possibly identical if $k > 1$) such that there are journeys of *at most* k hops (1) from u to v that arrives at v through $e^-(v)$, and (2) from w to u that leaves w through $e^+(w)$.

Temporal spanners can be obtained in a similar way to 1-hop dismountability by selecting all of the edges involved in these journeys for inclusion in the spanner. However, only the edges adjacent to the dismounted vertex are removed in the recursion, thus some edges used in a multi-hop journey may be selected several times over the recursion (with positive impact). We can then extend Fact 2 to k-hop dismountability as follows.

▶ **Fact 3.** *If a temporal graph $\mathcal{G}$ is fully k-hop dismountable, then this process yields a temporal spanner with at most $2k(n-2)+1 \simeq 2kn$ edges.*

Unfortunately, again, there exist arbitrarily large graphs which are not k-hop dismountable for any k (see the full version of this paper [7]). Nonetheless, k-hop dismountability is one of the components of the more sophisticated techniques in Sections 4 and 5.

4 The Fireworks Technique

In this section, we present an algorithm called *fireworks*, which exploits delegations among vertices in a more subtle way than dismountability. In particular, we take advantage of *one-sided delegations*, in which a vertex may be able to delegate only its emissions, or only its receptions. The combination of many such delegations is shown to lead to the removal of essentially half of the edges of the input clique. The residual instance has a particular structure that we exploit in Section 5 to obtain $O(n \log n)$-sparse spanners.

4.1 Forward Fireworks

The purpose of fireworks is to mutualize several one-sided delegations in a transitive way, so that many vertices do not need to reach the others vertices directly, most of their edges being consequently eliminated. Given a temporal clique $\mathcal{G} = (G, \lambda)$ with $G = (V, E)$, define the directed graph $G^- = (V, E^-)$ such that $(u, v) \in E^-$ iff $\{u, v\} = e^-(v)$, except that, if $e^-(u) = e^-(v)$ for some u and v, only one of the arcs is included (chosen arbitrarily).

▶ **Lemma 2.** *Directed paths in G^- correspond to journeys in $\mathcal{G}$.*

By construction, E^- induces a disjoint set of *out-trees* (one source, possibly several sinks). We transform E^- into a disjoint set $\mathcal{T}^- = (V, E_T^-)$ of *in-trees* (one sink, possibly several sources) as follows, see also Figure 4 for an illustration. Let E_T^- be initialized as a copy of E^-. For every v with outdegree at least 2 in E^-, let $(v, u_1), ..., (v, u_\ell)$ be its out-arcs with (v, u_ℓ) being the one with the *largest* label. For every $i < \ell$, if u_i is a sink vertex, then flip the direction of (v, u_i) in E_T^- (i.e., replace (v, u_i) by (u_i, v) in E_T^-); otherwise remove (v, u_i) from E_T^-. Let $\mathcal{T}^- = (V, E_T^-)$ be the resulting set of in-trees $\mathcal{T}_1^-, ..., \mathcal{T}_k^-$ (containing possibly more in-trees than the number of initial out-trees).

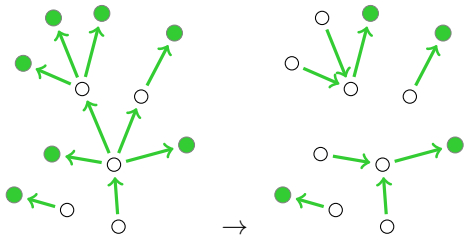


Figure 4 Example of transformation from a disjoint set of out-trees (V, E^-) to a disjoint set of in-trees (V, E_T^-). The colored vertices represent sink vertices.

▶ **Fact 4.** *The set of in-trees $\mathcal{T}^- = (V, E_T^-)$ has the following properties:*

1. *Directed paths in $\mathcal{T}^-$ correspond to journeys in $\mathcal{G}$.*
2. *Every vertex belongs to exactly one tree.*
3. *Every tree contains at least two vertices.*
4. *There is a unique sink in each tree.*
5. *The unique arc incident with a sink s corresponds to $e^-(s)$.*

Fact 4.1 follows from Lemma 2 because an arc (v, u_i) is only replaced by (u_i, v) if the label of (v, u_i) is less than the label of another arc (v, u_ℓ), so $(u_i, v), (v, u_\ell)$ is a journey in $\mathcal{G}$. Observe that some of the journeys induced by the arcs of $\mathcal{T}^-$ may include intermediate hops where the arc's label is not locally minimum for its head endpoint. However, as already discussed in Section 3, a delegation only requires that the label of the last hop of a journey be locally minimum, and that is the case here (Fact 4.5).

The motivation behind this construction is that all the vertices in each in-tree are able to delegate their emissions to the corresponding sink vertex. For this reason, the sink vertex will be called an *emitter* in the rest of the paper. An important consequence of our construction is that the number of emitters in $\mathcal{T}^-$ cannot exceed half of the total number of vertices.

▶ **Lemma 3.** *The number of emitters in $\mathcal{T}^-$ is at most $n/2$*

Proof. After the transformation from E^- to E_T^-, there is only one emitter in each in-tree $\mathcal{T}_i^- \in \mathcal{T}^-$ (Fact 4.4), and at most $n/2$ in-trees, each having at least 2 vertices (Fact 4.3). ◀

We are now ready to define a temporal spanner based on $\mathcal{T}^-$, which consists of the union of all edges involved in an in-tree and all edges incident with at least one emitter. More formally, let $S_T^- = \{\{u, v\} \in E : (u, v) \in \mathcal{T}^-\} \cup \{\{u, v\} \in E : u \text{ is an emitter}\}$.

▶ **Theorem 4.** *S_T^- is a temporal spanner of $\mathcal{G}$.*

Proof. By Fact 4, every vertex v of $\mathcal{G}$ that is a non-emitter in $\mathcal{T}^-$ can reach an emitter s through an edge $e^-(s)$. Furthermore, the inclusion of all edges incident to a vertex s that is an emitter in $\mathcal{T}^-$ ensures that v can still reach all other vertices afterwards and so can s. Therefore, every vertex can reach all other vertices by using only edges from S_T^-. ◀

We call this type of spanner a *forward fireworks cover*. An example is given in Figure 5, the corresponding journeys being depicted on the left side.

▶ **Theorem 5.** *Forward fireworks covers have at most $3\binom{n}{2}/4 + O(n)$ edges.*

Before moving to Section 4.2, we establish a small technical lemma that will be used in Section 5, but is worth being stated here as it pertains to the in-trees.

▶ **Lemma 6.** *Every non-emitter vertex v can reach a vertex v' in the same in-tree $\mathcal{T}_i^-$ (emitter or not) using a journey of length* at most two *that arrives at v' through $e^-(v')$.*

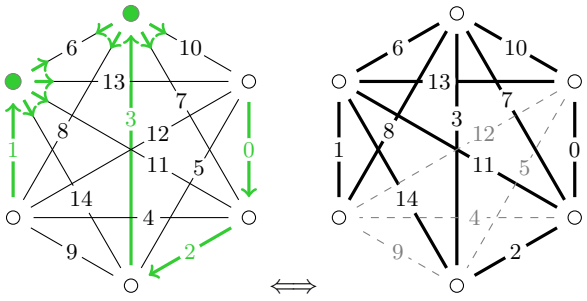


Figure 5 Example of forward fireworks cover and the resulting spanner.

4.2 Backward Fireworks

A symmetrical concept of fireworks can be defined based on the edges $\{u, v\} = e^+(v)$ of a temporal clique $\mathcal{G} = (G, \lambda)$. The above arguments can be adapted in a symmetrical way. First, we build a directed graph $G^+ = (V, E^+)$, which is a disjoint set of *in-trees*. A symmetrical transformation to the above converts this set into a disjoint set $\mathcal{T}^+ = (V, E_T^+)$ of *out-trees*, each with only one source which we call a *collector*. The collector s of an out-tree can reach all of the other vertices in this tree by journeys that leave s through its edge $e^+(s)$, thereby guaranteeing that every vertex that reaches s can reach all other vertices in the tree.

▶ **Lemma 7.** *The number of collectors in $\mathcal{T}^+$ is at most $n/2$*

We build a temporal spanner $S_T^+ = \{\{u, v\} : (u, v) \in \mathcal{T}^+\} \cup \{\{u, v\} : u \text{ is a collector}\}$ which we call a *backward fireworks cover*, and prove the following results by symmetrical arguments to the ones in Subsection 4.1.

▶ **Theorem 8.** *S_T^+ is a temporal spanner of the temporal clique $\mathcal{G}$.*

▶ **Theorem 9.** *Backward fireworks covers have at most $3\binom{n}{2}/4 + O(n)$ edges.*

An example of a backward fireworks cover is given in the full version. Finally, we establish a symmetrical property to the one in Lemma 6, to be used later in Section 5.

▶ **Lemma 10.** *Every non-collector vertex v can be reached by a vertex v' in the same out-tree $\mathcal{T}_i^+$ (collector or not) using a journey of length* at most two *that leaves v' through $e^+(v')$.*

4.3 Bidirectional Fireworks

A forward fireworks cover makes it possible to identify a subset of vertices, the *emitters*, such that every vertex can reach at least one emitter u through $e^-(u)$ and u can reach every other vertex afterwards *through a single edge*. Similarly, a backward fireworks cover makes it possible to identify a subset of vertices, the *collectors*, such that every vertex can be reached by at least one collector v through $e^+(v)$ and v can be reached by every other vertex before this *through a single edge*. Combining both ideas, we can define a sparser spanner that only includes the edges *between* emitters and collectors (plus, of course, the edges used for reaching an emitter and for being reached by a collector).

Precisely, let $\mathcal{T}^-$ be the disjoint set of in-trees obtained during the construction of a forward fireworks cover (see Figure 4), and let $\mathcal{T}^+$ be the disjoint set of out-trees obtained during the construction of a backward fireworks cover. Let X^- be the set of emitters (one per in-tree in $\mathcal{T}^-$) and let X^+ be the set of collectors (one per out-tree in $\mathcal{T}^+$). The two

sets can overlap, as a vertex may happen to be both an emitter in some tree in $\mathcal{T}^-$ and a collector in some tree in $\mathcal{T}^+$, which is not a problem. Let $H = (X^- \cup X^+, E_H)$ be the graph such that $E_H = \{\{u, v\} \in E : u \in X^-, v \in X^+\}$; in other words, H is the subgraph of G that connects all emitters with all collectors. Finally, let $S = \{\{u, v\} : (u, v) \in \mathcal{T}^- \cup \mathcal{T}^+\} \cup E_H$. We call S a bidirectional fireworks cover (or simply a *fireworks cover*).

▶ **Theorem 11.** *S is a temporal spanner of $\mathcal{G}$.*

Proof. Every non-emitter vertex can reach an emitter u through $e^-(u)$. Every emitter can reach *all* collectors afterwards. Every non-collector vertex can be reached by a collector v through $e^+(v)$. ◀

▶ **Theorem 12.** *Bidirectional fireworks covers have at most $\binom{n}{2}/2 + O(n)$ edges.*

5 Recursing or Sparsifying Further

After applying the fireworks technique, one is left with a residual instance (or spanner) made of all the edges between emitters X^- and collectors X^+, together with all the edges corresponding to the arcs of $\mathcal{T}^-$ and $\mathcal{T}^+$, these edges being denoted S^- and S^+ for simplicity. As we will see, the algorithm may recurse several times due to dismountability, thus it is worth mentioning that variables $\mathcal{G}$ and V refer to the instance of the current recursion. The algorithm considers two cases, depending on the outcome of the fireworks procedure. Either $X^- \cup X^+ \neq V$ (Case 1) or $X^- \cup X^+ = V$ (Case 2).

▶ **Case 1 ($X^- \cup X^+ \neq V$).** In this configuration, at least one vertex v is neither emitter nor collector. By Lemma 6, there exists a journey of length at most two from v that arrives at some vertex $u \neq v$ through $e^-(u)$. Similarly, by Lemma 10, there is a journey of length at most two from some vertex $w \neq v$ to v, leaving w through $e^+(w)$. As a result, v is 2-hop dismountable (see Section 3). One can thus select the corresponding edges (at most four) for future inclusion in the spanner and then recurse on $\mathcal{G}[V \setminus v]$, i.e., re-apply the fireworks technique from scratch. Then, either the recursion keeps entering Case 1 and dismounting the graph completely, or it eventually enters Case 2.

▶ **Case 2 ($X^- \cup X^+ = V$).** Both X^- and X^+ have size at most $n/2$ (Lemma 3 and 7), thus if their union is V, then both sets must be disjoint and of size exactly $n/2$. As a result, the graph which connects all vertices in X^- with all vertices in X^+ (called H in Section 4) is a complete bipartite graph. In fact, H possesses even more structure. Firstly, both S^- and S^+ are perfect matchings – by contradiction, if this is not the case, then at least one of the in-tree (out-tree) contains more than one edge, resulting in strictly less emitters (collectors) than $n/2$. Furthermore, every vertex is either an emitter or a collector, thus each of these edges connects an emitter with a collector, implying that the residual instance actually is H itself. Now, recall that every edge in S^- is locally minimum for the corresponding emitter (based on Fact 4.5), and every edge in S^+ is locally maximum for the corresponding collector. We then have the following stronger property.

▶ **Lemma 13.** *If the minimum edge of an emitter is not also minimum for the corresponding collector in H, then the residual instance is 2-hop dismountable. The same holds if the maximum edge of a collector is not also maximum for the corresponding emitter in H.*

Lemma 13 implies that either a vertex v is 2-hop dismountable and the algorithm can recurse as in Case 1, or the edges of the matchings are minimum (resp. maximum) *on both sides*. (An example of the latter case is given in the long version [7].)

In summary, either the algorithm recurses until the input clique is fully dismounted (through Case 1 or Case 2), resulting in an $O(n)$-dense spanner (Fact 3), or the *recursion stops* and the residual instance is sparsified further by a dedicated procedure, described now.

5.1 Sparsifying the Residual Instance

For simplicity, the sparsification of the residual instance is considered as a separate problem. The input is a labeled complete bipartite graph $B = (X^-, X^+, E_B)$ where X^- is the set of emitters, X^+ is the set of collectors, and the labels are inherited from $\mathcal{G}$. There are two perfect matchings S^- and S^+ in B such that the labels of the edges in S^- (resp. S^+) are minimum (resp. maximum) locally to both of their endpoints (Lemma 13). The objective is to remove as many edges as possible from E_B, while preserving S^-, S^+, and the fact that every emitter can reach *all* collectors by a journey. Indeed, these three properties ensure temporal connectivity of the graph (using the same arguments as in Theorem 11).

While both S^- and S^+ are matchings, our algorithm effectively exploits this property with respect to S^+ as follows.

▶ **Fact 5.** *If an emitter can reach another emitter, then it can reach the corresponding collector by adding to its journey the corresponding edge of S^+.*

This property makes it possible to reduce the task of reaching some collectors to that of reaching the corresponding emitter in S^+. It is however impossible for an emitter u to make a complete delegation to another emitter v, because the existence of a journey from u to v arriving through $e^-(v)$ would contradict the fact that S^- is also a matching. For this reason, when a journey from emitter u arrives at emitter v, some of v's edges have already disappeared. Nevertheless, the algorithm exploits such *partial* delegations, while paying extra edges for the missed opportunities (contained within a logarithmic factor). This is done by means of an iterative procedure called *layered delegations*, described over the remaining of this section. Note the term iterative, not recursive; from now on, the instance has a fixed vertex set and it is sparsified until the final bound is reached.

Layered Delegations

The algorithm proceeds by *eliminating* half of the emitters in each step j, while selecting a set S_j of edges for inclusion in the spanner, so that the eliminated emitters can reach all collectors by a mixture of direct edges and indirect journeys through other emitters (partial delegations). The set of non-eliminated emitters at step j (called *alive*) is denoted by X_j^-, with $X_1^- = X^-$. The set of collectors X^+ is invariant over the execution. We denote by $k = n/2$ the initial degree of the emitters in B (one edge shared with each collector), and by $e^i(v)$ the edge with the i^{th} smallest label (label of *rank* i) locally to a vertex v, in particular $e^1(v) = e^-(v)$ and $e^k(v) = e^+(v)$.

The k ranks are partitioned into subintervals of doubling size $\mathcal{I}_j = [2^{j+2} - 7, 2^{j+3} - 8]$, where j denotes the current step of the iteration, ranging from 1 to $\log_2 k - 3$. For simplicity, assume that k is a power of two, we explain below how to adapt the algorithm when this is not the case. For example, if $k = 128$, then $\mathcal{I}_1 = [1, 8], \mathcal{I}_2 = [9, 24], \mathcal{I}_3 = [25, 56]$, and $\mathcal{I}_4 = [57, 120]$. Computation step j is made with respect to the subgraph $B_j = (X_j^-, X^+, E_j)$ where $E_j = \{e^i(v) \in E_B : i \in \mathcal{I}_j, v \in X_j^-\}$, namely the edges of the currently alive emitters, whose ranks are in the interval $\mathcal{I}_j$.

▶ **Lemma 14.** *In each step j, X_j^- can be split into two sets X_a and X_b such that $|X_a| \geq |X_b|$ and every vertex in X_a can reach a vertex in X_b through a 2-hop journey (within B_j).*

Proof. The proof is in the full version [7] (together with an illustration). The main idea is to show that the average degrees of collectors in B_j forces the existence of sufficiently many two-hop journeys among emitters. ◀

▶ Remark 15. The computation of X_a proceeds by repeatedly considering the largest degree d of a collector and assigning $d-1$ of the corresponding emitters to X_a and one to X_b; it is therefore a greedy algorithm. The process is to be stopped whenever X_a reaches half the size of X_j^-. If X_a exceeds this threshold during step j, then some emitters can be arbitrarily transferred from X_a to X_b to preserve the fact that $|X_{j+1}^-|$ is a power of two. The case that $|X_1^-| = k$ is not a power of two is addressed similarly after the first iteration, in order to set the size of X_b to the highest power of two below k.

How X_a and X_b are then used: When an emitter u in X_a can reach another emitter v in X_b, the corresponding journey arrives at v through some edge $e^i(v)$ with $i \in \mathcal{I}_j$. We say that u partially delegates its emissions to v in the sense that all collectors that v can reach after this time can *de facto* be reached from u (through v), the other collectors being possibly no longer reachable from v after this time. Thus, the delegation is *partial*.

▶ **Lemma 16.** *If an emitter u makes a partial delegation to v in step j, then the number of collectors that u may no longer reach through v is at most $2^{j+3} - 8$.*

Proof. This number is the largest value in the current interval; it corresponds to the largest rank of the edge through which the journey from u may have arrived at v. All the edges whose rank locally to v is larger than $2^{j+1} - 2$ can still be used and thus the corresponding collectors are still reachable. (In fact, the collector corresponding to the edge with last index in $\mathcal{I}_j$ locally to v can also be considered as reached, but this is a detail.) ◀

A partial delegation from u to v in step j implies the removal of u from the set of emitters, the selection of the two edges of the journey from u to v, and the selection of at most $2^{j+3} - 8$ direct edges between u and the missed collectors. This implies the following fact.

▶ **Fact 6.** *In each step j, at most 2^{j+3} edges are selected relative to every eliminated emitter.*

More globally, let J_j be the edges used in all the delegation journeys from vertices in X_a to vertices in X_b in step j, and D_j the union of direct edges towards missed collectors. Let $S_j = J_j \cup D_j$. The algorithm thus consists of selecting all the edges in S_j for inclusion into the spanner. Then X_{j+1}^- is set to X_b and the iteration proceeds with the next step. The computation goes for j ranging from 1 to $\log_2 k - 3$, which leaves exactly *eight* final emitters alive. All the remaining edges of these emitters (call them S_{last}) are finally selected. Overall, the final spanner is the union of all selected edges, plus the edges corresponding to the two initial matchings, i.e., $S = (\cup_j S_j) \cup S_{last} \cup S^- \cup S^+$.

▶ **Theorem 17.** *S is a temporal spanner of the complete bipartite graph B and it is made of $O(n \log n)$ edges.*

Proof. The key observation for establishing *validity* of the spanner is that eliminated emitters reach all collectors either directly or through an emitter that can still reach this collector *afterwards*. This property applies transitively (thanks to the disjoint and increasing intervals) until eight emitters remain, all the edges of which are selected for simplicity. Therefore, every initial emitter can reach all collectors. The rest of the arguments are the same as in the proof of Theorem 11: all vertices in the input clique can reach at least one emitter u through $e^-(u)$, and be reached by at least one collector v through $e^+(v)$.

Regarding the size of the spanner, in step j, $\frac{k}{2^j}$ emitters are eliminated and at most 2^{j+3} edges are selected for each of them (Fact 6), amounting to at most $8k = 4n$ edges. The number of iterations is $\Theta(\log k) = \Theta(\log n)$. Finally, the sets S_{last}, S^-, and S^+ each contain $\Theta(n)$ edges (and S^+ is actually included in S_{last}). ◀

▶ **Corollary 18.** *Simple temporal cliques always admit $O(n \log n)$-sparse spanners.*

Proof. In each recursion of the global algorithm, either the residual instance of the fireworks procedure is 2-hop dismountable and the algorithm recurses on a smaller instance induced by a removed vertex, after selecting a *constant* number of edges, or the algorithm computes a $\Theta(n \log n)$-sparse spanner of the residual instance through the layered delegation process. Let n_1 be the number of times the graph is 2-hop dismounted and $n_2 = n - n_1$ be the number of vertices of the residual instance when the layered delegation process begins (if applicable, 0 otherwise). The resulting spanner has $\Theta(n_1) + \Theta(n_2 \log n_2) = O(n \log n)$ many edges. ◀

▶ Remark 19. The running time of the algorithm is polynomial (see the full version).

6 Concluding Remarks

In this paper, we established that sparse temporal spanners always exist in temporal cliques, proving constructively that one can find $O(n \log n)$ edges that suffice to preserve temporal connectivity. Our results hold for non-strict journeys with single or multiple labels on each edge, and strict journeys with single or multiple labels on each edge with the property that there is a subset of locally exclusive single labels. Our results give the first positive answer to the question of whether any class of dense graphs always has sparse temporal spanners.

To prove our results, we introduced several techniques (pivoting, delegation, dismounting and k-hop dismounting, forward and backward fireworks, partial delegation, and layered delegations), all of which are original and some of which might be of independent interest. Whether some of these techniques can be used for more general classes of graphs is an open question. Delegation and dismounting rely explicitly on the graph being complete; however, refined versions of these techniques like partial delegation might have wider applicability.

An open question is whether sparse spanners always exist in more general classes of dense graphs, keeping in mind that some dense graphs are unsparsifiable. Another question is whether a better density than $O(n \log n)$ could be obtained in the particular case of temporal cliques, in particular $O(n)$-dense spanners. At a deeper level, all these questions pertain to identifying and studying analogues of spanning trees in temporal graphs, which do not enjoy the same matroid structure as in standard graphs.

References

1 Eleni C. Akrida, Leszek Gasieniec, George B. Mertzios, and Paul G. Spirakis. The complexity of optimal design of temporally connected graphs. *Theory of Computing Systems*, 61(3):907–944, 2017.

2 Sunil Arya, David M Mount, and Michiel Smid. Dynamic algorithms for geometric spanners of small diameter: Randomized solutions. *Computational Geometry*, 13(2):91–107, 1999.

3 B. Awerbuch and S. Even. Efficient and reliable broadcast is achievable in an eventually connected network. In *Proceedings of 3rd Symposium on Principles of Distributed Computing (PODC)*, pages 278–281, 1984.

4 Kyriakos Axiotis and Dimitris Fotakis. On the Size and the Approximability of Minimum Temporally Connected Subgraphs. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 149:1–149:14, 2016.

5 Bin-Minh Bui-Xuan, Afonso Ferreira, and Aubin Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(2):267–285, 2003.

6 Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.

7 Arnaud Casteigts, Joseph G Peters, and Jason Schoeters. Temporal Cliques admit Sparse Spanners. *arXiv preprint*, 2019. `arXiv:1810.00104`.

8 Paul Chew. There is a planar graph almost as good as the complete graph. In *Proceedings of 2nd Symposium on Computational Geometry*, pages 169–177, 1986.

9 Michael Elkin. A near-optimal distributed fully dynamic algorithm for maintaining sparse spanners. In *Proceedings of 26th ACM Symposium on Principles of Distributed Computing*, pages 185–194, 2007.

10 Lee-Ad Gottlieb and Liam Roditty. Improved algorithms for fully dynamic geometric spanners and geometric routing. In *19th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 591–600, 2008.

11 Petter Holme and Jari Saramäki. *Temporal Networks*. Springer, 2013.

12 D. Kempe, J. Kleinberg, and A. Kumar. Connectivity and inference problems for temporal networks. In *32nd ACM Symposium on Theory of Computing (STOC)*, pages 504–513, 2000.

13 David Kempe, Jon Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. *Journal of Computer and System Sciences*, 64(4):820–842, 2002.

14 Othon Michail and Paul G Spirakis. Elements of the theory of dynamic networks. *Communications of the ACM*, 61(2):72–72, 2018.

15 Giri Narasimhan and Michiel Smid. *Geometric Spanner Networks*. Cambridge Univ. Press, 2007.

16 Ariel Orda and Raphael Rom. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of the ACM (JACM)*, 37(3):607–625, 1990.

17 David Peleg and Alejandro A Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989.

18 Tiphaine Viard, Matthieu Latapy, and Clémence Magnien. Computing maximal cliques in link streams. *Theoretical Computer Science*, 609:245–252, 2016.

19 Philipp Zschoche, Till Fluschnik, Hendrik Molter, and Rolf Niedermeier. The Complexity of Finding Small Separators in Temporal Graphs. In *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 45:1–45:17, 2018.

Distributed Reconfiguration of Maximal Independent Sets

Keren Censor-Hillel
Department of Computer Science, Technion, Israel
ckeren@cs.technion.ac.il

Mikaël Rabie
IRIF, Université de Paris, France
Aalto University, Finland
mikael.rabie@irif.fr

Abstract

In this paper, we investigate a *distributed maximal independent set (MIS) reconfiguration problem*, in which there are two maximal independent sets for which every node is given its membership status, and the nodes need to communicate with their neighbors in order to find a *reconfiguration schedule* that switches from the first MIS to the second. Such a schedule is a list of *independent sets* that is restricted by forbidding two neighbors to change their membership status at the same step. In addition, these independent sets should provide some covering guarantee.

We show that obtaining an actual MIS (and even a 3-dominating set) in each intermediate step is impossible. However, we provide efficient solutions when the intermediate sets are only required to be independent and 4-dominating, which is almost always possible, as we fully characterize.

Consequently, our goal is to pin down the tradeoff between the possible length of the schedule and the number of communication rounds. We prove that a constant length schedule can be found in $O(\texttt{MIS} + \texttt{R32})$ rounds, where `MIS` is the complexity of finding an MIS in a worst-case graph and `R32` is the complexity of finding a $(3, 2)$-ruling set. For bounded degree graphs, this is $O(\log^* n)$ rounds and we show that it is necessary. On the other extreme, we show that with a constant number of rounds we can find a linear length schedule.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Distributed algorithms

Keywords and phrases distributed graph algorithms, reconfiguration, maximal independent set

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.135

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Related Version `https://arxiv.org/abs/1810.02106`

Funding *Keren Censor-Hillel*: This project has received funding from the European Union's Horizon 2020 Research And Innovation Programe under grant agreement no. 755839.
Mikaël Rabie: This work was supported in part by the Academy of Finland, Grant 285721 and ANR DESCARTES (ref. DS0702-2016).

Acknowledgements The authors thank Alkida Balliu, Michal Dory, Seri Khoury, Dennis Olivetti, and Jukka Suomela for helpful discussions.

1 Introduction

Consider a distributed setting in which each node of a network receives an input from a higher-level application which tells it whether it is *selected* or not, such that the set of selected nodes is a maximal independent set (MIS), which we will denote by α. The reason that the application requires an MIS is because it needs the set of selected nodes to dominate

all nodes for the sake of, say, monitoring the network, but without having violations of two neighbors being in the set, because they may cause conflicting actions. Now, because of changes in the network traffic, the energy consumption, or any one of various conditions that may change, the application needs to change the selected set of nodes. Once a new input MIS, denoted by β, is given to the nodes by the application, the nodes need to *reconfigure* their states to that set while never sacrificing the safety condition of independence. In fact, for compatibility reasons, neighboring nodes cannot change their membership in the set at the same time, so a *sequence* of changes is needed for converging into the new MIS. We call such a sequence a *reconfiguration schedule*.

The length of the schedule is clearly a measure that is required to be minimized. Hence, an extreme solution would be to have all nodes declare themselves as unselected, and then the new set of nodes declare that they are selected. However, this very fast approach suffers from loosing the domination property throughout the reconfiguration schedule. Thus, the structure of the network must be taken into account, but since the topology is unknown, finding a schedule that maintains a good covering at all times necessitates that the nodes communicate. This brings another measure of complexity into question, which is the number of communication rounds that are needed in order to find a short schedule. Our goal in this work is to study the tradeoff between the possible length of the schedule and the number of communication rounds needed for finding it.

Unfortunately, as we show, it is not always possible to find schedules where each set is an MIS. This impossibility holds even if we relax the condition of domination and require only independent 3-dominating sets. Even when 3-domination *is* possible, it may be extremely inefficient (Section 6).

▶ **Theorem 1.** *Requiring* 3*-domination for intermediate steps is costly:*

1. *There exists a class of inputs $G = (V, E)$ with two MIS α and β such that there is no reconfiguration schedule with 3-dominating intermediate steps.*
2. *There exists a class of inputs $G = (V, E)$ with two MIS α and β such that any reconfiguration schedule is of length $\Omega(n)$ and needs $\Theta(n)$ rounds to be found, if intermediate steps must be 3-dominating.*

However, we prove that independence and 4-domination can indeed be obtained. Our main result is the following (Section 3).

▶ **Theorem 2** (informal)**.** *For any graph $G = (V, E)$ of diameter greater than 3 and any input of two MIS α, β, there exists a reconfiguration schedule of constant length* 28*, with independent 4-dominating intermediate steps. Moreover, such a schedule can be found in $O(\texttt{MIS} + \texttt{R32})$ rounds, where* `MIS` *is the complexity of finding an MIS on a worst-case graph and* `R32` *is the complexity of finding a $(3, 2)$-ruling set on a worst-case graph.*

Obtaining the above theorem turns out to be an involved task. Our key ingredients are the following. We prove that graphs with a not-too-small diameter always admit a schedule of reconfiguration steps from a given maximal independent set to another. Moreover, full knowledge of the topology of the graph is not necessary in order to be able to locally add an element to the set after having removed its neighbors (to avoid dependence). Rather, only local manipulations are needed for doing so.

The currently known complexities that give $O(\texttt{MIS} + \texttt{R32})$ are discussed in the related work part. Here, we draw attention to the fact that an immediate corollary of Theorem 2 is that for graphs of bounded degree we can compute the constant length schedule within $O(\log^* n)$ rounds. Further, we show that this is a lower bound by reducing the problem of

finding an MIS on a path to obtaining a constant-length schedule for MIS reconfiguration. The following theorem actually holds even if one requires only d-domination, for some constant $d \geq 4$ (Section 6).

▶ **Theorem 3.** *For any fixed k, there exists a class of k-regular inputs $G = (V, E)$ with two MIS α and β such that any reconfiguration schedule of constant length with 4-domination needs $\Theta(\log^* n)$ rounds to be found.*

If one wants to optimize the communication cost of finding a schedule rather than its length, we show that a (rather lengthy) schedule can be obtained within $O(1)$ rounds (Section 4).

▶ **Theorem 4** (informal). *For any graph $G = (V, E)$ and any input of maximal independent sets α, β to the MIS-reconfiguration problem, there exists a reconfiguration schedule of length $\Theta(f(n))$, where $f(n)$ is the largest identifier among the nodes in the graph, which can be found in $O(1)$ rounds.*

The construction generalizes itself on graphs with a distance-k coloring of c colors, with k big enough. It is possible, from this coloring, to compute a schedule of length $O(c)$ after a constant number of communication rounds. Let Δ be the maximal degree of the graph. A distance-k $O(\Delta^{2k})$ coloring can be found in $O(\log^* n)$ rounds [16], and a distance-k $O(\Delta^k)$ coloring can be found in $O(\log^* n + \sqrt{\Delta^k})$ rounds [4]. Hence, with the same respective communication complexities, we can find schedules of lengths $O(\Delta^{2k})$ and $O(\Delta^k)$.

Finally, as can be inferred from Theorem 2, 4-domination suffices for any graph with diameter greater than 3. For graphs with small diameter, we give an exact characterization of the conditions that allow the existence of a reconfiguration schedule (Section 5). This result implies that our algorithm from Theorem 2, combined with a trivial algorithm that collects the entire graph when the diameter is a small constant, produces an efficient reconfiguration schedule in all cases for which it exists.

1.1 Related work

Distributed Reconfiguration. Questions of distributed reconfiguration were actually not studied before 2018. Then, Bonamy et al. [6] considered distributed reconfiguration of colorings, with the goal of finding which length of schedule can be computed within a given number of communication rounds. The problem being PSPACE complete in the general case, several subcases were explored. Since finding looser restrictions for the transitions is important for making the problem local instead of having to solve a global PSPACE hard problem, the addition of extra colors in the intermediate colorings was allowed. This aided either having a solution, or finding one quickly.

Distributed Constructions. Our constructions sometimes make use of two fundamental subroutines, which find an MIS or a $(3, 2)$-ruling set in a graph. An (x, y)-ruling set is a set $S \subseteq V$ in which every two nodes are at distance at least x, and every node that is not in S is within distance at most y from S. Thus, an MIS is a $(2, 1)$-ruling set. Finding an MIS is one of the most fundamental problems in distributed computing. The celebrated randomized $O(\log n)$-round algorithms of Luby [19] and Alon et al. [1] have been recently improved by Ghaffari to $O(\log \Delta + 2^{O(\sqrt{\log \log n})})$ rounds, where Δ is the maximal degree in the network [9]. Deterministic solutions are the classic network-decomposition based algorithm of Panconesi and Srinivasan that runs in $2^{O(\sqrt{\log n})}$ rounds [21], and the $O(\Delta + \log^* n)$-round algorithm of Barenboim et al. [5]. The classic lower bound of Linial [17] shows that $\Omega(\log^* n)$ rounds

are necessary, Kuhn et al. gives a higher bound of $\Omega(\log \Delta / \log\log \Delta, \sqrt{\log n / \log\log n})$ [15]. The latest results of Balliu et al. [3] give the new best known lower bounds to find a MIS: There is no deterministic algorithm in $o(\Delta + \frac{\log n}{\log\log n})$ nor randomized algorithm in $o(\Delta + \frac{\log\log n}{\log\log\log n})$. The Figure 1 in [3] sums up all the results on MIS. A $(3, 2)$-ruling set can be computed by computing an MIS over G^2, and more general ruling sets have been studied in [2, 13, 14, 20, 22].

Centralized Reconfiguration of Maximal Independent Sets. Reconfigurations problems on graphs have been widely studied in the centralized setting during the last decade. An excellent survey on reconfiguration problems can be found in [23]. In the centralized setting, the transition rules are different, requiring that any intermediate set must be at least of a certain size. While having their own motivation in that setting, these rules are not the ones that are needed in the distributed setting, as they do not give covering guarantees (moreover, such properties would be costly to obtain in a distributed setting, due to their global nature).

In more detail, three kinds of transitions have been studied for the independent set reconfiguration problem. *Token Addition and Removal* [11], or $TAR(k)$: at each transition, one vertex is removed from or added to the current independent set, as long as there are at least k nodes in the independent set. *Token Jumping* [12]: at each transition, one vertex is removed from the independent set and another one is added somewhere else. *Token Sliding* [10]: at each transition, an edge containing a vertex of the independent set is chosen. This vertex is removed from the set and its neighbor on the other side of that edge is added to the set. The two first versions are actually equivalent when k corresponds to the size of the independent sets minus 1. Reconfiguration problems are in PSPACE, and independent set reconfiguration problems are in general PSPACE complete [10]. Studies over subclasses of graphs exist, and some polynomial algorithm or hardness proofs are given. For example, planar graphs [10], perfect graphs [12], trees [8] and bipartite graphs [18].

2 Preliminaries

We work in the classic LOCAL model of computation, in which n nodes in a synchronous network exchange messages with their neighbors in each round of computation.

Let $G = (V, E, U)$ denote a graph with an assigned subset $U \subseteq V$. An input to the MIS-reconfiguration problem is a pair $G_{input} = (V, E, \alpha), G_{output} = (V, E, \beta)$, where α and β are the initial and final maximal independent sets, respectively. We refer to a node $v \in \alpha$ as an *α-node*, and to a node $v \in \beta$ as a *β-node*. Notice that a node may be both an α-node and a β-node. We refer to node $v \in V \setminus (\alpha \cup \beta)$ as an *ϵ-node*. Throughout the proofs, we say that a node v is *covered* or *4-dominated* by a node u if $d(v, u) \leq 4$.

For a vertex $v \in V$, we denote by $N(v)$ the set of neighbors of v (i.e., $N(v) = \{u \in V : (u, v) \in E\}$), and given a set $U \subseteq V$ we define $N_U(v) = U \cap N(v)$ for the subset of neighbors of v that are in U, and we call this set the U-neighbors of v. For a subset $U \subseteq Y \subseteq V$ and a node $v \in Y$, we denote by $d_Y(v, U)$ the distance of v from U in the subgraph induced by Y.

▶ **Definition 5** (Reconfiguration Schedules). *For a given property P of $G = (V, E, U)$, an* (α, β, P)-reconfiguration schedule *(or simply a schedule) S of length ℓ is a sequence of subsets of V, $S = (S_0, \ldots, S_\ell)$, such that the following hold:*

1. *$S_0 = \alpha$ and $S_\ell = \beta$,*
2. *for every $0 < i < \ell$, the graph (V, E, S_i) satisfies P, and*
3. *for every $0 < i \leq \ell$, $S_i \oplus S_{i-1}$ is an independent set of (V, E).*

3 An MIS reconfiguration schedule of constant length

Our main theorem is the following.

▶ **Theorem 2** (formal). *Let P be the property of (V, E, U) that says that U is a (2,4)-ruling set. For any graph $G = (V, E)$ of diameter greater than 3 and any input $G_{input} = (V, E, \alpha), G_{output} = (V, E, \beta)$ to the MIS-reconfiguration problem, there exists an (α, β, P)-reconfiguration schedule of constant length 28. Moreover, such a schedule can be found in $O(\texttt{MIS} + \texttt{R32})$ rounds, where $\texttt{MIS}$ is the complexity of finding an MIS on a worst-case graph and $\texttt{R32}$ is the complexity of finding a $(3, 2)$-ruling set on a worst-case graph.*

In particular, Theorem 2 immediately implies a highly efficient solution for bounded degree graphs.

▶ **Corollary 6.** *The constant length schedule of Theorem 2 can be found in $O(\log^* n)$ rounds in graphs of bounded degree.*

We now describe the outline of the algorithm, as follows. Denote by W the set of connected components of $\alpha \cup \beta$. Our main approach is to reconfigure the independent sets according to the components in W. To this end, we first categorize each component in W according to its diameter and whether it is isolated or not: We say that a component $V_i \in W$ is *isolated* if for every ϵ-node u in its neighborhood, $N_\alpha(u)$ and $N_\beta(u)$ are contained in V_i.

Notice that within a constant number of rounds, all α and β-nodes can know whether they are in a component of diameter 0, 1, 2, or at least 3. Moreover, if their diameter is smaller than 3, they can know whether the component is isolated or not.

To avoid excessive notation, we will sometimes say that *we update the component V_i in steps $\{j, j+1\}$*. This means that we remove $\alpha \cap V_i$ from the independent set in step j and we add $\beta \cap V_i$ to the set in step $j+1$. Formally, this means that $S_j = S_{j-1} \setminus (\alpha \cap V_i)$ and $S_{j+1} = S_j \cup (\beta \cap V_i)$. Since we will sometimes update multiple components concurrently, we will have $S_j = S_{j-1} \setminus (\alpha \cap Z_j)$ and $S_{j+1} = S_j \cup (\beta \cap Z_j)$, where $Z_j = \bigcup_{i \in I_j} V_i$, with $I_j = \{i : V_i \text{ is being updated in steps } \{j, j+1\}\}$.

The high-level description of our algorithm is as follows. First, for components in W of diameter 0, we do not need to do anything, as such components are comprised only of nodes in $\alpha \cap \beta$. These nodes remain in the independent set S_i for the entire schedule, and we omit these components and all of their ϵ-neighbors from the remaining discussion. Our algorithm then handles non-isolated components and components of diameter ≥ 3, and finally handles the isolated components of diameter ≤ 2.

We begin by claiming that with an overhead of 2 rounds, we may assume that α and β are disjoint. Indeed, if we never remove nodes in $\alpha \cap \beta$ from the independent set, we no longer need to take care about $\alpha \cap \beta$ nor its neighborhood. You can find a more complete explanation of this in the full version of the paper [7].

3.1 Components of diameter ≥ 3

We continue with the following lemma, which is useful for handling components in W whose diameter is not too small. Roughly speaking, the way we handle components of sufficient diameter is by finding a set of α-nodes that are not too close to each other to ensure that β-nodes can be added not too far from them before we remove them from the independent set. This way, we can reconfigure the rest of the component, and then this set of α-nodes and their neighbors. We present the following lemma before the rest of the algorithm because we will need to use it, but notice that it is not the case that we begin the algorithm by reconfiguring components of diameter ≥ 3.

▶ **Lemma 7.** *Let P be the property of (V, E, U) that says that U is a (2,4)-ruling set, and let $G = (V, E, \alpha)$ and $G = (V, E, \beta)$ be an input to the MIS-reconfiguration problem such that $\alpha \cap \beta = \emptyset$, the set $Y = \alpha \cup \beta$ is a single connected component of diameter at least 3, and each ϵ-node is connected to an α-node and to a β-node. Then, there exists an (α, β, P)-reconfiguration schedule of length 8. Moreover, such a schedule can be found in $O(\texttt{R32})$ rounds.*

Proof. First, assume that the diameter of $Y = \alpha \cup \beta$ is either 3 or 4. Consider a shortest path of length 3 in Y, denoted by (v_1, v_2, v_3, v_4). Either v_1 or v_4 is in α, and is within distance 4 from all other nodes in the component. We denote this node by v, and define $S_1 = \{v\}$ and we have that it 4-dominates the entire component. In addition, it 4-dominates all ϵ-nodes, by the assumption of the lemma that all such nodes are neighbors of β-nodes, because v actually 3-dominates all β-nodes in the component. We then denote $R = \{u \in \beta \mid u \notin N(v)\}$ and define $S_2 = \{v\} \cup R$, and $S_3 = R$, and finally $S_4 = R \cup N(v)$. It is easy to verify that this results in a valid (α, β, P)-reconfiguration schedule. In particular, notice that, without loss of generality, if $v = v_1$, then R contains at least the node v_4, which 4-dominates the entire component.

For a diameter of Y that is at least 5, the high-level idea of the construction is as follows. Consider a $(3, 2)$-ruling set R over the nodes in α, where we imagine an edge between two nodes in α if they are at distance two in the subgraph induced by Y. We reconfigure all β-nodes that are at distance 5 from R in G by removing their α-neighbors first, then by adding them. Then, we do the same for β-nodes that are at distance 3 from R, and finally we repeat this one last time for the β-nodes in the direct neighborhood of R. The choice of a $(3, 2)$-ruling set ensures that all α-nodes in R have a β-node at distance 3 that will be reconfigured in the 4th step. However, while we trust β-nodes at distance 3 from R to cover α-nodes at distance 2 from R while R itself is being reconfigured, we must be careful when handling α-nodes at distance 2 from R that do not have a neighbor at distance 3. We overcome this caveat by taking care of these nodes separately.

Formally, we define a virtual multigraph $\tilde{G} = (\tilde{V}, \tilde{E})$ as follows. The set of virtual nodes $\tilde{V}$ consists of all α-nodes. If v and u in $\tilde{V}$ have a common β-neighbor, we add an edge u, v to $\tilde{E}$. Let R be a $(3, 2)$-ruling set in $\tilde{G}$. It is easy to see that in G, the set of nodes R is a $(6, 5)$-ruling set of Y. We denote R by R_0 and we define R_i for $3 \leq i \leq 5$ as $R_i = \{v \in Y \mid \text{the distance of } v \text{ from } R \text{ in the subgraph induced by } Y \text{ is } i\}$. Then we define $R_1 = \{v \in Y \mid d_Y(v, R) = 1 \text{ and } d_Y(v, R_3) = 2\} \cup \{v \in Y \mid N_Y(v) \subseteq R\}$, which captures all β-neighbors of R that either do not have other α-neighbors, or have other α-neighbors which in turn have β-neighbors that are farther from R. We separate those from the set $R_{-1} = N_Y(R_0) \setminus R_1$. We complete the partition by defining $R_2 = N_Y(R_1) \setminus R_0$ and $R_{-2} = N_Y(R_{-1}) \setminus R_0$. Note that for even i, R_i contains only α-nodes, and for odd i, R_i contains only β-nodes. We have that, for every $-2 \leq i \leq 5$, $N_Y(R_i) \subseteq R_{i-1} \cup R_{i+1}$ (with $R_{-3} = R_6 = \emptyset$). By construction of R, we have that each node in R has a node at distance 3 in R_3, hence it has a node at distance 2 in R_2 and a node at distance 1 in R_1.

We define $S_0 = \alpha$ and for $i = 0, 1, 2, 3$, we define $S_{2i+1} = S_{2i} \setminus R_{4-2i}$, $S_{2(i+1)} = S_{2i+1} \cup R_{5-2i}$. We claim that $S_0, \ldots, S_8$ is an (α, β, P)-reconfiguration schedule.

First, $S_0 = \alpha$ by definition, and because every β-node is within an odd distance of at most 5 from R and every α-node is within an even distance of at most 4 from R, we have that $S_8 = \beta$. This gives condition (1) of Definition 5.

For condition (2), it is easy to see that $S_i \oplus S_{i-1}$ is an independent set of (V, E) for every $1 \leq i \leq 8$. For an odd i this holds because to obtain S_i we only remove α-nodes from S_{i-1}, and no two such nodes can be neighbors. For an even i this holds because to obtain S_i we only add β-nodes to S_{i-1}, and no two such nodes can be neighbors.

It remains to show condition (3) of Definition 5. To show that S_i is independent for $i = 2, 4, 6$, notice that β-nodes in R_j (for $j = -1, 1, 3, 5$) are only added to the sequence after all α-nodes in R_k for $k \geq j - 1$ have been removed. By definition, S_0 is also independent. Hence, for $i = 1, 3, 5, 7$, S_i is independent because it is a subset of S_{i-1}.

Next, we need to show that S_i is 4-dominating for every $1 \leq i \leq 7$. Our focus will be for $i = 1, 3, 5, 7$, and for $i = 2, 4, 6$ it then follows because S_i contains S_{i-1}. We first show it on Y, and will prove it for ϵ-nodes afterward. For $i = 1$ this holds because all nodes in R_j for $j \leq 3$ are in or have neighbors in $R_{-2} \cup R_0 \cup R_2$. All nodes in R_j for $j = 4, 5$ are within distance 3 from R_2. Similarly, S_3 is 4-dominating because nodes in R_j for $j \leq 4$ are covered by R_0, nodes in R_5 are in the current independent set. For S_5, recall that for any node in R, there is a node at distance 3 from it in R_3, that node being in the current independent set since S_4. Hence, R_3 covers R_j for $-1 \leq j \leq 5$. R_{-2} is still included in S_5. Finally, for S_7, R_3 still covers R_j for $-1 \leq j \leq 5$. For each node in R_{-2}, there is a node at distance 3 from it in R_1 that has been added in S_6 that covers it.

Now, let u be an ϵ-node that has a node $a \in \alpha$ and $b \in \beta$ in its neighborhood. We show that a or b are always 3-dominated throughout the sequence. In a step where b has no α-neighbor in the independent set, it must be a step right before b gets added to the independent set. If b is in R_5 or in R_3 then when this happens, it is 3-dominated by an α-node in R_2 or R, respectively, and this node is still in the independent set. If $b \in R_{-1}$ then it is 2-dominated by nodes in R_{-2} and then R_1 (with an overlap in S_6, the construction ensures that such node exist at distance at most 3 from b). Finally, If $b \in R_1$ then either there is a β-node in R_3 that 2-dominates it, and this node is already in the independent set, or b is in $\{v \in Y \mid N_Y(v) \subseteq R\}$. Only in the latter case, we must resort to the α-neighbor of u and check that it is 3-dominated by S_5, as we removed R from S_5 and b is added in S_6.

Let i be such that $a \in R_i$. We need to make sure that a is 3-dominated at the step in which we reconfigure R_1. At this step, all of the β-nodes in R_3, R_5 are in the independent set, and hence their α-neighbors in R_2, R_4 are covered by nodes in distance 1, and nodes in R_0 are covered by nodes in distance 3. For α-nodes in R_{-2} they are still in the independent set at this step, and hence are 3-dominated.

This completes the correctness proof. For the round complexity, notice that simulating the $(3, 2)$-ruling set over $\tilde{G}$ can be done in G with a constant overhead. ◀

3.2 Non-isolated components

We first observe that components of diameter ≤ 2 are such that there is a complete bipartite graph between their α-nodes and β-nodes. Let u be an ϵ-node that is a neighbor of several components. Let W_u be the set of all components that are its neighbors, so that in particular, $V_i, V_j \in W_u$. For each pair of distinct components $V_i, V_j \in W_u$, if there is an α node in $N_\alpha(u) \cap V_i$ and a β node in $N_\beta(u) \cap V_j$, then we say that V_j is (u, α)-covered and that V_i is (u, β)-covered (note that this definition allows a single component to satisfy both conditions). As u is an ϵ-node, there must exist a component $V_{u,\alpha} \in W_u$ that is (u, β)-covered and a component $V_{u,\beta} \in W_u$ that is (u, α)-covered.

We say that a component $V_i \in W$ is α-covered (β-covered) if there is an ϵ-node u for which V_i is (u, α)-covered ((u, β)-covered). A component that is both is $\alpha\beta$-covered.

The key insight is that a (u, α)-covered component of diameter ≤ 2 is covered (dominated at distance 4) by some α-node of the component $V_{u,\beta}$ (and similarly with the β-node of $V_{u,\alpha}$). Moreover, any ϵ-node that is connected to an α-node (a β-node) in that component is covered by $V_{u,\beta}$ (or $V_{u,\alpha}$). This implies that an ϵ-node that is connected to two components that are updated in different steps is always covered by the component that is currently not

being updated. However, during the reconfiguration schedule, we need to be careful about ϵ-nodes that are connected to a single component, and ϵ-nodes that are connected to two components that are updated at the same time.

We denote by $C_{\alpha\beta}$ the set of $\alpha\beta$-covered components of diameter ≤ 2, and by C_α and C_β the sets of α-covered and β-covered components of diameter ≤ 2 that are not in $C_{\alpha\beta}$, respectively. Define the component graph $\tilde{G} = (W, \tilde{E})$, where there is an edge between $V_i, V_j \in W$ iff there exists an ϵ-node u such that V_i is (u, α)-covered and V_j is (u, β)-covered, or vice-versa. Notice that in $\tilde{G}$, the sets C_α and C_β are two disjoint independent sets.

We are finally ready to formally provide the algorithm for handling all components that are either non-isolated or have diameter ≥ 3.

▶ **Lemma 8.** *Let P be the property of (V, E, U) that says that U is a (2,4)-ruling set, and let $G = (V, E, \alpha)$ and $G = (V, E, \beta)$ be an input to the MIS-reconfiguration problem such that $\alpha \cap \beta = \emptyset$, and all connected components of $\alpha \cup \beta$ are either non-isolated or have diameter at least 3. Then, there exists an (α, β, P)-reconfiguration schedule of length 18. Moreover, such a schedule can be found in $O(\texttt{MIS} + \texttt{R32})$ rounds.*

Proof. Our reconfiguration schedule works according to the following *parts*.

1. Update components of diameter ≤ 2 in C_α.
   %Let M be an MIS over all nodes in $C_{\alpha\beta}$.
2. Update components of diameter ≤ 2 that are α-covered by a component in M.
3. Reconfigure components of diameter ≥ 3 using the schedule given by Lemma 7.
4. Update components in M.
5. Update components of diameter ≤ 2 in $C_{\alpha\beta}$ that were not previously updated.
6. Update components of diameter ≤ 2 in C_β that were not previously updated.

First, it is easy to see that the schedule has length 18. The part that reconfigures components of diameter ≥ 3 requires 8 steps, by Lemma 7. Each of the other 5 parts takes exactly 2 steps as described in the definition of updating components (removing α-nodes and then adding β-nodes), which sums to 18 reconfiguration steps in the schedule.

It remains to prove correctness. First, condition (1) of Definition 5 trivially holds, as the schedule reconfigures all nodes. Moreover, by Lemma 7 and by the definition of updating a component, it is also immediate that we do not reconfigure two neighbors in a single step, thus the schedule satisfies condition (3) of Definition 5. For condition (2), Lemma 7 and the definition of updating a component also guarantee that each S_i is an independent set. The remainder of the proof shows that each S_i in the schedule is also 4-dominating.

By the order of the reconfiguration steps in the schedule, each component that is being updated is covered by a component that is not concurrently being updated. This also holds for ϵ-nodes that are connected to a component that is not currently being updated. The main condition that must be verified is that ϵ-nodes remain covered even if all of their neighboring components are being concurrently updated in a certain part of the schedule.

Part 1 guarantees that S_1, S_2 are 4-dominating because for each component that is being updated, the α-node covering it is a member of S_1, S_2 and it also covers the required ϵ-nodes that are neighbors of the updated component, as explained earlier. For part 2, let u be an ϵ-node that is connected to two of the components that are being updated and is not connected to any component that is not being updated. One of the components must be connected to u via an α-node. Let V_i be such a component, let u_1 be the α-node connected to u, and let v be the α-node from a component of M that covers V_i. The distance between v and u_1 is 3: v is a distance 2 to a β-node of V_i and, because V_i is of diameter ≤ 2, within V_i all β-nodes are connected to all α-nodes. Hence, u is at distance 4 from v.

For part 3, the 4-domination within the components that are being reconfigured is given by Lemma 7. Notice that any ϵ-node connected to a component of diameter ≥ 3 is connected either to connected components of diameter ≥ 3 through both an α and a β-node, or to a component that is not being updated in those steps. In the first case it is covered by Lemma 7, and in the second it is covered by the component not being concurrently updated.

For part 4, notice that all components that are β-covering components of M have been updated in steps 2 or 3. Hence, as each component of M is in $C_{\alpha\beta}$, there is a β-node in the current set S_{12} that covers it. As M is independent, we do not have ϵ-node between two components that are being updated. For part 5, the ϵ-nodes between two components that are being updated are covered by an argument that is symmetric to the one used for part 2. Finally, for part 6, for each component that is being updated it holds that the β-node covering it is in a component that has already been updated and hence it is already in S_{16}.

Finally, we note that apart from a constant overhead in communication, the number of rounds required for computing the above schedule is proportional to that of finding the MIS M plus solving the diameter ≥ 3 components, which completes in $O(\texttt{MIS} + \texttt{R32})$ rounds, where `MIS` is the complexity of finding an MIS on a worst-case graph and and `R32` is the complexity of finding a $(3, 2)$-ruling set on a worst-case graph, as claimed. ◀

3.3 Isolated Components

What remains now is to handle components that are isolated and have diameter at most 2. When we address these components, we will also address all of their ϵ-neighbors. Hence, from this point onwards we will slightly abuse our terminology, and when we refer to such a component we refer to its nodes along with their ϵ-neighbors as the component. This means that now the components that we address might have a diameter that is increased by 1, and thus their diameter can be also 3. Note that the diameter cannot be increased by two as all α-nodes are connected to all β-nodes, and each ϵ-node is connected to an α-node and to a β-node of this component, otherwise the component would not be isolated.

By definition of isolated components, the neighborhood of an ϵ-nodes within such a component, besides containing vertices of the component itself, is only composed of other ϵ-nodes. Moreover, there is at least one additional ϵ-node in this neighborhood, as we consider graphs of diameter at least 4. We distinguish two kinds of isolated components, according to whether their diameter is at most 2, or whether it is 3.

For a component V_i of diameter ≤ 2, suppose u is an ϵ-node that is a neighbor of V_i. This node u has an α-node and a β-node in its neighborhood, that both cover the entire component. Therefore, to update such components, it suffices to make sure that a non-ϵ neighbor of u is in the current independent set during the two reconfiguration steps. By considering connected two of those components that cover each other, we can take an MIS M over those. The schedule of length 4 is: update M, and then update the other components.

Assume now that V_i is a component of diameter 3. It holds that there exists an ϵ-node u, an α-node a and a β-node b such that $(u, a) \notin E$ and $(u, b) \notin E$ (otherwise the diameter would be 2). Here is an informal description of a schedule of 6-steps for this component.

1. Remove $N_\alpha(u)$. The node a stays in the independent set and covers the entire component.
2. Add u in the set.
3. Remove the remaining α-nodes of the component. The node u covers everything.
4. Add b to the set. Note that b covers the component.
5. Remove u.
6. Add the remaining β-nodes of the component.

A caveat is encountered in case there are two such components, V_1 and V_2, whose selected ϵ-nodes, u_1 and u_2, are connected. In such case we cannot do the above 6-step schedule in parallel without violating independence. However, observe that if a single of those two ϵ-nodes is added to the set, it actually covers the second component as well, as it has a diameter of 3. As a consequence, taking an MIS over those ϵ-nodes gives us a selection of nodes that cover all the considered components. Hence, consider the schedule above as being for component V_1 and denote $u = u_1$, then we can add the following to steps 3 and 4 above:

3. Remove the remaining α-nodes of V_1 and all α-nodes of V_2. The node u covers everything.

4. Add b and the β-nodes of V_2 to the set. Note that V_2 is updated and b covers V_1.

We now formalize the above intuition in order to prove the following.[1]

▶ **Lemma 9.** *Let P be the property of (V, E, U) that says that U is a (2,4)-ruling set, and let $G = (V, E, \alpha)$ and $G = (V, E, \beta)$ be an input to the MIS-reconfiguration problem such that $\alpha \cap \beta = \emptyset$, and all connected components of $\alpha \cup \beta$ are isolated and have diameter at most 2. Then, there exists an (α, β, P)-reconfiguration schedule of length 10. Moreover, such a schedule can be found in $O(\texttt{MIS})$ rounds.*

3.4 Completing the proof

We can now wrap-up all the ingredients and prove Theorem 2.

Proof of Theorem 2. We describe the full (α, β, P)-reconfiguration schedule S. First, each node v in $V_{\alpha,\beta} = \alpha \cap \beta$ sends a message to its neighbors in $N(v)$ and outputs that it is in S_i for all $0 \leq i \leq 28$. Each node that received such a message, sends a message to its neighbors and outputs that it is not in S_i for all $0 \leq i \leq 28$. The nodes that produced an output terminate and any edges incident to them are removed from the graph.

Next, all nodes collect their 4-hop neighborhood to decide whether they are in a component of diameter ≥ 3 or not, and if not then whether they are in an isolated component.

The components of diameter ≥ 3 and the non-isolated components compute the reconfiguration schedule of 18 steps, as given in Lemma 8, which we denote by $S'_0, \ldots, S'_{18}$. The isolated components of diameter ≤ 3 compute the reconfiguration schedule of 10 steps, as given in Lemma 9, which we denote by $S''_0, \ldots, S''_{10}$.

Formally, the (α, β, P)-reconfiguration schedule is now $S_i = S''_0 \cup S'_i \cup V_{\alpha,\beta}$ for $0 \leq i \leq 18$ and $S_i = S''_{i-18} \cup S'_{18} \cup V_{\alpha,\beta}$ for $18 \leq i \leq 28$. It is computed within $O(\texttt{MIS} + \texttt{R32})$ rounds. ◀

4 MIS reconfiguration in a constant number of rounds

▶ **Theorem 4** (formal). *Let P be the property of (V, E, U) that says that U is a (2,4)-ruling set. For any graph $G = (V, E)$ and any input $G_{input} = (V, E, \alpha), G_{output} = (V, E, \beta)$ to the MIS-reconfiguration problem, there exists an (α, β, P)-reconfiguration schedule of length $\Theta(f(n))$, where $f(n)$ is the largest identifier among the nodes in the graph, which can be found in $O(1)$ rounds.*

To prove this, we first prove the following lemma, stating that we can always reconfigure locally an independent set to add elements from β without losing any element in $\alpha \cap \beta$.

[1] You can find all the missing proofs in the full version of the paper [7].

▶ **Lemma 10.** *Let P be the property of (V, E, U) that says that U is a (2,4)-ruling set. For any graph $G = (V, E)$ of diameter greater than 5, two MIS α and β and $v \in \beta \setminus \alpha$, there exists an MIS γ such that*

1. *$(\alpha \cap \beta) \subset \gamma$ and $v \in \gamma$, and*
2. *there exists an (α, γ, P)-reconfiguration schedule of length 6. Moreover, for finding the reconfiguration schedule the nodes only need to know the topology of their 5-hop neighborhood and therefore can be found in $O(1)$ rounds.*

Lemma 10, means that for any element v in β, we can add v to the current MIS in a constant number of steps without losing any element of β already in the MIS. It allows us to prove Theorem 4 as follows.

Proof of Theorem 4. Nodes use their identifiers to know when to start their own reconfiguration. A node with identifier k uses slots $[6k + 1, 6(k + 1)]$ for its schedule. Since a node only needs to know its 5-hop neighborhood, this completes in $O(1)$ rounds. ◀

If the identifiers are guaranteed to be $\{1, \ldots, n\}$ then Theorem 4 gives that a constant number of rounds is sufficient for a linear length schedule. However, we can do even better by using coloring algorithms, as stated in the following corollary.

▶ **Corollary 11.** *Let P be the property of (V, E, U) that says that U is a (2,4)-ruling set. For any graph $G = (V, E)$ and any input $G_{input} = (V, E, \alpha), G_{output} = (V, E, \beta)$ to the MIS-reconfiguration problem, if the nodes are given a k-coloring of G^{10}, then there exists an (α, β, P)-reconfiguration schedule of length $O(k)$, which can be found in $O(1)$ rounds.*

5 A complete characterization for the existence of a reconfiguration schedule with 4-domination

The following gives an exact characterization of inputs for which there exists a reconfiguration schedule with 4-domination. In what follows, we say that two sets of nodes U_1 and U_2 are *fully connected* if every node in U_1 is a neighbor of every node in U_2. If U_1 contains only a single node, then we simply say that this node is fully connected to U_2.

▶ **Theorem 12.** *Let P be the property of (V, E, U) that says that U is a (2,4)-ruling set. For any input $G_{input} = (V, E, \alpha), G_{output} = (V, E, \beta)$ to the MIS-reconfiguration problem, there does not exists an (α, β, P)-reconfiguration schedule if and only if:*

1. *The sets α and β are fully connected.*
2. *Let ϵ_α (resp. ϵ_β) be the set of ϵ-nodes that are fully connected to α (resp. β). Then all the ϵ-nodes are in $\epsilon_\alpha \cup \epsilon_\beta$.*
3. *Let $G' = (V' = \epsilon_\alpha \cup \epsilon_\beta, E' = \overline{E_{V'}})$, where $\overline{E_{V'}}$ is the complementary of E restricted to vertices of V'. Then there is no path from $\epsilon_\alpha \setminus \epsilon_\beta$ to $\epsilon_\beta \setminus \epsilon_\alpha$ in G'.*

6 Impossibility results for MIS reconfiguration

We show here two types of impossibility results. One is the necessity of 4-domination in the sense that 3-domination cannot be obtained, and the other is the necessity of $\Omega(\log^* n)$ rounds with 4-domination, even on bounded degree graphs where it matches the complexity we provide in Corollary 6.

Impossibility of MIS reconfiguration with 3-domination.

▶ **Theorem 1.** *Requiring 3-domination for intermediate steps is costly:*

1. *There exists a class of inputs $G = (V, E)$ with two MIS α and β such that there is no reconfiguration schedule with 3-dominating intermediate steps.*
2. *There exists a class of inputs $G = (V, E)$ with two MIS α and β such that any reconfiguration schedule is of length $\Omega(n)$ and needs $\Theta(n)$ rounds to be found, if intermediate steps must be 3-dominating.*

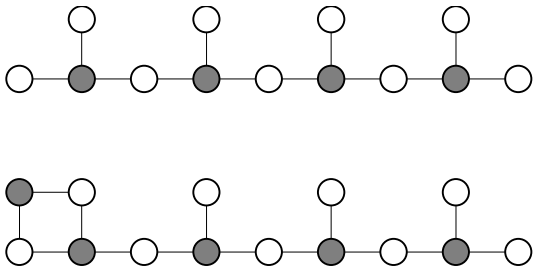


Figure 1 White nodes are α-nodes, and grey nodes are β-nodes. For the graph on the top, there is no schedule with 3-dominating sets. For the graph on the bottom, any schedule with 3-dominating sets must be of linear length and requires a linear number of rounds to be found.

An $\Omega(\log^* n)$ lower bound for MIS reconfiguration with 4-domination.

▶ **Theorem 3.** *For any fixed k, there exists a class of k-regular inputs $G = (V, E)$ with two MIS α and β such that any reconfiguration schedule of constant length with 4-domination needs $\Theta(\log^* n)$ rounds to be found.*

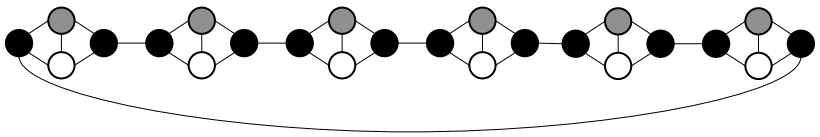


Figure 2 White nodes are α-nodes, grey nodes are β-nodes, and black nodes are ϵ-nodes. Here, $\Omega(\log^* n)$ rounds are needed with 4-domination.

7 Discussion and Open Questions

This paper defines relevant constraints for finding a reconfiguration schedule of maximal independent sets in a distributed setting. For constant-length schedules in bounded-degree graphs we completely settle the required complexity, as we provide an algorithm completing in $\Theta(\log^* n)$ communication rounds, and prove that no lower complexity exists. A main open question that remains is: Can a better complexity be found for general graphs?

Our definition only uses addition and removal of elements to the intermediate independent sets. We propose the following question: Can an efficient distributed reconfiguration schedule be found if the system allows that intermediate steps are 3-dominating and the transitions used can be any combination of addition, removal and Token Sliding?

Finally, we used as a hypothesis that the given independent sets are maximal. Our algorithm still works when the sets are not maximal, as it suffices to complete those. For example, if we are given (2,4)-ruling sets (which is equivalent to the 4-domination condition of P), the problem is solved. An interesting question could be to generalize for other (a, b)-ruling sets. What relation between a and b is needed to ensure that a schedule exists, and that it can be found efficiently with a distributed algorithm?

References

1. Noga Alon, László Babai, and Alon Itai. A Fast and Simple Randomized Parallel Algorithm for the Maximal Independent Set Problem. *J. Algorithms*, 7(4):567–583, 1986. doi:10.1016/0196-6774(86)90019-2.
2. Baruch Awerbuch, Andrew V. Goldberg, Michael Luby, and Serge A. Plotkin. Network Decomposition and Locality in Distributed Computation. In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 364–369, 1989. doi:10.1109/SFCS.1989.63504.
3. Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. Lower bounds for maximal matchings and maximal independent sets. *arXiv preprint*, 2019. arXiv:1901.02441.
4. Leonid Barenboim, Michael Elkin, and Uri Goldenberg. Locally-Iterative Distributed ($\Delta+1$): -Coloring below Szegedy-Vishwanathan Barrier, and Applications to Self-Stabilization and to Restricted-Bandwidth Models. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 437–446, 2018. doi:10.1145/3212734.3212769.
5. Leonid Barenboim, Michael Elkin, and Fabian Kuhn. Distributed (Delta+1)-Coloring in Linear (in Delta) Time. *SIAM J. Comput.*, 43(1):72–95, 2014. doi:10.1137/12088848X.
6. Marthe Bonamy, Paul Ouvrard, Mikaël Rabie, Jukka Suomela, and Jara Uitto. Distributed Recoloring. In Ulrich Schmid and Josef Widder, editors, *32nd International Symposium on Distributed Computing (DISC 2018)*, volume 121 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:17, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.DISC.2018.12.
7. Keren Censor-Hillel and Mikaël Rabie. Distributed Reconfiguration of Maximal Independent Sets. *arXiv preprint*, 2018. arXiv:1810.02106.
8. Erik D Demaine, Martin L Demaine, Eli Fox-Epstein, Duc A Hoang, Takehiro Ito, Hirotaka Ono, Yota Otachi, Ryuhei Uehara, and Takeshi Yamada. Linear-time algorithm for sliding tokens on trees. *Theoretical Computer Science*, 600:132–142, 2015.
9. Mohsen Ghaffari. An Improved Distributed Algorithm for Maximal Independent Set. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 270–277, 2016. doi:10.1137/1.9781611974331.ch20.
10. Robert A Hearn and Erik D Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *arXiv preprint*, 2002. arXiv:cs/0205005.
11. Takehiro Ito, Erik D Demaine, Nicholas JA Harvey, Christos H Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12-14):1054–1065, 2011.
12. Marcin Kamiński, Paul Medvedev, and Martin Milanič. Complexity of independent set reconfigurability problems. *Theoretical computer science*, 439:9–15, 2012.
13. Kishore Kothapalli and Sriram V. Pemmaraju. Super-Fast 3-Ruling Sets. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India*, pages 136–147, 2012. doi:10.4230/LIPIcs.FSTTCS.2012.136.

14 Fabian Kuhn, Yannic Maus, and Simon Weidner. Deterministic Distributed Ruling Sets of Line Graphs. *CoRR*, abs/1805.07209, 2018. arXiv:1805.07209.

15 Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local Computation: Lower and Upper Bounds. *J. ACM*, 63(2):17:1–17:44, 2016. doi:10.1145/2742012.

16 Nathan Linial. Distributive Graph Algorithms Global Solutions from Local Data. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, SFCS '87, pages 331–335, Washington, DC, USA, 1987. IEEE Computer Society. doi:10.1109/SFCS.1987.20.

17 Nathan Linial. Locality in Distributed Graph Algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992. doi:10.1137/0221015.

18 Daniel Lokshtanov and Amer E Mouawad. The complexity of independent set reconfiguration on bipartite graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 185–195. SIAM, 2018.

19 Michael Luby. A Simple Parallel Algorithm for the Maximal Independent Set Problem. *SIAM J. Comput.*, 15(4):1036–1053, 1986. doi:10.1137/0215074.

20 Shreyas Pai, Gopal Pandurangan, Sriram V. Pemmaraju, Talal Riaz, and Peter Robinson. Symmetry Breaking in the Congest Model: Time- and Message-Efficient Algorithms for Ruling Sets. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, pages 38:1–38:16, 2017. doi:10.4230/LIPIcs.DISC.2017.38.

21 Alessandro Panconesi and Aravind Srinivasan. On the Complexity of Distributed Network Decomposition. *J. Algorithms*, 20(2):356–374, 1996. doi:10.1006/jagm.1996.0017.

22 Johannes Schneider, Michael Elkin, and Roger Wattenhofer. Symmetry breaking depending on the chromatic number or the neighborhood growth. *Theor. Comput. Sci.*, 509:40–50, 2013. doi:10.1016/j.tcs.2012.09.004.

23 Jan van den Heuvel. The complexity of change. *Surveys in Combinatorics*, 409(2013):127–160, 2013.

Stochastic Graph Exploration*

Aris Anagnostopoulos
Sapienza University of Rome, Italy
aris@diag.uniroma1.it

Ilan R. Cohen
CWI, Amsterdam, The Netherlands
ilanrcohen@gmail.com

Stefano Leonardi
Sapienza University of Rome, Italy
leonardi@diag.uniroma1.it

Jakub Łącki
Google Research, New York, USA
jlacki@google.com

Abstract

Exploring large-scale networks is a time consuming and expensive task which is usually operated in a complex and uncertain environment. A crucial aspect of network exploration is the development of suitable strategies that decide which nodes and edges to probe at each stage of the process.

To model this process, we introduce the *stochastic graph exploration problem*. The input is an undirected graph $G = (V, E)$ with a source vertex s, stochastic edge costs drawn from a distribution $\pi_e, e \in E$, and rewards on vertices of maximum value R. The goal is to find a set F of edges of total cost at most B such that the subgraph of G induced by F is connected, contains s, and maximizes the total reward. This problem generalizes the stochastic knapsack problem and other stochastic probing problems recently studied.

Our focus is on the development of efficient nonadaptive strategies that are competitive against the optimal adaptive strategy. A major challenge is the fact that the problem has an $\Omega(n)$ adaptivity gap even on a tree of n vertices. This is in sharp contrast with $O(1)$ adaptivity gap of the stochastic knapsack problem, which is a special case of our problem. We circumvent this negative result by showing that $O(\log nR)$ resource augmentation suffices to obtain $O(1)$ approximation on trees and $O(\log nR)$ approximation on general graphs. To achieve this result, we reduce stochastic graph exploration to a memoryless process – the *minesweeper* problem – which assigns to every edge a probability that the process terminates when the edge is probed. For this problem, interesting in its own, we present an optimal polynomial time algorithm on trees and an $O(\log nR)$ approximation for general graphs.

We study also the problem in which the maximum cost of an edge is a logarithmic fraction of the budget. We show that under this condition, there exist polynomial-time oblivious strategies that use $1 + \epsilon$ budget, whose adaptivity gaps on trees and general graphs are $1 + \epsilon$ and $8 + \epsilon$, respectively. Finally, we provide additional results on the structure and the complexity of nonadaptive and adaptive strategies.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Stochastic approximation

Keywords and phrases stochastic optimization, graph exploration, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.136

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

* Partially done while the authors were visiting the Algorithms and Uncertainty program at the Simons Institute for the Theory of Computing, Berkeley.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 136; pp. 136:1–136:14

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

Funding *Stefano Leonardi*: Partially supported by ERC Advanced Grant 788893 AMDROMA "Algorithmic and Mechanism Design Research in Online Markets".

Acknowledgements The authors would like to thank Yossi Azar, Anupam Gupta, Peter Auer and C. Seshadhri for fruitful discussions.

1 Introduction

Exploring networked data is a time consuming and expensive task which is also subject to several limitations. For example, social networks can be explored only through the use of specific APIs made available by the provider which restrict the number of nodes that can be probed and limit the number of neighbors of each node that can be discovered with one probe. The cost and the difficulty of exploring large-scale networks can be an obstacle to collecting suitable snapshots for the purpose of testing new network analysis tools. The testing is more often executed on static networks made available in public repositories [18, 17] collected in the past for other purposes. It is therefore of crucial importance the development of effective and efficient methods to explore large-scale networks.

The core of a network exploration method is the definition of a probing strategy that decides which nodes or edges to probe at each stage of the process. Both the edge-probe and the node-probe models are useful in this setting. In the case of the exploration of social networks, a node-probing strategy allows to gain knowledge on a subset of the neighbors of the probed node. In the case of the exploration of the Twitter graph, an edge-probing strategy allows to gain information on those tweets of a user that are retweeted from his followers.

One main difficulty in the definition of an effective probing strategy is the intrinsic uncertain nature in terms of cost and probability of success of the process of discovering links in a network, especially if these links represent complex relationships between nodes. In order to confirm the existence of a link between two nodes, it may be required to execute several experiments whose outcome cannot be predicted in advance. Examples are the in-vitro reactions between proteins needed to discover protein-to-protein interaction networks [22, 6] or the influence between humans in social networks.

The second main difficulty stems from the adaptive nature of the optimal probing strategy that needs to be updated from time to time while new parts of the network are discovered. Adaptive strategies are computationally expensive, given that they must be continuously updated. In the case of large network exploration, the communication cost of adaptive strategies is also high since many machines are usually working in parallel at the exploration process, and the updated strategy must be communicated to the machines participating in the process. We are therefore interested in devising *nonadaptive probing strategies* that are simple and that define the sequence of probes in advance before the process is started. The obvious drawback is that nonadaptive probing strategies may be suboptimal.

Several recent works [20, 21, 16] have focused on the task of exploring real-world networks when a limited budget is available. However, these papers do not provide a comprehensive theoretical study of these problems. In this work we initiated the study of exploring an undirected network from a root node. The graph has costs on the edges and rewards on the vertices. A budget limits the total cost of the of the graph edges that are probed.

More formally, the input of the *stochastic graph exploration problem* is an undirected graph $G = (V, E)$ with a source vertex $s \in V$, *stochastic* edge costs $C : E \to \mathbb{R}_{\geq 0}$ distributed according to π_e, $e \in E$, and deterministic rewards of vertices $w : V \to \mathbb{R}_{\geq 0}$. (The model can be easily extended to rewards distributed according to independent random variables.) During the graph-exploration process we construct a set of edges $F \subseteq E$ that we probe and

we traverse. All vertices of the subgraph of G spanned by F must be connected to s via the edges of F. We probe edges one by one and we add them to F. The actual cost of an edge e, drawn from the distribution π_e, is revealed only when the edge is traversed. The goal is to maximize the total reward from the vertices spanned by the edge set F while the total cost of the edges in F remains bounded by a prespecified budget B. As soon as we probe an edge such that the total cost exceeds B the process terminates.

In the stochastic graph exploration problem, we aim to design simple polynomial-time computable *nonadaptive strategies* with a reward as close as possible to the reward obtained by the optimal *adaptive strategy*, which decides on the next edge to be traversed after the cost of all previously traversed edges is revealed (see Section 2 for precise definitions). This is customary in a class of stochastic optimization problems [4], for which it is common to bound the *adaptivity gap* of the nonadaptive strategy.

The stochastic graph exploration problem generalizes some important stochastic optimization problems. If the graph G is a star graph, our problem models *exactly* the stochastic knapsack problem [8, 4]. Stochastic knapsack admits an $O(1)$ adaptivity gap, that is, there exists an optimal nonadaptive strategy, which approximates the optimal adaptive strategy up to a constant factor. The nonadaptive strategy is devised by exploiting a suitable LP relaxation for the problem because the standard formulation has an unbounded integrality gap defined as the worst-case ratio between the optimal integral cost and optimal fractional cost of the LP. In the LP version of the problem that is used, the costs of the edges are reduced to their truncated (by the maximum budget) expected costs and the rewards are also reduced by the probability that the cost of the item is below the maximum budget.

If the network we need to explore is a tree, the stochastic graph problem is a stochastic knapsack problem with precedence constraints: only a subset of items are available in the beginning and adding each item to the knapsack will make some new items – the direct descendants of the explored node – available. Unfortunately, as opposed to the knapsack problem, the adaptivity gap of the stochastic graph exploration problem that we consider is unbounded even on a tree network and therefore the LP-based approach of stochastic knapsack cannot directly be extended.

The stochastic graph exploration problem also models stochastic graph probing problems. Probing problems in graphs have been introduced [7, 14] because of their applications to kidney exchange and online dating. Consider a probing probability for each edge $p : E \to [0, 1]$, that is, edge e will materialize with probability $p(e)$ each time is probed, independently of the other edges and of the previous probes. The goal is to maximize the number of vertices that are connected to a source vertex s by the set F of edges that have been successfully probed when the total number of probes is limited by B. Nonadaptive strategies probe a list of edges in a sequence till success or the total budget B is reached. The stochastic graph exploration problem we study models the stochastic graph probing problem by setting the costs of the edges distributed according to $\mathbf{Pr}(C_e = i) = (1 - p(e))^{i-1}p(e)$, with i being the number of probes needed to discover edge e.

1.1 Summary of Our Results

Our main contribution is the definition of the stochastic graph exploration problem and the study of the adaptivity gap of nonadaptive probing strategies. Here is a summary of our results:

Our first result is an $\Omega(n)$ adaptivity gap for the stochastic graph exploration problem even on a spider graph, which is a tree containing a single node of degree more than two. (Observe that the problem for a simple path is easy because the optimal strategy will traverse sequentially the edges of the path starting from the root.)

One first direction we pursue to circumvent the impossibility result is to allow a limited amount of resource augmentation: instead of using budget B, we allow the algorithm to use a budget of $\beta \cdot B$, for some value of β. We call an algorithm (α, β)-approximate if it computes a strategy which uses budget $\beta \cdot B$, and obtains an expected reward of at least $1/\alpha$ times the optimal reward (obtained by an adaptive algorithm). We present polynomial time computable nonadaptive strategies in a graph of n vertices that are $(O(1), O(\log nR))$-approximate for trees and $(O(\log nR), O(\log nR))$-approximate for general graphs, with R being the maximum reward of a vertex.

The idea is to transform the stochastic exploration problem into a memoryless stochastic process, which we call the *minesweeper problem*, and which may be of independent interest. In the minesweeper problem, the budget and the edge costs are replaced by probabilities $p(e)$, which are specified for every edge e. When an edge e is probed, the process stops with probability $1 - p(e)$. Hence, the final reward of a vertex is discounted by the probability that the strategy does not stop before the vertex is acquired. The minesweeper problem is, in fact, a special case of stochastic graph exploration, where the support of each π_e (distribution of cost of edge e) is $\{0, B + 1\}$ and the budget is B.

We prove that an α-approximate strategy for the minesweeper problem implies an $(O(\alpha), O(\log nR))$-approximate nonadaptive strategy for the stochastic graph exploration problem. The idea of the reduction is as follows. We construct a minesweeper problem instance, where $p(e) = \mathbf{Pr}(\pi_e < X_B)$, where X_B is random variable that follows an exponential distribution with parameter B. We first show that, for any subset of edges F, the probability that their total cost in the stochastic graph exploration is at most B is at most a constant factor of the probability that minesweeper would stop on this set. On the other hand, the expected additional reward that can be achieved from minesweeper after the total cost becomes larger than $O(B \log nR)$ is negligible.

We then show how to compute in polynomial time an optimal strategy for the minesweeper problem on trees and an $O(\log nR)$-approximate strategy on general graphs. These results imply imply an $(O(1), O(\log nR))$-approximate strategy for trees and an $(O(\log nR), O(\log nR))$-approximate strategy for general graphs. To show the optimal result on trees we prove two facts. First, the order of traversal of the edges in each subtree can be determined independently. Second, we show a simple optimality condition which helps us determine how many edges from each subtree should be probed before switching to a different subtree. We remark that our approach is in a spirit similar to the greedy optimal strategy defined by the Gittins index [10, 9] for multi-armed bandit problems. However, differently from the standard setting of the Gittins index, in the minesweeper problem, a whole new set of arms is made available for each node of the tree reached by the exploration process. Moreover, in the minesweeper problem, the discount factor is not constant because it depends on the probability assigned to the traversed edge. This approach is not viable for general graphs, and we provide an approximate solution instead, by showing a reduction of minesweeper to max-prize problem [5].

We also pursue a second direction to circumvent the lower bound on the adaptivity gap for trees: we restrict the distributions by considering the case when the edge costs are bounded by $\frac{\epsilon^2 B}{c \log n}$ for a suitable constant c. We show, under this condition, the existence of a polynomial time computable $(1 + \epsilon, 1 + \epsilon)$-approximate nonadaptive strategy for trees and $(1 + \epsilon, 8 + \epsilon)$-approximate nonadaptive strategy for any graph G. We note that this approach can be extended to prove a result with resource augmentation similar to the one we obtained through reduction to the minesweeper problem. Yet, we believe that both the minesweeper problem and the reduction technique can be of independent interest.

Our final result is related to the problem of finding a nonadaptive probing strategy that is $(o(n), O(1))$-approximate. We leave open this challenging problem even for trees. However, we establish an interesting result for the characterization of nonadaptive strategies. We prove that any nonadaptive strategy that probes edges in order until it succeeds or until the budget is exceeded can be $(O(1), O(1))$-approximated by a set strategy, which probes all edges at once and obtains a reward only if all edges of a set are successfully probed within budget. We specifically prove that the adaptivity gap of a nonadaptive strategy can be approximated up to a factor of 6 by a set strategy that uses budget $9B$. We use this result to give an algorithm for finding a strategy for trees, which is $(O(1), O(1))$-approximate, compared to the best *nonadaptive* strategy. Surprisingly, the resulting strategy is adaptive.

1.2 Related Work

The adaptivity gap of stochastic problems has been studied for the knapsack problem [8, 4] which is a special case of the problem we study. The adaptivity gap has also been studied for budgeted multi-armed bandits [19, 12, 11] by resorting to suitable linear programming relaxation. Differently from previous work on budgeted multi-armed bandit problems, we consider the setting in which new arms appear after some arms are pulled. Stochastic probing problems have also been studied for matching [1, 7, 2] motivated from kidney exchange and for more general classes of matroid optimization problems [14, 15].

The stochastic graph exploration problem we introduce is also related to the *stochastic orienteering* problem [3, 13]. In stochastic orienteering, the set of traversed edges must form a path in a metric graph with deterministic costs on the edges, while the time spent on a node is a random variable, which follows an a-priori known distribution. In stochastic graph exploration, the random variables are the costs of the edges of the graph but we cannot ensure that the costs on the edges form a metric since the random variables are independent.

1.3 Organization of the Paper

In Section 2 we formally define our problems. In Section 3 we show the lower bounds on the adaptivity gap for stochastic graph exploration. In Section 4 we show our reduction to the minesweeper problem and our results for stochastic graph exploration with resource augmentation. In Section 5 we present a near-optimal set strategy for trees. In Section 6 we present our results for the case of edges of small costs and, finally, in Section 7 we study the power of resource augmentation for relating the cost of nonadaptive strategies to the cost of optimal set strategies.

2 Problem Definition

We start by an auxiliary definition. Let $G = (V, E)$, with $|V| = n$, be an undirected graph and $s \in V$. We say that a set $F \subseteq E$ is *connected to* s if F induces a connected subgraph of G and s is the endpoint of at least one $e \in F$.

Let us now define the StochasticExploration problem (in the following sometimes abbreviated by SGE). This problem instance is given by a tuple (G, s, C, w), where G is an undirected graph $G = (V, E)$, $s \in V$ is a source vertex, C is a function that assigns *stochastic* edge costs to each edge, and $w : V \to \mathbb{R}_{\geq 0}$ is a function that assigns (deterministic) reward to each vertex.[1] And we denote R as the maximum reward of a vertex i.e. $R = \max_{v \in V} w(v)$.

[1] The results hold also if the rewards are random variables that are independent of each other and the

Formally, for each $e \in E$, $C(e)$ is a random variable distributed according to π_e that takes values in $\mathbb{R}_{\geq 0}$, all random variables $C(e)$ being jointly independent. For an edge (u, v) we will often denote $C(u, v) = C((u, v))$.

Consider the following single-player game. The player has an initial budget of B ($B = 1$ if not specified) and maintains an initially empty subset F of E, which we call the set of *acquired edges*. In each step the player can choose an edge $e \in E \setminus F$ and *probe* it (if $F = E$, the game finishes). Probing an edge e is only allowed when $F \cup \{e\}$ is connected to s. When e is probed, the actual cost $C(e)$ of e, drawn from the distribution π_e, is revealed. If the cost e is not greater than the remaining budget, e is *acquired* (added to F) and $C(e)$ is subtracted from the budget. If $C(e)$ exceeds the remaining budget, the game finishes. The goal of the player is to maximize the final *payoff* of F, which is the total reward of all vertices in the subgraph of G induced by F.

Let us now define the MineSweeper problem, which we often abbreviate to MS. This problem is defined by a tuple (G, s, p, w), where G is an undirected graph, $s \in V$ is a start vertex, $p : E \to [0, 1]$ is a function that assigns to each edge e the probability that e materializes and $w : V \to \mathbb{R}_{\geq 0}$ is a function that assigns (deterministic) reward to each vertex. The only difference between MS and SGE is in how edges are probed. There are no edge costs or budget. Instead, whenever an edge e is probed, it materializes (independently of the other edges) with probability $p(e)$ and is acquired immediately. If the edge does not materialize, the process ends immediately. Note that as in SGE, probing an edge e is only allowed when $F \cup \{e\}$ is connected to s. Note that the MineSweeper problem is a special case of the StochasticExploration problem, by letting, for each edge e, π_e be the distribution in which with probability $p(e)$ we obtain the value 0 and with probability $1 - p(e)$ the value $B + 1$.

We consider the following types of strategies for both problems:

- An *adaptive* strategy is a mapping from the set of already acquired edges (and the remaining budget, in the case of SGE) to the next edge to be probed.
- A *nonadaptive* strategy, also called a *list* strategy, is described by a sequence $e_1, \ldots, e_k$ consisting of distinct elements of E, such that for each $1 \leq i \leq k$, the set $\{e_1, \ldots, e_i\}$ is connected to s. In this strategy, the edges are simply probed according to their order in the sequence.
- A *set* strategy is a nonadaptive strategy with the additional restriction that it does not obtain any payoff if it does not acquire all edges from the list.[2]

For a strategy S for SGE, we denote by $r(\mathcal{I}_{\mathrm{SGE}}, S, B)$ the expected payoff of strategy S for the SGE problem instance $\mathcal{I}_{\mathrm{SGE}} = (G, s, C, w)$ with initial budget of B, which is the expected reward of the set of nodes in the returned solution. When $B = 1$ we sometimes omit the third argument of $r(\cdot)$. Similarly, we denote by $r_{\mathrm{MS}}(\mathcal{I}_{\mathrm{MS}}, S)$ the expected payoff of strategy S for the MS problem instance $\mathcal{I}_{\mathrm{MS}}$. We call a strategy S *optimal* for $\mathcal{I}$ with budget B, if for all strategies S', $r(\mathcal{I}, S, B) \geq r(\mathcal{I}, S', B)$. Let $\mathrm{OPT}_{\mathrm{ad}}$ be the optimal adaptive strategy for the SGE problem and $\mathrm{OPT}_{\mathrm{na}}$ be the optimal nonadaptive strategy. We call a strategy S α-approximate, if for each instance $\mathcal{I}$, $r(\mathcal{I}, S) \geq 1/\alpha \cdot r(\mathcal{I}, \mathrm{OPT}_{\mathrm{ad}})$. Finally, an algorithm ALG is (α, β)-approximate if for any instance $\mathcal{I}$ it computes a α-approximate strategy by using a β factor resource augmentation, i.e. $r(\mathcal{I}, \mathrm{ALG}(\mathcal{I}), \beta \cdot B) \geq 1/\alpha \cdot r(\mathcal{I}, \mathrm{OPT}_{\mathrm{ad}}, B)$.

edge costs. It suffices to replace each reward with its expected value.

[2] Note that we abuse earlier definitions slightly for the sake of simplicity.

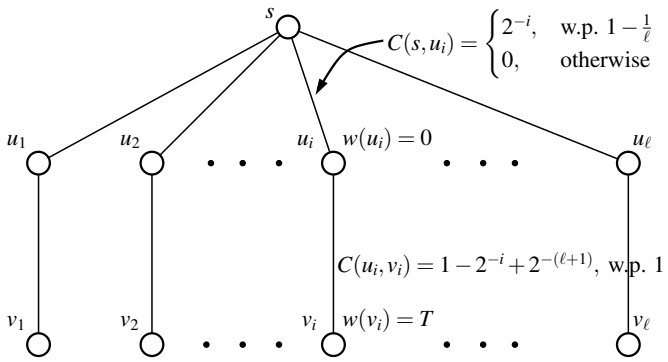


Figure 1 An instance in which the optimal adaptive strategy obtains a payoff which is $\Omega(n)$ larger than the payoff of the optimal nonadaptive strategy.

3 Lower Bounds

In this section we prove a lower bound on the *adaptivity gap* of StochasticExploration. Namely, we show an instance $\mathcal{I}_{LB} = (G, s, C, w)$ such that $r(\mathcal{I}_{LB}, \mathrm{OPT}_{\mathrm{ad}})/r(\mathcal{I}_{LB}, \mathrm{OPT}_{\mathrm{na}}) = \Omega(n)$, where $\mathrm{OPT}_{\mathrm{ad}}$ and $\mathrm{OPT}_{\mathrm{na}}$ denote the optimal adaptive and nonadaptive strategies.

The instance $\mathcal{I}_{LB}$ is shown in Figure 1. The graph G contains the set of nodes $\{s, u_1, u_2, \ldots, u_\ell, v_1, \ldots, v_\ell\}$, and the set of edges (s, u_i) and (u_i, v_i) for each $i \in [\ell]$. For each $i \in [\ell]$ we set $w(u_i) = 0$, $w(v_i) = T$, $C(s, u_i) = 2^{-i}$ with probability $1 - 1/l$ and 0 otherwise, and $C(u_i, v_i) = 1 - 2^{-i} + 2^{-(\ell+1)}$ with probability 1.

▶ **Lemma 1.** *Let OPT_{ad} and OPT_{na} denote the optimal adaptive and nonadaptive strategies for instance $\mathcal{I}_{LB}$. Then, $r(\mathcal{I}_{LB}, OPT_{ad})/r(\mathcal{I}_{LB}, OPT_{na}) = \Omega(n)$.*

One natural approach for StochasticExploration instance is to replace the stochastic edge costs with the truncated expected costs, that is, set the cost of an edge e to $\mathbb{E}[\min\{1, C(e)\}]$. However as this following example illustrates this approach does not lead to a good solution, even if constant budget augmentation is allowed.

▶ **Lemma 2.** *Let OPT_{ad} denote the optimal adaptive strategy for an instance $\mathcal{I}$ and let n be the number of vertices in the instance. Let OPT_{na} be the optimal nonadaptive strategy computed on instance $\mathcal{I}_{TR}$ obtained from $\mathcal{I}$ by setting edge costs $\mathbb{E}[\min\{1, C(e)\}]$, $e \in E$. Assume the nonadaptive algorithm is allowed to use a budget of $1 < c < n/10$. Then, there exists an instance $\mathcal{I}$ such that $r(\mathcal{I}, OPT_{ad})/r(\mathcal{I}_{TR}, OPT_{na}) = \Omega(n/2^{2c})$.*

4 The General Case and the Minesweeper Problem

In this section we describe algorithms for solving StochasticExploration, which use logarithmic budget augmentation. We first show how to reduce an instance of SGE to MineSweeper and then present solutions for MineSweeper on trees and general graphs. During the description of the reduction we also introduce the logarithmic budget augmentation. First, we observe that in the MineSweeper problem we do not have budget so there is no history that an algorithm may have to remember, except for the edges that it has probed (and succeeded). This implies the following:

▶ **Observation 1.** *There exists an optimal strategy for the* MineSweeper *problem that is nonadaptive.*

4.1 Reduction from StochasticExploration to MineSweeper

In this section we show how, given an instance $\mathcal{I}_{\text{SGE}} = (G, s, C, w)$ of StochasticExploration, we transform it to an instance $\mathcal{I}_{\text{MS}} = (G, s, p, w)$ of MineSweeper. The graph and the rewards remain the same; the challenge is to define the correct edge probability function $p(\cdot)$ for MS and relate it to the cost function $C(\cdot)$ of SGE. For each edge e' we transform the cost distribution $C(e')$ to the probability that the edge materializes, $p(e')$ (a scalar). Let $X_{e'}$ be a random variable distributed according to the exponential distribution with parameter 1, let $c_{e'}$ be the cost, which is distributed according to $C(e')$, and we set $p(e') = \mathbf{Pr}(X_{e'} > c_{e'})$. Next we show how this choice couples the two problems.

First, we show that for any subset of edges F the probability that their total cost in SGE is at most 1 is at most a factor e times of the probability that all the edges in F materialize, and therefore MS does not stop on this set. Let $\mathcal{E}_F$ be the event that all the edges in F materialize and $\mathcal{G}_F$ the event that $\sum_{e' \in F} c_{e'} \leq 1$. The following lemma makes use of properties of the exponential distribution.

▶ **Lemma 3.** *For any $F \subseteq E$ we have that $\mathbf{Pr}(\mathcal{G}_F) \leq e \cdot \mathbf{Pr}(\mathcal{E}_F)$.*

This lemma allows us to prove the following lemma, which gives a strategy for MS that is competitive with the optimal adaptive strategy for SGE. The idea behind the proof is to define a strategy for MS in such a way that we can couple the execution of the two strategies in the corresponding problems.

▶ **Lemma 4.** *Consider an* SGE *instance $\mathcal{I}_{\text{SGE}} = (G, s, C, w)$ and let $\mathcal{I}_{\text{MS}} = (G, s, p, w)$ be an instance for* MS *as defined previously. Let OPT_{ad} denote the optimal adaptive strategy for* SGE *and OPT_{MS} the optimal strategy for* MS*. We have that*

$$r((G, s, C, w), OPT_{ad}, 1) \leq e \cdot r_{MS}((G, s, C, w), OPT_{MS}).$$

Recall from Observation 1 that the optimal strategy for the MineSweeper problem is nonadaptive, therefore it can be specified by a list of edges that are selected sequentially until for one of them there is a failure. Let OPT_{MS} be such an optimal sequence of edges. Next we show that the sequence of edges OPT_{MS} can provide an approximate result to the StochasticExploration problem if we allow for some budget augmentation.

▶ **Lemma 5.** *Consider an* SGE *instance $\mathcal{I}_{\text{SGE}} = (G, s, C, w)$ and let $\mathcal{I}_{\text{MS}} = (G, s, p, w)$ be an instance for* MS *as defined previously. Let OPT_{MS} be the optimal sequence of edges for the* MineSweeper *instance, and let S be the (nonadaptive) strategy for* StochasticExploration *that probes the same edges, in the same order. Then we have that*

$$r((G, s, C, w), S, 2\ln(nR)) \geq r_{MS}((G, s, C, w), OPT_{MS}) - o(1),$$

where $R = \max_{v \in V} w(v)$.

Collecting the results of Lemmas 4 and 5 we obtain the following theorem.

▶ **Theorem 6.** *Consider an* SGE *instance $\mathcal{I}_{\text{SGE}} = (G, s, C, w)$ and let $\mathcal{I}_{\text{MS}} = (G, s, p, w)$ be an instance for* MS *as defined previously. Then*

$$r((G, s, C, w), OPT_{na}, 2\ln(nR)) + o(1) \geq r_{MS}((G, s, C, w), OPT_{MS}) \geq \frac{r((G, s, C, w), 1, OPT_{ad})}{e}.$$

4.2 MineSweeper on Trees

We show that the minesweeper problem on trees can be solved optimally in near-linear time.

▶ **Theorem 7.** *Consider the instance $\mathcal{I} = (T, s, p, w)$ of the minesweeper problem, where T is a tree. The optimal strategy, OPT_{MS}, for* MineSweeper *on T can be computed in $O(n \log n)$ time, where n is the number of vertices of T.*

The algorithm is surprisingly simple and based on a greedy approach. We define the utility of an edge to be the expected payoff from probing it, divided by the probability that the edge does not materialize. The algorithm is based on two observations. First, we observe that if there is a node x in the graph with a single child y and the utility of the edge xy is larger than the utility of the edge connecting x and its parent, then without loss of optimality we can assume that the edge xy is probed right after the edge connecting x and its parent, so we can merge these two edges into a single one. Second, if there is a node x, such that one can probe all edges in the subtree of x in the order of decreasing utilities (and not violate the constraint that an edge can be probed only after one of its endpoints has been acquired) then one can replace the entire subtree of x with a line, which is a subtree imposing the concrete order of probing edges. It turns out that by using both these rules one can find the optimal order of probing edges efficiently.

We obtain the algorithm by generalizing some existing results from the area of scheduling. At the same time our analysis is arguably simpler. We give the proof of Theorem 7 in the full version of the paper.

4.3 MineSweeper on general graphs

In this section we present an algorithmic solution to Minesweeper for general graphs, which provides a bicriteria approximation for our problem. We prove the following theorem.

▶ **Theorem 8.** *Consider the instance $\mathcal{I} = (G, s, p, w)$ of the minesweeper problem, where $G = (V, E)$ is an undirected graph. An $O(\log nR)$-approximate strategy can be computed in polynomial time.*

In the following we provide a sketch of the proof. Assume that the optimal solution is the sequence of edges $S^* = (e_1, \ldots, e_k)$. We first observe that the edges in S^* must form a tree. Define $\mathcal{M}(E')$ to be the event that all the edges in the set E' materialize. Also let $w(e_1, \ldots, e_i) = \sum_{j=1}^{i} w(e_i)$. Then S^* is a sequence that maximizes

$$O^* = \sum_{i=1}^{k} \mathbf{Pr}(\mathcal{M}(\{e_1, \ldots, e_i\}), \neg\mathcal{M}(\{e_{i+1}\})) \cdot w(e_1, \ldots, e_i)\,.$$

For $\ell = 0, 1, \ldots, \ln nR$, define $I(\ell)$ to be all values j such that $w(e_1, \ldots, e_j) \in [2^\ell, 2^{\ell+1} - 1]$, and $\iota(\ell)$ to be the smallest such j.

We can write after some manipulations:

$$O^* \leq \sum_{\ell=0}^{\ln nR} 2w(e_1, \ldots, e_{\iota(\ell)}) \cdot \mathbf{Pr}\big(\mathcal{M}(\{e_1, \ldots, e_{\iota(\ell)}\})\big) \leq 2\ln(nR) \cdot w(\tilde{E}) \cdot \mathbf{Pr}\big(\mathcal{M}(\tilde{E})\big)\,,$$

with $\tilde{E} \subset E$ being the set of edges that defines a tree that contains s and maximizes $w(\tilde{E}) \cdot \mathbf{Pr}\big(\mathcal{M}(\tilde{E})\big)$. Therefore, our goal becomes that of finding that set of edges $\tilde{E}$ that forms a tree and maximizes $w(\tilde{E}) \cdot \mathbf{Pr}\big(\mathcal{M}(\tilde{E})\big)$.

For this purpose, we use the problem of *max-prize tree*. In the max-prize tree [5] we are given an undirected graph $G = (V, E)$ with a source vertex $s \in V$, (deterministic) edge costs $c : E \to \mathbb{R}_{\geq 0}$, deterministic rewards on the vertices $w : V \to \mathbb{R}_{\geq 0}$, and a budget $B \in \mathbb{R}$. The objective is to build a subgraph $G' = (V', E')$ of G such that (1) G' is a tree, (2) $s \in V'$, and (3) $\sum_{e \in E'} c(e) \leq B$, that maximizes $\sum_{v \in V'} w(v)$.

We use for our approximation the 8-approximation algorithm for the max-prize–tree problem given by Blum et al. [5].

5 Approximating Set Strategy on Trees

In this section we show an algorithm for computing a strategy for trees, which is $(1, 1+\epsilon)$-approximate compared to the optimal set strategy. The strategy itself is adaptive.

▶ **Lemma 9.** *Let $\mathcal{I} = (G, s, C, w)$ be a* SGE *instance, where G is a tree. Let OPT_{set} be the optimal set strategy for $\mathcal{I}$. Then, in $O(n^4/\epsilon^2)$ time we can compute an adaptive strategy S, such that $r(\mathcal{I}, S, 1+\epsilon) \geq r(\mathcal{I}, OPT_{set}, 1)$. Moreover, if edge costs are not stochastic, that is, the support of each distribution π_e has size 1, the algorithm runs in $O(n^3/\epsilon)$ time and the resulting strategy is not adaptive.*

We briefly describe the ideas behind the algorithm. Consider the instance $\mathcal{I} = (T, s, C, w)$, where T is a tree. We root the tree at s and assume an order on the children of each node. Consider the sequence $P = \langle e_1, \ldots, e_n \rangle$ of the tree edges built with the following recursive algorithm. Given a node of T, iterate through its descendant edges (according to their order) and for each such edge output it and recur on the other endpoint. This traverses the tree in a preorder fashion. We define $\prec$ to be the linear order on the edges of T induced by this traversal. In the following, we assume that the edges are ordered according to $\prec$, for example, by a maximal element of a set of edges, we mean the edges that is largest according to $\prec$.

We say that a subset A of edges of T is *feasible*, if each edge $e \in A$ is either incident to the root of T, or the parent edge of e also belongs to A. Observe that given sufficient budget, a strategy can acquire any feasible set of edges of T. This follows from the fact that for each edge e of T, its parent comes before it in P. Our algorithm will probe some feasible set of edges according to the order $\prec$, that is, after probing an edge e it will not probe any edge f such that $f \prec e$.

The algorithm for computing our strategy is based on dynamic programming. A simple and inefficient approach is to use an exponential number of states. Namely, each state can be characterized by the set of edges acquired so far, denoted by A, and the remaining budget, which we discretize to a multiple of ϵ/n. Knowing the set A allows us to find all such edges e that $A \cup \{e\}$ is a feasible set and e comes after the maximal element of A in the order $\prec$. The key idea is that we can improve the number of states to polynomial, by taking advantage of the following property of the ordering $\prec$.

▶ **Lemma 10.** *Let A be a nonempty feasible set of edges of T and let e be the maximal edge of A. Given e (and without knowing A) we can compute the set F_e of all edges f such that $e \prec f$ and $A \cup \{f\}$ is a feasible set.*

6 Bounded Edge Costs

In this section, we deal with the special case of StochasticExploration, where the cost of each edge is bounded by $O(\frac{\epsilon^2}{\ln n})$ and the ratio between the smallest and largest reward R is polynomial in n. We prove that in this setting a $(O(1), 1+\epsilon)$ strategy for SGE can be computed in polynomial time.

▶ **Theorem 11.** *Let $\mathcal{I} = (G, s, C, w)$ be an instance of* SGE*, where $C(e) = O(\frac{\epsilon^2}{\ln n})$ (for each edge e and some $0 < \epsilon = O(1)$), $R \leq \epsilon n^{O(1)}$, and the smallest reward is 1. Then, in polynomial time, we can compute a nonadaptive $(O(1), 1+\epsilon)$-approximate strategy for $\mathcal{I}$. Additionally, if G is a tree, then in time $O(n^3/\epsilon)$ we can compute a nonadaptive $(1+\epsilon, 1+\epsilon)$-approximate strategy for $\mathcal{I}$.*

To prove the theorem, we consider the following strategy. We replace the stochastic edge costs with their expected values (i.e., the edge cost distributions in the modified instance have size 1). Then, we show that the optimal set strategy using budget augmented by a factor of $1+\epsilon$ gives a $(1+\epsilon)$-approximate solution.

For ease of notation, we scale the edge costs and the budgets by a factor of $\Theta(\epsilon^2/\ln n)$, so that the edge costs are bounded by 1 and the available budget is $B = O(\epsilon^2/\ln n)$.

First, we bound the payoff of an adaptive strategy when the expected cost of its acquired edges is more than $B \cdot (1+\epsilon)$. Let $\mu_e = \mathbf{E}[C(e)]$, and $\mu(F) = \sum_{e \in F} \mu_e$.

▶ **Lemma 12.** *Let $0 < \epsilon < 1/3$ and let $\mathcal{I} = (G, s, C, w)$ be an instance of* SGE*, in which $B \geq 5c/\epsilon^2 \cdot \ln n$. Let F be a set of edges acquired by some adaptive strategy. If $\mu(F) \geq (1+\epsilon) \cdot B$ then the probability that $C(F) \leq B$ is at most n^{-c}.*

Next, we show that if the expected cost of some set of edges is close to the budget, then this cost is highly concentrated around the expected value. This enables us to give a set strategy with small budget augmentation.

▶ **Lemma 13.** *Let $\mathcal{I} = (G, s, C, w)$ be an instance of* SGE*. For any set of edges F and any $\tilde{B} \geq 5c/\epsilon^2 \cdot \ln n$, if $\mu(F) = \tilde{B}$ then the probability that $C(F) \geq (1+\epsilon)\tilde{B}$ is at most n^{-c}.*

▶ **Lemma 14.** *Let $\mathcal{I} = (G, s, C, w)$ be an instance of* SGE*, where $B \geq 5c/\epsilon^2 \ln n$, the maximum reward R satisfies $R \leq \epsilon n^{c-1}$, and the minimum reward is 1. Let $\mathcal{I}_e$ be obtained from $\mathcal{I}$ by replacing each edge cost with its expected value. Let OPT^{ϵ}_{set} be the optimal set strategy using budget $(1+\epsilon)B$ for $\mathcal{I}_e$ and OPT_{ad} be the optimal adaptive strategy using budget B for $\mathcal{I}$. Then, $(1+\epsilon) r(\mathcal{I}, OPT^{\epsilon}_{set}, (1+\epsilon)B) \geq r(\mathcal{I}, OPT_{ad}, B)$.*

Observe that finding the optimal set strategy on $\mathcal{I}_e$ is NP-hard, as it generalizes the knapsack problem. However, it becomes tractable, if we augment the budget. In particular, for trees, we use the algorithm of Lemma 9, and for general graphs, in Section 4.3, we show how to use the solution of the max-prize problem.

7 Nonadaptive strategies

In this section we consider nonadaptive strategies for the stochastic exploration problem. The main result of this section is that, for the graph exploration problem, that there exists a *set-strategy* with a constant budget augmentation, which is a constant competitive compared to the best nonadaptive algorithm. Recall that, a *set-strategy* is to choose a set of edges (without an internal order) and to try to probe all of the edge in that set. The gain of strategy for a set of edges, is non-zero only if the *entire set* was successfully probed (i.e., if the total cost of the set is smaller than the budget), and then it collects the rewards of all the vertices connected to this set. Therefore, the expected gain of *set-strategy* given a set of edges, is the total gain of vertices spanned by these edges times the probability that the total cost of these edges would not be greater than the specified budget.

First, we are able to show how much is the increment in the probability to successfully probe a set, when using a constant budget augmentation.

7.1 Power of Budget Augmentation

Let $S = \{e_1, e_2, \ldots, e_n\}$ be a set of edges and let $c_i \triangleq C(e_i)$. Define $C_k^n = \sum_{i=k}^{n} c_i$ the realized cost of the subset of the edges $\{e_k, \ldots e_n\}$ and, for ease of notation, let $C^j = C_1^j$. For any $i \in [n]$ let $P_i(a)$ be the probability that the sum of cost of the edges $\{e_1, \ldots e_i\}$ is at most a, that is, $P_i(a) = \mathbf{Pr}\big(C^i \leq a\big)$.

The next lemma will allow us to take advantage of budget augmentation.

▶ **Lemma 15.** *Assume that for each edge e_i, $i \in [n]$ we have $c_i \in [0,1]$. Then*

$$P_n(3) \geq P_n(1)\left(1 - \ln(P_n(1))\right).$$

Interestingly, the multiplicative factor increases as the probability to succeed with the original budget decreases. We will use this fact, but to compare to a list-strategy we need stronger guarantees, we simply use the above lemma twice and deduce the following.

▶ **Corollary 16.**

$$P_n(9) \geq P_n(1)\frac{\left(1 - \ln(P_n(1))\right)^2}{2}$$

7.2 List Strategy vs. Set Strategy

Now, we are ready to prove the main claim of this section, that we are able to compare the strategies using a budget augmentation. Consider an SGE problem instance $\mathcal{I} = (G, s, C, w)$. Let $\mathrm{S}_{ls} = \langle e_1, \ldots, e_n \rangle$ be a nonadaptive strategy (a feasible sequence of edges) and let v_i denote the vertex whose reward is obtained when e_i is acquired. The expected payoff of probing the list with budget $B(\geq 1)$ is by linearity of expectation:

$$r(\mathcal{I}, \mathrm{S}_{ls}, B) = \sum_{j=1}^{n} w(v_j) \cdot \mathbf{Pr}\big(C^j \leq B\big).$$

Given a non-adaptive strategy $\mathrm{S}_{ls} = \langle e_1, \ldots, e_n \rangle$, consider n different set strategies S_k, for $k = \{1 \ldots n\}$, where $S_k = \{e_1, \ldots e_k\}$. Note that the expected payoff of S_k with budget $9 \cdot B$ is

$$r(\mathcal{I}, S_k, 9B) = \mathbf{Pr}\big(C^k \leq 9B\big) \cdot \sum_{j=1}^{k} w(v_j).$$

Finally, we show that there exists $k \in \{1, \ldots, n\}$ such that the set strategy S_k with budget $9B$ obtains a constant fraction of strategy S_{ls}.

▶ **Lemma 17.**

$$\max_k \{r(\mathcal{I}, S_k, 9B)\} \geq 0.46 \cdot r(\mathcal{I}, S_{ls}, B).$$

7.3 Algorithm for Trees

By combining Lemma 17 with the algorithm of Lemma 9, we obtain the following.

▶ **Theorem 18.** *Let $\mathcal{I} = (G, s, C, w)$ be a* SGE *instance, where G is a tree. Let OPT_{na} be the optimal nonadaptive strategy for $\mathcal{I}$. Then, in $O(n^4/\epsilon^2)$ time we can compute an adaptive strategy S, such that $r(\mathcal{I}, S, 9 + \epsilon) \geq 0.46 \cdot r(\mathcal{I}, OPT_{na}, 1)$.*

8 Conclusions

In this work we have introduced the stochastic exploration problem on graphs which generalizes the stochastic knapsack problem [8, 4]. We proved that, differently from stochastic knapsack, no $o(n)$ adaptivity gap is possible unless we allow some resource augmentation on the budget. We provided algorithms with bounded adaptivity gap and logarithmic resource augmentation by reducing stochastic exploration to a related memoryless problem – the minesweeper problem. We also considered the case of edges with small costs for which it is possible to provide an algorithm with $O(1)$ adaptivity gap and $O(1)$ resource augmentation. The most challenging problem left open from our work is the one of devising an algorithm with $O(1)$ approximation factor that uses only $O(1)$ resource augmentation for general graphs. The problem is open even for trees. We provided a set of additional results on the structure of optimal adaptive strategies and on the power of resource augmentation for set strategies with respect to list strategies that can help in addressing this problem.

References

1 Marek Adamczyk. Improved analysis of the greedy algorithm for stochastic matching. *Information Processing Letters*, 111(15):731–737, 2011. doi:10.1016/j.ipl.2011.05.007.

2 Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra. When LP Is the Cure for Your Matching Woes: Improved Bounds for Stochastic Matchings. *Algorithmica*, 63(4):733–762, August 2012.

3 Nikhil Bansal and Viswanath Nagarajan. *On the Adaptivity Gap of Stochastic Orienteering*, pages 114–125. Springer International Publishing, Cham, 2014. doi:10.1007/978-3-319-07557-0_10.

4 Anand Bhalgat, Ashish Goel, and Sanjeev Khanna. *Improved Approximation Results for Stochastic Knapsack Problems*, pages 1647–1665. SIAM, 2011. doi:10.1137/1.9781611973082.127.

5 Avrim Blum, Shuchi Chawla, David R. Karger, Terran Lane, Adam Meyerson, and Maria Minkoff. Approximation Algorithms for Orienteering and Discounted-Reward TSP. *SIAM Journal on Computing*, 37(2):653–670, 2007.

6 Michael Caldera, Pisanu Buphamalai, Felix Mueller, and Jorg Menche. Interactome-based approaches to human disease. *Current Opinion in Systems Biology*, 3:88–94, 2017.

7 Ning Chen, Nicole Immorlica, Anna R. Karlin, Mohammad Mahdian, and Atri Rudra. *Approximating Matches Made in Heaven*, pages 266–278. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. doi:10.1007/978-3-642-02927-1_23.

8 Brian C. Dean, Michel X. Goemans, and Jan Vondrak. Approximating the Stochastic Knapsack Problem: The Benefit of Adaptivity. *Mathematics of Operations Research*, 33(4):945–964, 2008. doi:10.1287/moor.1080.0330.

9 Esther Frostig and Gideon Weiss. Four proofs of gittins multiarmed bandit theorem, 1999.

10 Author(s) J. C. Gittins and J. C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society, Series B*, pages 148–177, 1979.

11 Sudipto Guha and Kamesh Munagala. Approximation Algorithms for Budgeted Learning Problems. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 104–113, New York, NY, USA, 2007. ACM. doi:10.1145/1250790.1250807.

12 Anupam Gupta, Ravishankar Krishnaswamy, Marco Molinaro, and R. Ravi. Approximation Algorithms for Correlated Knapsacks and Non-martingale Bandits. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 827–836, 2011.

13 Anupam Gupta, Ravishankar Krishnaswamy, Viswanath Nagarajan, and R. Ravi. Running Errands in Time: Approximation Algorithms for Stochastic Orienteering. *Mathematics of Operations Research*, 40(1):56–79, 2015.

14 Anupam Gupta and Viswanath Nagarajan. *A Stochastic Probing Problem with Applications*, pages 205–216. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

15 Anupam Gupta, Viswanath Nagarajan, and Sahil Singla. Algorithms and Adaptivity Gaps for Stochastic Probing. In *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 1731–1747, Philadelphia, PA, USA, 2016. Society for Industrial and Applied Mathematics. URL: `http://dl.acm.org/citation.cfm?id=2884435.2884555`.

16 R. Laishram, K. Areekijseree, and S. Soundarajan. Predicted max degree sampling: Sampling in directed networks to maximize node coverage through crawling. In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 940–945, May 2017. `doi:10.1109/INFCOMW.2017.8116502`.

17 Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

18 Jure Leskovec and Rok Sosič. SNAP: A General-Purpose Network Analysis and Graph-Mining Library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1, 2016.

19 Will Ma. *Improvements and Generalizations of Stochastic Knapsack and Multi-Armed Bandit Approximation Algorithms: Extended Abstract*, pages 1154–1163. SIAM, 2014. `doi:10.1137/1.9781611973402.85`.

20 S. Soundarajan, T. Eliassi-Rad, B. Gallagher, and A. Pinar. MaxReach: Reducing network incompleteness through node probes. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 152–157, August 2016. `doi:10.1109/ASONAM.2016.7752227`.

21 Sucheta Soundarajan, Tina Eliassi-Rad, Brian Gallagher, and Ali Pinar. *epsilon* W GXX: Adaptive Edge Probing for Enhancing Incomplete Networks. In *Proceedings of the 2017 ACM on Web Science Conference*, WebSci '17, pages 161–170, New York, NY, USA, 2017. ACM. `doi:10.1145/3091478.3091492`.

22 M. Vidal, ME Cusick, and AL Barabasi. Interactome networks and human disease. *Cell.*, 144(6):986–98, 2011.

Energy Consumption of Group Search on a Line

Jurek Czyzowicz
Université du Québec en Outaouais, Gatineau, Québec, Canada

Konstantinos Georgiou
Department of Mathematics, Ryerson University, Toronto, Ontario, Canada

Ryan Killick
School of Computer Science, Carleton University, Ottawa, Ontario, Canada

Evangelos Kranakis
School of Computer Science, Carleton University, Ottawa, Ontario, Canada

Danny Krizanc
Department of Mathematics & Comp. Sci., Wesleyan University, Middletown, CT, USA

Manuel Lafond
Department of Computer Science, Université de Sherbrooke, Sherbrooke, Québec, Canada

Lata Narayanan
Department of Comp. Sci. and Software Eng., Concordia University, Montreal, Québec, Canada

Jaroslav Opatrny
Department of Comp. Sci. and Software Eng., Concordia University, Montreal, Québec, Canada

Sunil Shende
Department of Computer Science, Rutgers University, Camden, NJ, USA

Abstract

Consider two robots that start at the origin of the infinite line in search of an *exit* at an unknown location on the line. The robots can collaborate in the search, but can only communicate if they arrive at the same location at exactly the same time, i.e. they use the so-called *face-to-face* communication model. The group search time is defined as the worst-case time as a function of d, the distance of the exit from the origin, when both robots can reach the exit. It has long been known that for a single robot traveling at unit speed, the search time is at least $9d - o(d)$; a simple doubling strategy achieves this time bound. It was shown recently in [15] that $k \geq 2$ robots traveling at unit speed also require at least $9d$ group search time.

We investigate *energy-time trade-offs* in group search by two robots, where the energy loss experienced by a robot traveling a distance x at constant speed s is given by s^2x, as motivated by energy consumption models in physics and engineering. Specifically, we consider the problem of minimizing the total energy used by the robots, under the constraints that the search time is at most a multiple c of the distance d and the speed of the robots is bounded by b. Motivation for this study is that for the case when robots must complete the search in $9d$ time with maximum speed one ($b = 1;\ c = 9$), a single robot requires at least $9d$ energy, while for two robots, all previously proposed algorithms consume at least $28d/3$ energy.

When the robots have bounded memory and can use only a constant number of fixed speeds, we generalize an algorithm described in [3, 15] to obtain a family of algorithms parametrized by pairs of b, c values that can solve the problem for the entire spectrum of these pairs for which the problem is solvable. In particular, for each such pair, we determine optimal (and in some cases nearly optimal) algorithms inducing the lowest possible energy consumption.

We also propose a novel search algorithm that simultaneously achieves search time $9d$ and consumes energy $8.42588d$. Our result shows that two robots can search on the line in optimal time $9d$ while consuming less total energy than a single robot within the same search time. Our algorithm uses robots that have unbounded memory, and a finite number of dynamically computed speeds. It can be generalized for any c, b with $cb = 9$, and consumes energy $8.42588b^2d$.

2012 ACM Subject Classification Computing methodologies → Mobile agents; Theory of computation → Online algorithms

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 137; pp. 137:1–137:15

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

Keywords and phrases Evacuation, Exit, Line, Face-to-face Communication, Robots, Search

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.137

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Related Version Full version hosted on arXiv `https://arxiv.org/abs/1904.09714`.

Funding Research supported by NSERC discovery grants, NSERC graduate scholarship, and NSF.

1 Introduction

The problem of searching for a treasure at an unknown location in a specified continuous domain was initiated over fifty years ago [6, 7]. Search domains that have been considered include the infinite line [2, 6, 7, 34], a set of rays [10, 11], the unit circle [12, 24, 37], and polygons [27, 33, 35]. Consider a robot (sometimes called a mobile agent) starting at some known location in the domain and looking for *an exit* that is located at an unknown distance d away from the start. What algorithm should the robot use to find the exit as soon as possible? The most common cost measure used for the search algorithm is the worst-case *search time*, as a function of the distance d of the exit from the starting position. For a fixed-speed robot, the search time is proportional to the length of the trajectory of the robot. Other measures such as turn cost [28] and different costs for revisiting [9] have also been studied.

We consider for the first time the *energy* consumed by the robots while executing the search algorithm. The energy used by a robot to travel a distance x at speed s is computed as s^2x and is motivated from the concept of viscous drag in fluid dynamics; see Section 2 for details on the energy model. For a single robot searching on the line, the classic *Spiral Search* algorithm (also known as the doubling strategy) has search time $9d$ and is known to be *optimal* when the robot moves with unit speed. Since in the worst case, the robot travels distance $9d$ at unit speed, the energy consumption is $9d$ as well. Clearly, as the speed of the robot increases, the time to find the exit decreases but the energy used increases. Likewise, as the speed of the robot decreases, the time to find the exit increases, while the energy consumption decreases. Thus there is a natural trade-off between the time taken by the robot to search for the exit and the energy consumed by the robot. To investigate this trade-off, we consider the problem of minimizing the total energy used by the robots to perform the search when the speed of the robot is bounded by b, and the time for the search is at most a multiple c of the distance d from the starting point to the exit.

Group search by a set of $k \geq 2$ collaborating robots has recently gained a lot of attention. In this case, the search time is the time when *all* k robots reach the exit. The problem has also been called *evacuation*, in view of the application when it is desired that all robots reach and evacuate from the exit. Two models of communication between the robots have been considered. In the wireless communication model, the robots can instantly communicate with each other at any time and over any distance. In the face-to-face communication model (F2F), two robots can communicate only when in the same place at the same time. In many search domains, and for both communication models, group search by $k \geq 2$ agents has been shown to take less time than search by a single agent; see for example [24, 27].

In this paper, we focus on group search on the line, by two robots using the F2F model. Chrobak *et al* [15] showed that group search in this setting cannot be performed in time less than $9d - o(d)$, regardless of the number of robots, assuming all robots use at most

unit speed. They also describe several strategies that achieve search time $9d$. In the first strategy, the two robots independently perform the Spiral Search algorithm, using unit speed during the entire search. Next, they consider a strategy first described in [3], that we call the *Two-Turn* strategy, whereby two robots head off independently in opposite directions at speed $1/3$; when one of them finds the exit, it moves at unit speed to chase and catch the other robot, after which they both return at unit speed to the exit. Finally, they present a new strategy, called the *Fast-Slow* algorithm in which one robot moves at unit speed, while the other robot moves at speed $1/3$, both performing a spiral search. The doubling strategy is very energy-inefficient, it uses energy $18d$ if the two robots always travel together, or $14D$ if the robots start by moving in opposite directions. The other two algorithms both use energy $28d/3 > 9d$. Interestingly, the two strategies that achieve an energy consumption of $28d/3$ with search time $9d$, both use two different and pre-computed speeds, but are quite different in terms of the robot capacities needed. In the Two-Turn strategy, the robots are extremely simple and use constant memory; they use only three states. In Fast-Slow and Spiral Search, the robots need unbounded memory, and perform computations to determine how far to go before turning and moving in the opposite direction.

Memory capability, time- and speed-bounded search, and energy consumption by a two-robot group search algorithm on the line: these considerations motivate the following questions that we address in our paper:

1. Is there a search strategy for constant-memory robots that has energy consumption $< 9d$?
2. Is there *any* search strategy that uses time $9d$ and energy $< 9d$?

1.1 Our results

We generalize the Two-Turn strategy for any values of c, b. We analyze the entire spectrum of values of c, b for which the problem admits a solution, and for each of them we provide optimal (and in some cases nearly optimal) speed choices for our robots (Theorem 5). In particular, and somewhat surprisingly, our proof makes explicit how for any fixed c the optimal speed choices do not simply "scale" with b; rather more delicate speed choices are necessary to comply with the speed and search time bounds. For the special case of $c \cdot b = 9$, our results match with the specific Two-Turn strategy described in [15]. Our results further show that no Two-Turn strategy can achieve energy consumption less than $9d$ while keeping the search time at $9d$. In fact, we conjecture that this trade-off is impossible for *any* group search strategy that uses only constant memory robots.

In the unbounded-memory model, for the special case of $c = 9$ and $b = 1$, we give a novel search algorithm that achieves energy consumption of $8.42588d$, thus answering the second question above in the affirmative. This result shows that though two robots cannot search faster than one robot on the line [15], somewhat surprisingly, two robots can search using less total energy than one robot, in the same optimal time. Our algorithm uses robots that have unbounded memory, and a finite number of dynamically computed speeds. Note that our algorithm can be generalized for any c, b with $cb = 9$, and utilizes energy $8.42588b^2d$ (Theorem 16).

1.2 Related Work

Several authors have investigated various aspects of mobile robot (agent) search, resulting in an extensive literature on the subject in theoretical computer science and mathematics (e.g., see [1, 30] for reviews). Search by constant-memory robots has been done mainly for finite-state automata (FSA) operating in discrete environments like infinite grids, their

finite-size subsets (labyrinths) and other graphs. The main concern of this research was the feasibility of search, rather than time or energy efficiency. For example, [14] showed that no FSA can explore all labyrinths, while [8] proved that one FSA using two pebbles or two FSAs, communicating according to the F2F model can explore all labyrinths. However, no collection of FSAs may explore all finite graphs communicating in the F2F model [39] or wireless model [17]. On the other hand, all graphs of size n may be explored using a robot having $O(\log n)$ memory [38].

Exploration of infinite grids is known as the ANTS problem [29], where it was shown that four collaborating FSAs in the semi-synchronous execution model and communicating according to the F2F scenario can explore an infinite grid. Recently, [13] showed that four FSAs are really needed to explore the grid (while three FSAs can explore an infinite band of the 2-dimensional grid).

Continuous environment cases have been investigated in several papers when the efficiency of the search is often represented by the time of reaching the target (e.g., see [2, 6, 7, 34]). Even in the case of continuous environment as simple as the infinite line, after the seminal papers [6, 7], various scenarios have been studied where the turn cost has been considered [28], the environment was composed of portions permitting different search speeds [26], some knowledge about the target distance was available [10] or where some other parameters are involved in the computation of the cost function [9] (e.g. when the target is moving).

The group search, sometimes interpreted as the evacuation problem has been studied first for the disc environment under the F2F [12, 18, 24, 27, 36] and wireless [18] communication scenarios and then also for other geometric environments (e.g., see [27]). Other variants of search/evacuation problems with a combinatorial flavour have been recently considered in [16, 19, 20, 31, 32]. Some papers investigated the line search problem in the presence of crash faulty [25] and Byzantine faulty agents [23]. The interested reader may also consult the recent survey [22] on selected search and evacuation topics.

The energy used by a mobile robot is usually considered as being spent solely for travelling. As a consequence, in the case of a single, constant speed robot the search time is proportional to the distance travelled and the energy used by a robot. Therefore the problems of minimization of time, distance or energy are usually equivalent for most robots' tasks. For teams of collaborating robots, the searchers often need to synchronize their walks in order to wait for information communicated by other searchers (e.g, see [12, 18, 36]), hence the time of the task and the distance travelled are different. However, the distance travelled by a robot and its energy used are still commensurable quantities.

To the best of our knowledge, energy consumption as a function of mobile robot speed which is based on laws of engineering physics (related to the *drag force*) has never been studied in the search literature before. Our present work is motivated by [15], which proves that the competitive ratio 9 is tight for group search time with two mobile agents in the F2F model when both agents have unit maximal speeds. More exactly, it follows from [15] that having more unit-speed robots cannot improve the group search time obtained by a single robot. Nevertheless, our paper shows that using more robots can improve the energy spending, while keeping the group-search time still the best possible.

Chrobak et al [15] present interesting examples of group search algorithms for two distinct speed robots communicating according to the F2F scenario. An interested reader may consult [4], where optimal group search algorithms for a pair of distinct maximal speed robots were proposed for both communication scenarios (F2F and wireless) and for any pair of robots' maximal speeds. It is interesting to note that, according to [4], for any distinct-speed robots with F2F communication, the optimal group search time is obtained only if one of the robots perform the search step not using its full speed.

Paper Organization. In Section 2 we formally define the evacuation problem EE_c^b, and proper notions of efficiency. Our algorithms and their analysis for constant-memory robots is presented in Section 3, while in Section 4 we introduce and analyze algorithms for unbounded-memory robots. Whenever we omit proofs, due to space limitations, we provide an outline of our arguments. The interested reader may consult the full version of our paper [21] for the missing details.

2 Preliminaries

Two robots start walking from the origin of an infinite (bidirectional) line in search of a hidden exit at an unknown absolute distance d from the origin. The exit is considered found only when one of the robots walks over it. An algorithm for group search by two robots specifies trajectories for both robots and terminates when both robots reach the exit. The time by which the second robot reaches the exit is referred to as the *search time* or the *evacuation time*.

Robot models. The two robots operate under the F2F communication model in which two robots can communicate only when they are in the same place at the same time. Each robot can change its speed at any time. We distinguish between *constant-memory* robots that can only travel at a constant number of hard-wired speeds, and *unbounded-memory* robots that can dynamically compute speeds and distances, and travel at any possible speed.

Energy model. A robot moving at constant speed s traversing an interval of length x is defined to use energy $s^2 \cdot x$. This model is well motivated from first principles in physics and engineering and corresponds to the energy loss experienced by an object moving through a viscous fluid [5]. In particular, an object moving with constant speed s will experience a *drag* force F_D proportional[1] to s^2. In order to maintain the speed s over a distance x the object must do work equal to the product of F_D and x resulting in a continuous energy loss proportional to the product of the object's squared speed and travel distance. For simplicity we have taken the proportionality constant to be one.

The total energy that a robot uses traveling at speeds $s_1, s_2, \ldots, s_t$, traversing intervals $x_1, x_2, \ldots, x_t$, respectively, is defined as $\sum_{i=1}^{t} s_i^2 \cdot x_i$. For group search with two robots, the *energy consumption* is defined as the sum total of the two robots' energies used until the search algorithm terminates.

For each $d > 0$ there are two possible locations for the exit to be at distance d from the origin: we will refer to either of these as input instances d for the group search problem. Our goal is to solve the following *optimized* search problem parametrized by two values, $\mathbf{b}$ and $\mathbf{c}$:

▶ **Problem $\mathrm{EE}_{\mathbf{c}}^{\mathbf{b}}$.** Design a group search algorithm for two robots in the F2F model that minimizes the energy consumption for d-instances under the constraints that the search time is no more than $\mathbf{c} \cdot \mathbf{d}$ and the robots use speeds that are at most $\mathbf{b}$. When there are no speed limits on the robots (i.e. $b = \infty$), we abbreviate EE_c^{∞} by EE_c. Note that b, c are inputs to the algorithm, but d and the exact location of the exit are not known.

[1] The constant of proportionality has (SI) units kg/m and depends, among other things, on the shape of the object and the density of the fluid through which it moves.

As it is standard in the literature on related problems, we assume that the exit is at least a known constant distance away from the origin. In this work, we pick the constant equal to 2, although our arguments can be adjusted to any other constant. It is not difficult to show that EE_c^b is well defined for each $b, c > 0$ with $bc \geq 1$, and the optimal offline solution, for instance d, is for both robots to move at speed $\frac{1}{c}$ to the exit. This offline algorithm has energy consumption $\frac{2d}{c^2}$. Consider an online algorithm for EE_c^b, which on any instance d has energy consumption at most $e(c, b, d)$. The *competitive ratio* of the algorithm is defined as $\sup_{d>0} \frac{c^2}{2d}\ e(c, b, d)$.

Due to [15], and when $b = 1$, no online algorithm (for two robots) can have evacuation time less than $9d - \epsilon$ (for any $\epsilon > 0$ and for large enough d). By scaling, using arbitrary speed limit b, we obtain the following fact.

▶ **Observation 1.** *No online F2F algorithm can solve* EE_c^b *if* $cb < 9$.

3 Solving EE_c^b with Constant-Memory Robots

In this section we propose a family of algorithms for solving EE_c^b (including $b = \infty$). The family uses an algorithm that is parametrized by three discrete speeds: s, r and k. The robots use these speeds depending on finite state control as follows:

▶ **Algorithm $\mathcal{N}_{s,r,k}$.** Robots start moving in opposite directions with speed s until the exit is found by one of them. The finder changes direction and moves at speed $r > s$ until it catches the other robot. Together the two robots return to the exit using speed k.

▶ **Lemma 2.** *Let* b, c *be such that there exist* s, r, k *for which* $\mathcal{N}_{s,r,k}$ *is feasible. Then, for instance* d *of* EE_c^b*, the induced evacuation time of* $\mathcal{N}_{s,r,k}$ *is* $d \cdot T(s, r, k)$ *and the induced energy consumption is* $2d \cdot E(s, r, k)$*, where*

$$T(s, r, k) := \frac{2(k + r)}{k(r - s)} + \frac{1}{s}, \quad E(s, r, k) := \frac{r}{r - s}\left(s^2 + r^2 + 2k^2\right)$$

We propose a systematic way in order to find optimal values for s, r, k of algorithm $\mathcal{N}_{s,r,k}$ for optimization problem EE_c^b (including $b = \infty$), whenever such values exist.

▶ **Theorem 3.** *Algorithm* $\mathcal{N}_{s,r,k}$ *gives rise to a feasible solution to problem* EE_c^b *if and only if* $bc \geq 9$*. For every such* $b, c > 0$*, the optimal choices of* $\mathcal{N}_{s,r,k}$ *can be obtained by solving Non Linear Program:*

$$\begin{aligned} &\min_{s,r,k \in \mathbb{R}} E(s, r, k) && (\mathrm{NLP}_c^b)\\ s.t.\ \ &T(s, r, k) \leq c \\ &r \geq s \\ &0 \leq s, r, k \leq b \end{aligned}$$

where functions $E(\cdot, \cdot, \cdot), T(\cdot, \cdot, \cdot)$ *are as in Lemma 2. Moreover, if* s_0, r_0, k_0 *are the optimizers to* NLP_c^b*, then the competitive ratio of* $\mathcal{N}_{s_0,r_0,k_0}$ *equals* $c^2 \cdot E(s_0, r_0, k_0)$.

The following subsections are devoted to solving NLP_c^b, effectively proving Theorem 5. First in Section 3.1 we solve the case $b = \infty$ and we use our findings to solve the case of bounded speeds b in the follow-up Section 3.2.

3.1 Optimal Choices of $\mathcal{N}_{s,r,k}$ for the Unbounded-Speed Problem

In this section we propose solutions to the unbounded-speed problem EE_c. Since EE_c is the same as $\mathrm{EE}_c^\infty$the problem is well-defined for every fixed $c > 0$. Moreover, by the proof of Theorem 3 algorithm $\mathcal{N}_{s,r,k}$ induces a feasible solution for every $c > 0$ as well, and the optimal speeds can be found by solving NLP_c^∞. Indeed, in the remainder of the section we show how to choose optimal values for s, r, k for solving EE_c with $\mathcal{N}_{s,r,k}$. Let

$$\sigma \approx 2.65976, \rho \approx 11.3414, \kappa \approx 6.63709, \tag{1}$$

whose exact values are the roots of an algebraic system and will be formally defined later. The main theorem of this section reads as follows.

▶ **Theorem 4.** *Let σ, ρ, κ as in (1). For every $c > 0$, the optimal speeds of $\mathcal{N}_{s,r,k}$ for problem EE_c are $s = \frac{\sigma}{c}$, $r = \frac{\rho}{c}$, $k = \frac{\kappa}{c}$. Moreover, the competitive ratio of the corresponding solution is independent of c and equals $\frac{\rho\left(2\kappa^2+\rho^2+\sigma^2\right)}{\rho-\sigma} \approx 292.369$.*

A high level outline of the proof of Theorem 4 is as follows. First we show that any optimal choices of the speeds of $\mathcal{N}_{s,r,k}$ must satisfy the time constraint of NLP_c^∞ tightly. Then, we show that finding optimal speeds s, r, k of $\mathcal{N}_{s,r,k}$ for the general problem EE_c reduces to problem EE_1. Finally, we obtain the optimal solution to NLP_1^∞ by standard tools of nonlinear programming (KKT conditions).

3.2 (Sub)Optimal Choices of $\mathcal{N}_{s,r,k}$ for the Bounded-Speed Problem

In this section, we show how to choose optimal values for s, r, k for solving EE_c^b with $\mathcal{N}_{s,r,k}$, for the entire spectrum of c, b values for which the problem is solvable by online algorithms.

The main result of this section is the following:

▶ **Theorem 5.** *Let $\gamma_1 \approx 9.06609$, $\gamma_2 = \rho \approx 11.3414$, and σ, ρ, κ as in (1). For every $c, b > 0$ with $cb \geq 9$, the following choices of speeds s, r, k are feasible for $\mathcal{N}_{s,r,k}$*

	$9 \leq cb \leq \gamma_1$	$\gamma_1 < cb < \gamma_2$	$cb \geq \gamma_2$
s	$\frac{-\sqrt{(bc)^2-10bc+9}+bc-3}{2c}$	$0.532412b - 0.0262661b^2c$	σ/c
r	b	b	ρ/c
k	b	$\frac{2bs}{bcs-b-cs^2-s}$	κ/c

The induced competitive ratio is given by:

$$f(x) := \begin{cases} \frac{1}{2}x\left(x\left(x-\sqrt{(x-9)(x-1)}\right)+\sqrt{(x-9)(x-1)}+3\right), & 9 \leq x \leq \gamma_1 \\ \frac{x^2\left((0.532412-0.0262661x)^2+\frac{11595.8(20.2699-1.x)^2}{(x(x-2.46798)-398.916)+2221.18)^2}+1\right)}{0.0262661x+0.467588}, & \gamma_1 < x < \gamma_2 \\ 292.369 & x \geq \gamma_2 \end{cases}$$

and the induced energy, for instances d, is $f(cb)\frac{2d}{c^2}$. Moreover, the competitive ratio depends only on the product cb.

In particular, the speeds' choices are optimal when $cb \leq \gamma_1$ and when $cb \geq \gamma_2$. When $\gamma_1 < cb < \gamma_2$, the derived competitive ratio is no more than 0.03 additively off from that induced by optimal choices of s, r, k.

▶ **Corollary 6.** *For $c = 9, b = 1$, the bounded-memory robot algorithm $\mathcal{N}_{s,r,k}$ has energy consumption $28d/3$ and competitive ratio 378.*

Theorem 5 is proven by solving NLP_c^b of Theorem 3. Speed values s, r, k, are chosen optimally when cb is either at most γ_1 or at least γ_2 (i.e. optimizers to NLP_c^b admit analytic description). The optimal speed parameters when $\gamma_1 < cb < \gamma_2$ cannot be determined analytically (they are roots of high degree polynomials). The values that appear in Theorem 5 are heuristically chosen, but interestingly induce nearly optimal competitive ratio.

The proof of Theorem 5 is given by Lemma 7 (the case $cb \leq \gamma_1$), Lemma 8 (the case $cb \geq \gamma_2$), and Lemma 9 (the case $\gamma_1 < cb < \gamma_2$). Next we state these Lemmata, and we sketch their proofs.

▶ **Lemma 7.** *For every $c \in (9/b, \gamma_1/b]$, where $\gamma_1 \approx 9.06609$, the optimizers to NLP_c^b are $k = r = b$, and $s_b = \frac{-\sqrt{(bc)^2-10bc+9}+bc-3}{2c}$. The induced competitive ratio is $f(cb)$, (see definition of $f(x)$ for $x \leq \gamma_1$ in statement of Theorem 5), and the energy consumption, for instances d, is $f(cb)\frac{2d}{c^2}$.*

For proving Lemma 7, first we recall the known optimizer for the special case $cb = 9$, and we identify the tight constraints. Requiring that the exact same inequality constraints to NLP_c^b remain tight, we ask how large can the product cb be so as to have KKT condition hold true. From the corresponding algebraic system, we obtain the answer $cb \leq \gamma_1 \approx 9.06609$.

Similarly, from Theorem 4 we know the optimizers to NLP_c^b for large enough values of cb, and the corresponding tight constraints to the NLP. Again, using KKT conditions, we show that the same constraints remain tight for the optimizers as long as $cb \geq \gamma_2 \approx 11.3414$. This way we obtain the following Lemma.

▶ **Lemma 8.** *For every $c > \rho/b \approx 11.3414/b$, the optimal speeds of $\mathcal{N}_{s,r,k}$ for EE_c^b are $s = \sigma/c,\ r = \rho/c,\ k = \kappa/c$, i.e. they are the same as for EE_c^∞. If the target is placed at distance d from the origin, then the induced energy equals $584.738\frac{d}{c^2}$. Moreover, the induced competitive ratio is 292.369, and is independent of b, c.*

The case $\gamma_1 < cb < \gamma_2$ can be solved optimally only numerically, since the best speed values are obtained by roots to a high degree polynomial. Nevertheless, the following lemma proposes a heuristic choice of speeds (that of Theorem 5) which is surprisingly close to the optimal.

▶ **Lemma 9.** *The choices of s, r, k of Theorem 5 when $\gamma_1 < cb < \gamma_2$ are feasible. Moreover, the induced competitive ratio is at most 0.03 additively off from the competitive ratio induced by the optimal choices of speeds (evaluated numerically).*

The trick in order to find "good enough" optimizers to NLP_c^b is to guess the subset of inequality constraints that remain tight when $\gamma_1 < cb < \gamma_2$. First, we observe that constraint $r \leq b$ is tight for the provable optimizers for all c, b when $cb \in [9, \gamma_1] \cup [\gamma_2, \infty)$. As the only other constraint that switches from being tight to non-tight in the same interval is $k \leq b$, we are motivated to maintain tightness for constraints $r \leq b$ and the time constraint. Still the algebraic system associated with the corresponding KKT conditions cannot be solved analytically. To bypass this difficulty, and assuming we know (optimal) speed s, we use the tight time constraint to find speed k as a function of c, b, s. From numerical calculations, we see that optimal speed s is nearly optimal in c, and so we heuristically set $s = \alpha c + \beta$. We choose α, β so as to have s satisfy optimality conditions for the boundary values $cb = \gamma_1, \gamma_2$. After we identify all parameters to our solution, we compare the value of our solution to the optimal one (obtained numerically), and we verify (using numerical calculations) that our heuristic solution is only by at most 0.03 additively off. The advantage of our analysis is that we obtain closed formulas for the speed parameters for all values of $cb \geq 9$.

4 Solving EE_c^b with Unbounded-Memory Robots

In this section we prove Theorem 16, that is we solve EE_c^b by assuming that the two robots have unbounded memory, and in particular that they can perform time and state dependent calculations and tasks. Note that, by scaling, our results hold for all b, c for which $cb = 9$. For simplicity our exposition is for the natural case $c = 9$ and $b = 1$. Also, as before, d will denote the unknown distance to the exit from the origin. Moreover, the exit is still assumed, for the purposes of performance analysis, to be at least 2 away from the origin.

Throughout the execution of our evacuation algorithm, robots can be in 3 different states (similar to the case of constant-memory robots). First, both robots start with the *Exploration State* and they remain in this until the exit is located. While in the exploration state, robots execute an elaborate exploration that requires synchronous movements in which robots, at a high level, stay in good proximity, still they expand the searched space relatively fast. Then, the exit finder enters the *Chasing State* in which the robot, depending on its distance from the origin, calculates a speed at which to move in order to catch and notify the other robot. Lastly, when the two robots meet, they both enter the *Exit State* in which both robots move toward the exit with the smallest possible speed while meeting the time constraint.

Our algorithm takes as input the values of $c = 9, b = 1$, and use a speed value $s \leq b$, that will be chosen later. When the exit finder switches its state from Exploration to Chasing, it remembers the distance d of the exit to the origin, as well as the value k of a counter that was used while in the Exploration State. When the exit finder catches the other robot, they both switch to the Exit State, and they remember their distance p from the origin, as well as the value of time t that their rendezvous was realized. The speed of their Exit State will be determined as a function of p, d, t (and hence of s, c, b as well).

4.1 A Critical Component: l-Phase Explorations

We adopt the language of [15] in order to discuss a structural property that any feasible evacuation algorithm for EE_9^1 satisfies. As a result, the purpose of this section is to provide high level intuition for our evacuation algorithm that is presented in subsequent sections.

We refer to the two robots (starting exploration from the origin) as L and R, intended to explore to the left and to the right of the origin, respectively. The robot trajectories can be drawn on the Cartesian plane where point-location $(x, -t)$ will correspond to point x on the line being visited by some robot at time t. The following Theorem (due to [15]) was originally phrased for the time-evacuation unit-speed robots' problem. We adopt the language of our problem.

▶ **Theorem 10** ([15]). *For any feasible solution to* EE_9^1*, the point-location of any robot lies within the cone spanned by vectors* $\binom{-1}{-3}, \binom{1}{-3}$.

Next we present some preliminaries toward describing our k-phase exploration algorithms. A *phase* is a pair (s, r) where $s \in [0, 1]$ is a speed and $r \in \mathbb{R}$ is a *distance ratio*, possibly negative. An *l-phase algorithm* is determined by a position p_0 on the line and a sequence $S = (s_1, r_1), \ldots, (s_k, r_l)$ of l phases (movement instructions). Whenever $r_i x < 0$, movement will be to the left, whereas $r_i x > 0$ will correspond to movement to the right.

l-PHASE EXPLORATION: GIVEN p_0 AND $S = (s_1, r_1), \ldots, (s_l, r_l)$
Go to p_0 at speed $1/3$
repeat
  $x \leftarrow$ current position
  for $i = 1, \ldots, l$ **do**
    Travel at speed s_i for a distance of $r_i \cdot x$
  end
end

We will make sure that each time the loop is executed, position x and corresponding time induce point-locations of the robots that lie in the boundary of the cone of Theorem 10. If a loop starts at location x, then it takes additional time $\sum_{i\in[l]} \frac{|r_i||x|}{s_i}$ to complete one iteration. We will be referring to quantity $1 + \sum_{i\in[l]} \frac{|r_i|}{3s_i}$ as the expansion factor of Exploration S.

4.2 Algorithm $\mathcal{A}(s)$: The Exploration, Chasing and Exit States

In this section we give a formal description of our evacuation algorithm. The most elaborate part of it is when robots are in Exploration States, in which they will perform 3-phase exploration. It can be shown that 3-phase exploration based evacuation algorithms that do not violate the constraints of problem EE_9^1 have expansion factor at most 4. Moreover, among those, the ones who minimize the induced energy consumption makes robots move at speed 1 in the first and third phase[2]. Robot's speed in the second phase will be denoted by s.

We now present a specific 3-phase exploration algorithm, that we denote by $\mathcal{A}(s)$, complying with the above conditions, with phases $(-1, 1), (4s/(1-s), s)$ and $(4-4s/(1-s), 1)$, where s is an exploration speed to be determined later. Robot L will execute the 3-phase exploration with starting position -1, while robot R with starting position 2. When subroutine $travel(v, p)$ is invoked, the robot sets its speed to v and, from its current position, goes toward position p on the line until it reaches it. We depict the trajectories of the robots while in the Exploration State in Figure 1.

EXPLORATION STATE OF L
$travel(1/3, -1)$
$k \leftarrow 0$
repeat
  $travel(1, 0)$
  $travel(s, -4^{k+1} \cdot \frac{s}{1-s})$
  $travel(1, -4^{k+1})$
  $k \leftarrow k + 1$
end

EXPLORATION STATE OF R
$travel(1/3, 2)$
$k \leftarrow 0$
repeat
  $travel(1, 0)$
  $travel(s, 2 \cdot 4^{k+1} \cdot \frac{s}{1-s})$
  $travel(1, 2 \cdot 4^{k+1})$
  $k \leftarrow k + 1$
end

[2] The proof of these facts is lengthy and technical, and is not required for the correctness of our algorithm, rather it only justifies some parameter choices

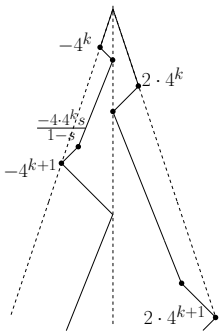


Figure 1 A representation of position (x-axis, vertical dashed line is 0) and time (y-axis), and the trajectory followed by the two robots (solid lines). The two diagonal dashed lines form the "1/3 cone" of Theorem 10.

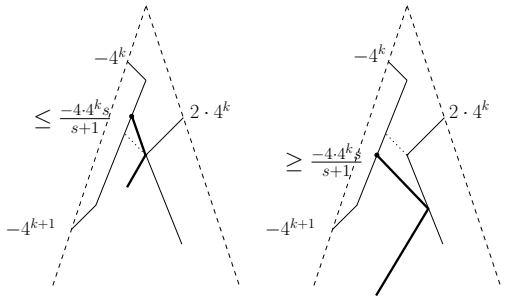


Figure 2 The robots' behavior when the exit is found by L is indicated by the bold line. In the first case (left), the catch-up speed is slower than 1 (and the rendezvous is realized at the turning point of the non-finder), whereas it is 1 in the second case (right).

A complete execution of one repeat loop within the Exploration State will be referred to as a *round*. Variable k counts the number of completed rounds. Each robot stays in the Exploration State till the exit it found. When switching to the Chasing state (which happens only for the exit finder), robot remembers its current value of counter k, as well as the distance d of the exit to the origin. Based on these values (as well as s) it calculates the most efficient trajectory in order to catch the other robot (predicting, when applicable, that the rendezvous can be realized while the other robot is approaching the exit finder). When the rendezvous is realized, robots store their current distance p to the origin, as well as the time t that has already passed. Then, robots need to travel distance $p + d$ to reach the exit. Knowing they have time $9d - t$ remaining, they go to the exit together as slow as possible to reach the exit in time exactly $9d$. Figure 2 provides an illustration of the behavior of the robots after finding the exit.

Chasing State
$K \leftarrow 4^k$
if *I am R* **then**
  $K \leftarrow 2 \cdot 4^k$
end
$s' \leftarrow \min\left\{\frac{d}{4K - d/s}, 1\right\}$
Travel toward the other robot at speed s' until meeting it at distance p from the origin, and at time t.

Exit State
$\bar{s} \leftarrow \frac{p+d}{9d-t}$
Go toward the exit with speed $\bar{s}$.

4.3 Performance Analysis & an Optimal Choice for Parameter s

In this section we are ready to provide the details for proving Theorem 16. Evacuation algorithm $\mathcal{A}(s)$ is not feasible to EE_c^b for all values of speed parameter s (of the Exploration States). We will show later that trajectories induce evacuation time at most $9d$ only if $s \in [1/3, 1/2]$. In what follows, and even though we have not fixed the value of s yet, we

will assume that s has some value between $1/3$ and $1/2$. The purpose of this section is to fix a value for parameter s, show that $\mathcal{A}(s)$ is feasible to EE_9^1, and subsequently compute the induced energy consumption and competitive ratio. As a reminder, each iteration of the repeat loop of the Exploration States is called a round, and k is a counter for these rounds.

▶ **Proposition 11.** *For every $k \geq 0$, and at the start of its k-th round,*
- *robot L is at position -4^k at time $3 \cdot 4^k$, and*
- *robot R, is at position $2 \cdot 4^k$ at time $6 \cdot 4^k$.*

Let $X \in \{L, R\}$ be one of the robots. We define $K(X,k) = 4^k$ if $X = L$, and $K(X,k) = 2 \cdot 4^k$ if $X = R$, i.e. the position of X at the start of round k. We will often analyze 3 cases for the distance d of the exit with respect to $K := K(X,k)$ (as it also appears in the description of the Chasing State), associated with the following closed intervals

$$D_1(K) := [K, 4Ks/(s+1)],\ D_2(K) := [4Ks/(s+1), 4Ks/(1-s)],\ D_3(K) := [4Ks/(1-s), 4K].$$

We may simply write D_1, D_2 and D_3 if K is clear from the context. Note that during the second phase of round K, robot L explores D_1 and D_2, whereas D_3 is explored during the third phase. The same statement holds for R. The following lemma will be useful in analyzing the worst case evacuation time and energy consumption of our algorithm.

▶ **Lemma 12.** *Suppose that robot $X \in \{L, R\}$ finds the exit at distance d when its round counter has value k. Let p and t be, respectively, the position and time at which X first meets with the other robot after having found the exit, and set $K := K(X,k)$. Then the following hold:*
1. *If $d \in D_1$, then $p = 0$ and $t = 8K$.*
2. *If $d \in D_2$, then $|p| = \frac{d+ds-4Ks}{1-s}$ and $t = 8K + \frac{d+d/s-4K}{1-s}$.*
3. *If $d \in D_3$, then $|p| = 2ds/(1-s)$ and $t = 8K + 2d + 2ds/(1-s)$.*

Using the lemma above, we can now prove that $\mathcal{A}(s)$ meets the speed bound and the evacuation time bound.

▶ **Lemma 13.** *For any $s \in [1/3, 1/2]$, evacuation algorithm $\mathcal{A}(s)$ is feasible to EE_9^1.*

Lemma 12 allows us to derive the speed s_{b1}, s_{b2} and s_{b3} at which both robots go toward the exit after meeting for the cases $d \in D_1, d \in D_2$ and $d \in D_3$, respectively. We also know the speed s_{c1} at which the exit-finder catches up to the other robot when $d \in D_1$. We define

$$s_{b1} := \tfrac{d}{9d-8K},\quad s_{c1} = \tfrac{d}{4K-d/s},\quad s_{b2} := \tfrac{2d-4Ks}{d(8-9s-1/s)+4K(2s-1)},\quad s_{b3} := \tfrac{d(1+s)}{d(7-9s)+8K(s-1)}$$

The speed s_{b2} is a simple rearrangement of the speed $\frac{d+qs}{9d-(8K+q)}$, where $q = \frac{d+d/s-4K}{1-s}$, and s_{b3} is obtained by rearranging $\frac{d+2ds/(1-s)}{9d-(8K+2d+2ds/(1-s))}$.

Next we compute the energy consumption. For given K, d and s, denote by $E_L(K,d,s)$ the energy spent by robot L from time 3 to time $9d$ when it exits. Similarly, $E_R(K,d,s)$ is the energy spent by R from time 6 to time $9d$. Then, the energy consumption is $E(K,d,s) := \frac{1}{3} + E_L(K,d,s) + E_R(K,d,s)$. For any K and s, we also define $F(K,s) := (K-1)(5-4s(s+1))$.

▶ **Lemma 14.** *Suppose that robot $X \in \{L, R\}$ finds the exit at distance d when its round counter has value k, and let $K := K(X,k)$. Then*

$$E(K,d,s) = \frac{1}{3} + \begin{cases} F(K,s) + 3K + d(s^2 + s_{c1}^2 + 2s_{b1}^2) & \text{if } d \in D_1 \\ F(K,s) + 3K + \left(\dfrac{2d-4Ks}{1-s}\right)(1 + s^2 + 2s_{b2}^2) & \text{if } d \in D_2 \\ F(K,s) + 3K - 4Ks(s+1) + \dfrac{2d}{1-s}(s^3 + s_{b3}^2(s+1) + 1) & \text{if } d \in D_3. \end{cases}$$

Denote by $E_i(k,d,s)$ the value of $E(K,d,s)$ when $d \in D_i$, $i = 1,2,3$. Our intension now is to fix speed value s that solves the following Nonlinear Program

$$\min_{s\in[1/3,1/2]} \left\{ \max \left\{ \sup_{d\in D_1, k\geq 1, X} \frac{E_1(K,d,s)}{d}, \sup_{d\in D_2, k\geq 1, X} \frac{E_2(K,d,s)}{d}, \sup_{d\in D_3, k\geq 1, X} \frac{E_3(K,d,s)}{d} \right\} \right\}.$$

For every $s \in [1/3, 1/2]$ we show in Lemma 15 that $\frac{E_1(K,d,s)}{d}$ is decreasing in $d \in D_1$, that $\frac{E_2(K,d,s)}{d}$ is increasing in $d \in D_2$, and that $\frac{E_3(K,d,s)}{d}$ is decreasing in $d \in D_3$. Then, the best parameter s can be chosen so as to make all worst case valued $\frac{E_i(K,d,s)}{d}$ equal (if possible) when $i = 1,2,3$. The optimal s can be found by numerically finding the roots of a high degree polynomial, and accordingly, we heuristically set $s = 0.39403$, inducing the best possible energy consumption for algorithm $\mathcal{A}(s)$. All relevant formal arguments are within the proof of the next lemma.

▶ **Lemma 15.** *On instance d of EE_9^1, algorithm $\mathcal{A}(s)$ induces energy consumption at most $8.42588d$, when $s = 0.39403$.*

By Lemma 15, we conclude that for the specific value of s, algorithm $\mathcal{A}(s)$ has competitive ratio $\frac{9^2}{2} 8.42588 \approx 341.24814$, concluding the proof of Theorem 16.

▶ **Theorem 16.** *For every $c, b > 0$ with $cb = 9$, there is an evacuation algorithm for unbounded-memory autonomous robots solving EE_c^b inducing energy consumption $8.42588b^2 d$ for instances d, and competitive ratio 341.24814.*

5 Discussion

The main contribution of our paper was to introduce an energy consumption model appropriate to linear search and investigate how the F2F communication model affects time/energy trade-offs until completion of the search by two robots, considering two different computational capabilities for the robots. Our approach inspired new algorithms that take better account of the impact of the change in the speed of the robots during the course of the search and leads to better understanding through evaluation of trade-offs of the overall performance of the algorithms.

Our paper raises several interesting problems worth investigating. In addition to improving the trade-offs in the algorithms proposed, one may wish to pursue new avenues for research by examining additional search domains, like the unit disk, in the spirit of [18]. It would also be natural to consider more realistic models of linear search with multiple agents some of which may be faulty [23, 25]. Further, it would be interesiting to investigate randomized search algorithms in our setting as well as more general models in which the energy loss experienced by a robot traveling a distance x at constant speed s is given by $s^a x$, for some fixed positive exponent a.

References

1. S. Alpern and S. Gal. *The theory of search games and rendezvous*. Springer, 2003.
2. R. Baeza Yates, J. Culberson, and G. Rawlins. Searching in the plane. *Information and Computation*, 106(2):234–252, 1993.
3. R. Baeza-Yates and R. Schott. Parallel searching in the plane. *Computational Geometry*, 5(3):143–154, 1995.
4. E. Bampas, J. Czyzowicz, L. Gasieniec, D. Ilcinkas, R. Klasing, T. Kociumaka, and D. Pajak. Linear search by a pair of distinct-speed robots. *Algorithmica*, 81(1):317–342, 2019.

5 G. K. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge Mathematical Library. Cambridge University Press, 2000.

6 A. Beck. On the linear search problem. *Israel J. of Mathematics*, 2(4):221–228, 1964.

7 R. Bellman. An optimal search. *SIAM Review*, 5(3):274–274, 1963.

8 M. Blum and D. Kozen. On the power of the compass (or, why mazes are easier to search than graphs). In *FOCS*, pages 132–142, 1978.

9 P. Bose and J.-L. De Carufel. A General Framework for Searching on a Line. *Theoretical Computer Science*, pages 703:1–17, 2017.

10 P. Bose, J.-L. De Carufel, and S. Durocher. Searching on a line: A complete characterization of the optimal solution. *Theoretical Computer Science*, pages 569:24–42, 2015.

11 S. Brandt, K.-T. Foerster, B. Richner, and R. Wattenhofer. Wireless Evacuation on m Rays with k Searchers. In *SIROCCO*, pages 140–157, 2017.

12 S. Brandt, F. Laufenberg, Y. Lv, D. Stolz, and R. Wattenhofer. Collaboration without Communication: Evacuating Two Robots from a Disk. In *CIAC*, pages 104–115, 2017.

13 S. Brandt, J. Uitto, and R. Wattenhofer. A Tight Lower Bound for Semi-Synchronous Collaborative Grid Exploration. In *DISC*, pages 13:1–13:17, 2018.

14 L. Budach. Automata and labyrinths. *Math. Nachrichten*, 86:195–282, 1978.

15 M. Chrobak, L. Gasieniec, Gorry T., and R. Martin. Group Search on the Line. In *SOFSEM*, pages 164–176. Springer, 2015.

16 H. Chuangpishit, K. Georgiou, and P. Sharma. Average Case - Worst Case Tradeoffs for Evacuating 2 Robots from the Disk in the Face-to-Face Model. In *ALGOSENSORS'18*. Springer, 2018.

17 S. A. Cook and C. Rackoff. Space lower bounds for maze threadability on restricted machines. *SIAM Journal on Computing*, 9(3):636–652, 1980.

18 J. Czyzowicz, L. Gasieniec, T. Gorry, E. Kranakis, R. Martin, and D. Pajak. Evacuating Robots via Unknown Exit in a Disk. In *DISC*, pages 122–136. Springer, 2014.

19 J. Czyzowicz, K. Georgiou, R. Killick, E. Kranakis, D. Krizanc, L. Narayanan, J. Opatrny, and S. Shende. God Save the Queen. In *(FUN)*, pages 16:1–16:20, 2018.

20 J. Czyzowicz, K. Georgiou, R. Killick, E. Kranakis, D. Krizanc, L. Narayanan, J. Opatrny, and S. Shende. Priority Evacuation from a Disk Using Mobile Robots. In *SIROCCO*, pages 209–225, 2018.

21 J. Czyzowicz, K. Georgiou, R. Killick, E. Kranakis, M. Lafond, D. Krizanc, L. Narayanan, J. Opatrny, and S. Shence. Energy Consumption of Group Search on a Line. *CoRR*, abs/1904.09714, 2019. `arXiv:1904.09714`.

22 J. Czyzowicz, K. Georgiou, and E. Kranakis. Group Search and Evacuation. In *Distributed Computing by Mobile Entities, Current Research in Moving and Computing, LNCS*, volume 11340, pages 335–370, 2019.

23 J. Czyzowicz, K. Georgiou, E. Kranakis, D. Krizanc, L. Narayanan, J. Opatrny, and S. Shende. Search on a Line by Byzantine Robots. In *ISAAC*, pages 27:1–27:12, 2016.

24 J. Czyzowicz, K. Georgiou, E. Kranakis, L. Narayanan, J. Opatrny, and B. Vogtenhuber. Evacuating Robots from a Disc Using Face to Face Communication. In *CIAC 2015*, pages 140–152, 2015.

25 J. Czyzowicz, E. Kranakis, D. Krizanc, L. Narayanan, and Opatrny J. Search on a Line with Faulty Robots. In *PODC*, pages 405–414. ACM, 2016.

26 J. Czyzowicz, E. Kranakis, D. Krizanc, L. Narayanan, J. Opatrny, and M. Shende. Linear Search with Terrain-Dependent Speeds. In *CIAC*, pages 430–441, 2017.

27 J. Czyzowicz, E. Kranakis, K. Krizanc, L. Narayanan, J. Opatrny, and S. Shende. Wireless Autonomous Robot Evacuation from Equilateral Triangles and Squares. In *ADHOCNOW*, pages 181–194. Springer, 2015.

28 E. D. Demaine, S. P. Fekete, and S. Gal. Online searching with turn cost. *Theoretical Computer Science*, 361(2):342–355, 2006.

29 Y. Emek, T. Langner, D. Stolz, J. Uitto, and R. Wattenhofer. How Many Ants Does it Take to Find the Food? *Theor. Comput. Sci.*, page 608:255–267, 2015.

30 S. Gal. *Search Games*. Wiley Encyclopedia for Operations Research and Management Science, 2011.

31 K. Georgiou, G. Karakostas, and E. Kranakis. Search-and-Fetch with One Robot on a Disk - (Track: Wireless and Geometry). In *ALGOSENSORS*, pages 80–94, 2016.

32 K. Georgiou, G. Karakostas, and E. Kranakis. Search-and-Fetch with 2 Robots on a Disk - Wireless and Face-to-Face Communication Models. In *ICORES*, pages 15–26. SciTePress, 2017.

33 F. Hoffmann, C. Icking, R. Klein, and K. Kriegel. The polygon exploration problem. *SIAM Journal on Computing*, 31(2):577–600, 2001.

34 M.-Y. Kao, J. H. Reif, and S. R. Tate. Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. *Information and Computation*, 131(1):63–79, 1996.

35 J. Kleinberg. On-line search in a simple polygon. In *SODA*, pages 8–15. SIAM, 1994.

36 I. Lamprou, R. Martin, and S. Schewe. Fast Two-Robot Disk Evacuation with Wireless Communication. In *DISC*, pages 1–15, 2016.

37 D. Pattanayak, H. Ramesh, P.S. Mandal, and S. Schmid. Evacuating Two Robots from Two Unknown Exits on the Perimeter of a Disk with Wireless Communication. In *ICDCN*, pages 20:1–20:4, 2018.

38 O. Reingold. Undirected st-connectivity in log-space. In *STOC*, pages 376–385, 2005.

39 H.-A. Rollik. Automaten in planaren Graphen. *Acta Informatica*, 13(3):287– 298, 1980.

Computing Exact Solutions of Consensus Halving and the Borsuk-Ulam Theorem

Argyrios Deligkas
Department of Computer Science, University of Liverpool, Liverpool, UK
Leverhulme Research Centre for Functional Materials Design, Liverpool, UK
argyrios.deligkas@liv.ac.uk

John Fearnley
Department of Computer Science, University of Liverpool, Liverpool, UK
john.fearnley@liv.ac.uk

Themistoklis Melissourgos
Department of Computer Science, University of Liverpool, Liverpool, UK
t.melissourgos@liv.ac.uk

Paul G. Spirakis
Department of Computer Science, University of Liverpool, Liverpool, UK
Computer Engineering and Informatics Department, University of Patras, Patras, Greece
p.spirakis@liv.ac.uk

Abstract

We study the problem of finding an exact solution to the consensus halving problem. While recent work has shown that the approximate version of this problem is PPA-complete [28, 29], we show that the exact version is much harder. Specifically, finding a solution with n agents and n cuts is FIXP-hard, and deciding whether there exists a solution with fewer than n cuts is ETR-complete. We also give a QPTAS for the case where each agent's valuation is a polynomial.

Along the way, we define a new complexity class BU, which captures all problems that can be reduced to solving an instance of the Borsuk-Ulam problem exactly. We show that FIXP $\subseteq$ BU $\subseteq$ TFETR and that LinearBU = PPA, where LinearBU is the subclass of BU in which the Borsuk-Ulam instance is specified by a linear arithmetic circuit.

2012 ACM Subject Classification Theory of computation → Complexity classes

Keywords and phrases PPA, FIXP, ETR, consensus halving, circuit, reduction, complexity class

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.138

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Related Version A full version of the paper is available at http://arxiv.org/abs/1903.03101.

Funding *John Fearnley*: The work of this author was supported by the EPSRC grant EP/P020909/1.
Paul G. Spirakis: The work of this author was supported by the EPSRC grant EP/P02002X/1.

1 Introduction

Dividing resources among agents in a fair manner is among the most fundamental problems in multi-agent systems [16]. Cake cutting [6, 8, 7, 15], and rent division [14, 33, 25] are prominent examples of problems that lie in this category. At their core, each of these problems has a desired solution whose existence is usually proved via a theorem from algebraic topology such as Brouwer's fixed point theorem, Sperner's lemma, or Kakutani's fixed point theorem.

In this paper, we focus on a fair-division problem called *consensus-halving*: an object A represented by $[0, 1]$ is to be divided into two halves A_+ and A_-, so that n agents agree that A_+ and A_- have the same value. Provided the agents have bounded and continuous

valuations over A, this can always be achieved using at most n cuts, and this fact can be proved via the Borsuk-Ulam theorem from algebraic topology [44]. The necklace splitting and ham-sandwich problems are two other examples of fair-division problems for which the existence of a solution can be proved via the Borsuk-Ulam theorem [4, 5, 37].

Recent work has further refined the complexity status of *approximate* consensus halving, in which we seek a division of the object so that every agent agrees that the values of A_+ and A_- differ by at most ϵ. Since the problem always has a solution, it lies in TFNP, which is the class of function problems in NP that always have a solution. More recent work has shown that the problem is PPA-complete [28], even for ϵ that is inverse-polynomial in n [29]. The problem of deciding whether there exists an approximate solution with k-cuts when $k < n$ is NP-complete [27]. These results are particularly notable, because they identify consensus halving as one of the first natural PPA-complete problems.

While previous work has focused on approximate solutions to the problem, in this paper we study the complexity of solving the problem *exactly*. For problems in the complexity class PPAD, which is a subclass of both TFNP and PPA, prior work has found that there is a sharp contrast between exact and approximate solutions. For example, the Brouwer fixed point theorem is the theorem from algebraic topology that underpins PPAD. Finding an approximate Brouwer fixed point is PPAD-complete [37], but finding an exact Brouwer fixed point is complete for (and the defining problem of) a complexity class called FIXP [26].

It is believed that FIXP is significantly harder than PPAD. While PPAD $\subseteq$ TFNP $\subseteq$ FNP, there is significant doubt about whether FIXP $\subseteq$ FNP. The reason for this is that there are Brouwer instances for which all solutions are irrational. This is not particularly relevant when we seek an approximate solution, but is a major difficulty when we seek an exact solution. For example, the square-root-sum problem asks us to decide for integers a_1, a_2, $\ldots$, a_n, t, whether $\sum_{i=1}^{n} \sqrt{a_i} \leq t$. This deceptively simple problem is not known to lie in NP, and can be reduced to the problem of finding an exact Brouwer fixed point [26], which provides evidence that FIXP may be significantly harder than FNP.

Our contribution. In this paper, we study the complexity of solving the consensus halving problem exactly. In our formulation of the problem, the valuation function of the agents is presented as an arbitrary arithmetic circuit, and the task is to cut A such that all agents agree that A_+ and A_- have exactly the same valuation. We study two problems. The (n, n)-Consensus Halving problem asks us to find an exact solution for n-agents using at most n-cuts, while the (n, k)-Consensus Halving problem asks us to decide whether there exists an exact solution for n-agents using at most k-cuts, where $k < n$.

Our results for (n, n)-Consensus Halving are intertwined with a new complexity class that we call BU. This class consists of all problems that can be reduced in polynomial time to the problem of finding a solution of the Borsuk-Ulam problem. We show that (n, n)-Consensus Halving lies in BU, and is FIXP hard. The hardness for FIXP implies that the exact variant of consensus halving is significantly harder than the approximate variant: while the approximate problem is PPA-complete, the exact variant is unlikely to be in FNP.

We show that (n, k)-Consensus Halving is ETR-complete. The complexity class ETR consists of all decision problems that can be formulated in the *existential theory of the reals*. It is known that NP $\subseteq$ ETR $\subseteq$ PSPACE [17], and it is generally believed that ETR is distinct from the other two classes. So our result again shows that the exact version of the problem seems to be much harder than the approximate version, which is NP-complete [27].

Just as FIXP can be thought of as the exact analogue of PPAD, we believe that BU is the exact analogue of PPA, and we provide some evidence to justify this. It has been shown that LinearFIXP = PPAD [26], which is the version of the class in which arithmetic circuits are

restricted to produce piecewise *linear* functions (FIXP allows circuits to compute piecewise polynomials). We likewise define LinearBU, which consists of all problems that can be reduced to a solution of a Borsuk-Ulam problem using a piecewise linear function, and we show that LinearBU = PPA.

The containment LinearBU ⊆ PPA can be proved using similar techniques to the proof that LinearFIXP ⊆ PPAD. However, the proof that PPA ⊆ LinearBU utilises our BU containment result for consensus halving. In particular, when the input to the consensus halving problem is a piecewise linear function, our containment result shows that the problem actually lies in LinearBU. The PPA-hardness results for consensus halving show that piecewise-linear-consensus halving is PPA-hard, which completes the containment [28, 29].

Finally, we show that, for the case where each agent has a non-piecewise polynomial valuation of constant (resp. $O(\log n)$) degree, an approximate solution to the problem can be found using $O(\log n)$ (resp. $O(poly \log n)$) cuts, which then yields a QPTAS for the problem.

For detailed proofs of the results presented, we refer the reader to the full version [22].

Related work. Although for a long period there were a few results about PPA, recently there has been a flourish of PPA-completeness results. The first PPA-completeness result was given by [32] who showed PPA-completeness of the Sperner problem for a non-orientable 3-dimensional space. In [30] this result was strengthened for a non-orientable and locally 2-dimensional space. In [3], 2-dimensional Tucker was shown to be PPA-complete; this result was used in [28, 29] to prove PPA-completeness for approximate consensus halving. In [23] PPA-completeness was proven for a special version of Tucker and for problems of the form "given a discrete fixed point in a non-orientable space, find another one". Finally, in [24] it was shown that octahedral Tucker is PPA-complete. In [35], a subclass of 2DLinearFIXP ⊆ FIXP that consists of 2-dimensional fixed-point problems was studied, and it was proven that 2DLinearFIXP = PPAD.

A large number of problems are now known to be ETR-complete: geometric intersection problems [34, 39], graph-drawing problems [1, 9, 18, 40], matrix factorization problems [42, 43], the Art Gallery problem [2], and deciding the existence of constrained (symmetric) Nash equilibria in (symmetric) normal form games with at least three players [10, 11, 12, 13, 31].

2 Preliminaries

2.1 Arithmetic circuits

An arithmetic circuit represents a continuous function $f : \mathbb{R}^n \to \mathbb{R}^m$, and is defined by a pair $(V, \mathcal{T})$, where V is a set of nodes and $\mathcal{T}$ is a set of gates. There are n nodes in V that are *input nodes*, and m nodes in V that are *output nodes*. When a value $x \in \mathbb{R}^n$ is presented at the input nodes, the circuit computes values for all other nodes $v \in V$, which we will denote as $x[v]$. The values of $x[v]$ for the m output nodes determine the value of $f(x) \in \mathbb{R}^m$.

Every node in V, other than the input nodes, is required to be the output of exactly one gate in $\mathcal{T}$. Each gate $g \in \mathcal{T}$ enforces an arithmetic constraint on its output node, based on the values of some other node in the circuit. Cycles are not allowed in these constraints. We allow the operations $\{\zeta, +, -, *\zeta, *, \max, \min\}$, which correspond to the gates shown in Table 1. Note that every gate computes a continuous function over its inputs, and thus any function f that is represented by an arithmetic circuit of this form is also continuous.

We study two types of circuits in this paper. *General* arithmetic circuits are allowed to use any of the gates that we have defined above. *Linear* arithmetic circuits allow only the operations $\{\zeta, +, -, *\zeta, \max, \min\}$, and the $*$ operation (multiplication of two variables) is disallowed. Observe that a linear arithmetic circuit computes a piecewise linear function.

Table 1 The types of gates and their constraints.

Gate	Constraint
$G_\zeta(\zeta, v_{out})$	$x[v_{out}] = \zeta$, where $\zeta \in \mathbb{Q}$
$G_+(v_{in1}, v_{in2}, v_{out})$	$x[v_{out}] = x[v_{in1}] + x[v_{in2}]$
$G_-(v_{in1}, v_{in2}, v_{out})$	$x[v_{out}] = x[v_{in1}] - x[v_{in2}]$
$G_{*\zeta}(\zeta, v_{in}, v_{out})$	$x[v_{out}] = x[v_{in1}] \cdot \zeta$, where $\zeta \in \mathbb{Q}$
$G_*(v_{in1}, v_{in2}, v_{out})$	$x[v_{out}] = x[v_{in1}] \cdot x[v_{in2}]$
$G_{\max}(v_{in1}, v_{in2}, v_{out})$	$x[v_{out}] = \max\{x[v_{in1}], x[v_{in2}]\}$
$G_{\min}(v_{in1}, v_{in2}, v_{out})$	$x[v_{out}] = \min\{x[v_{in1}], x[v_{in2}]\}$

2.2 The Consensus Halving problem

In the consensus halving problem there is an object A that is represented by the $[0, 1]$ line segment, and there are n agents. We wish to divide A into two (not necessarily contiguous) pieces such that every agent agrees that the two pieces have equal value. Simmons and Su [44] have shown that, provided the agents have bounded and continuous valuations over A, then we can find a solution to this problem using at most n cuts.

In this paper we consider instances of the consensus halving problem where the valuations of the agents are presented as arithmetic circuits. Each agent has a valuation function $f_i : [0, 1] \to \mathbb{R}$, but it is technically more convenient if they give us a representation of the *integral* of this function. So for each agent i, we are given an arithmetic circuit computing $F_i : [0, 1] \to \mathbb{R}$ where for all $x \in [0, 1]$ we have $F_i(x) = \int_0^x f(y)\, dy$. Then, the value of any particular segment of $[a, b]$ to agent i can be computed as $F_i(b) - F_i(a)$.

A solution to the consensus halving problem is given by a *k-cut* of the object A, which is defined by a vector of *cut-points* $(t_1, t_2, \ldots, t_k) \in [0, 1]^k$, and a vector of *signs* $(s_1, s_2, \ldots, s_{k+1}) \in \{-1, +1\}^{k+1}$. The cut-points t_i split A into up to $k + 1$ pieces. Note that they may in fact split A into fewer than $k + 1$ pieces in the case where two cut-points $t_i = t_j$ overlap. We define X_i to be the ith piece of A, meaning that $X_0 = [0, t_1]$, $X_i = [t_i, t_{i+1}]$ for all i in the range $1 \leq i < k$, and $X_k = [t_k, 1]$.

The sign vector determines which half of A the piece belongs to. We define $A_+ := \{X_i : s_i = +1\}$ and $A_- := \{X_i : s_i = -1\}$ to be the two halves. For each agent i, we denote the value A_+ to agent i as $F_i(A_+) := \sum_{[a,b] \in A_+} (F_i(b) - F_i(a))$, and we define $F_i(A_-)$ analogously. The k-cut is a solution to the consensus halving problem if $F_i(A_+) = F_i(A_-)$ for all agents i.

We define two computational problems. Simmons and Su [44] have proved that there always exists a solution using at most n-cuts, and our first problem is to find that solution.

(n, n)-Consensus Halving

Input: For every agent $i \in [n]$, an arithmetic circuit F_i computing the integral of agent i's valuation function.
Task: Find an n-cut for A such that $F_i(A_+) = F_i(A_-)$, for every agent $i \in [n]$.

For $k < n$ a solution to the problem may or may not exist. So we define the following decision variant of the problem.

(n, k)-Consensus Halving

Input: For every agent $i \in [n]$, an arithmetic circuit F_i computing the integral of agent i's valuation function.
Task: Decide whether there exists a k-cut for A such that $F_i(A_+) = F_i(A_-)$, for every agent $i \in [n]$.

For either of these two problems, if all of the inputs are represented by linear arithmetic circuits, then we refer to the problem as Linear Consensus Halving. We note that the known hardness results [27, 28] for consensus halving fall into this class. Specifically, those results produce valuations that are piecewise constant, and so the integral of these functions is piecewise linear, and these functions can be written down as linear arithmetic circuits [36].

3 The Class BU

The Borsuk-Ulam theorem states that every continuous function from the surface of an $(d+1)$-dimensional sphere to the d-dimensional Euclidean space maps at least one pair of antipodal points to the same point.

▶ **Theorem 1** (Borsuk-Ulam). *Let $f : S^d \to \mathbb{R}^d$ be a continuous function, where S^d is a $(d+1)$-dimensional sphere. Then, there exists an $x \in S^d$ such that $f(x) = f(-x)$.*

This theorem actually works for any domain D that is antipode-preserving homeomorphism of S^d, where by "antipode-preserving" we mean that for every $x \in D$ we have that $-x \in D$. In this paper, we choose S^d to be the sphere in $d+1$ dimensions with respect to L_1 norm: $S^d := \left\{x \mid x = (x_1, x_2, \ldots, x_{d+1}),\ \sum_{i=1}^{d+1} |x_i| = 1\right\}$.

We define the *Borsuk-Ulam* problem as follows.

Borsuk-Ulam

Input: A continuous function $f : \mathbb{R}^{d+1} \to \mathbb{R}^d$ presented as an arithmetic circuit.
Task: Find an $x \in S^d$ such that $f(x) = f(-x)$.

Note that we cannot constrain an arithmetic circuit to only take inputs from the domain S^d, so we instead put the constraint that $x \in S^d$ onto the solution.

The complexity class BU is defined as follows.

▶ **Definition 2** (BU). *The complexity class* BU *consists of all search problems that can be reduced to* Borsuk-Ulam *in polynomial time.*

3.1 LinearBU

When the input to a Borsuk-Ulam instance is a linear arithmetic circuit, then we call the problem Linear Borsuk-Ulam, and we define the class LinearBU as follows.

▶ **Definition 3** (LinearBU). *The complexity class* LinearBU *consists of all search problems that can be reduced to* Linear Borsuk-Ulam *in polynomial time.*

We will show that LinearBU = PPA. The proof that LinearBU $\subseteq$ PPA is similar to the proof that Etessami and Yannakakis used to show that LinearFIXP $\subseteq$ PPAD [26], while the fact that PPA $\subseteq$ LinearBU will follow from our results on consensus halving in Section 4.

To prove LinearBU $\subseteq$ PPA we will reduce to the *approximate* Borsuk-Ulam problem. It is well known that the Borsuk-Ulam theorem can be proved via Tucker's lemma, and Papadimitriou noted that this implies that finding an approximate solution to a Borsuk-Ulam problem lies in PPA [37]. This is indeed correct, but the proof provided in [37] is for a slightly different problem[1]. Since our results will depend on this fact, we provide our own definition and self-contained proof here. We define the approximate Borsuk-Ulam problem as follows.

ϵ-Borsuk-Ulam

Input: A continuous function $f : \mathbb{R}^{d+1} \to \mathbb{R}^d$ presented as an arithmetic circuit, along with two constants $\epsilon, \lambda \in \mathbb{R}$.
Task: Find one of the following.
1. A point $x \in S^d$ such that $\|f(x) - f(-x)\|_\infty \leq \epsilon$.
2. Two points $x, y \in S^d$ such that $\|f(x) - f(y)\|_\infty > \lambda \cdot \|x - y\|_\infty$.

The first type of solution is an approximate solution to the Borsuk-Ulam problem, while the second type of solution consists of any two points that witness that the function is not λ-Lipschitz continuous in the L_∞-norm. The second type of solution is necessary, because an arithmetic circuit is capable, through repeated squaring, of computing doubly-exponentially large numbers, and the reduction to Tucker may not be able to find an approximate solution for such circuits. We now re-prove the result of Papadimitriou in the following lemma.

▶ **Lemma 4** ([37]). *ϵ-Borsuk-Ulam is in PPA.*

To show that LinearBU $\subseteq$ PPA we will provide a polynomial time reduction from Linear Borsuk-Ulam to ϵ-Borsuk-Ulam. To do this, we follow closely the technique used by Etessami and Yannakakis to show that LinearFIXP $\subseteq$ PPAD [26]. The idea is to make a single call to ϵ-Borsuk-Ulam to find an approximate solution to the problem for a suitably small ϵ, and to then round to an exact solution by solving a linear program. To build the LP, we depend on the fact that we have access to the linear arithmetic circuit that represents f.

▶ **Lemma 5.** *Linear Borsuk-Ulam is in PPA.*

4 Containment Results for Consensus Halving

4.1 (n, n)-Consensus Halving is in BU and LinearBU = PPA

We show that (n, n)-Consensus Halving is contained in BU. Simmons and Su [44] show the existence of an n-cut solution to the consensus halving problem by applying the Borsuk-Ulam theorem, and we follow their approach in this reduction. However, we must show that the approach can be implemented using arithmetic circuits. We take care in the reduction to avoid G_* gates, and so if the inputs to the problem are all linear arithmetic circuits, then our reduction will produce a Linear Borsuk-Ulam instance. Hence, we also show that (n, n)-Linear Consensus Halving is in LinearBU.

▶ **Theorem 6.** *The following two containments hold.*
- (n, n)-Consensus Halving *is in* BU.
- (n, n)-Linear Consensus Halving *is in* LinearBU.

[1] The problem used in [37] presents the function as a polynomial-time Turing machine rather than an arithmetic circuit, and the Lipschitzness of the function is guaranteed by constraining the values that it can take.

We note that this also implies that PPA $\subseteq$ LinearBU, thereby completing the proof that PPA = LinearBU. Specifically, Filos-Ratsikas and Goldberg have shown that *approximate*-(n, n)-Consensus Halving is PPA-complete, and their valuation functions are piecewise constant. Therefore, the integrals of these functions are piecewise linear, and so their approximate-(n, n)-Consensus Halving instances can be reduced to (n, n)-Linear Consensus Halving. Hence (n, n)-Linear Consensus Halving is PPA-hard, which along with Lemma 5 implies the following corollary.

▶ **Corollary 7.** PPA = LinearBU.

4.2 (n, k)-Consensus Halving is in ETR

The existential theory of the reals consists of all true existentially quantified formulae using the connectives $\{\wedge, \vee, \neg\}$ over polynomials compared with the operators $\{<, \leq, =, \geq, >\}$. The complexity class ETR captures all problems that can be reduced in polynomial time to the existential theory of the reals.

We prove that (n, k)-Consensus Halving is in ETR. The reduction simply encodes the arithmetic circuits using ETR formulas, and then constrains $F_i(A_+) = F_i(A_-)$ for every agent i.

▶ **Theorem 8.** (n, k)-Consensus Halving *is in* ETR.

Using the same technique, we can also reduce Borsuk-Ulam to an ETR formula. In this case, we get an ETR formula that always has a solution, and so this result places the problem in TFETR, which is the subclass of ETR in which the formula is guaranteed to be true.

▶ **Theorem 9.** BU $\subseteq$ TFETR.

5 Hardness Results for Consensus Halving

In this section we prove that (n, n)-Consensus Halving is FIXP-hard and that $(n, n-1)$-Consensus Halving is ETR-hard. These two reductions share a common step of embedding an arithmetic circuit into a consensus halving instance. So we first describe this step, and then move on to proving the two individual hardness results.

5.1 Embedding a circuit in a Consensus Halving instance

Our approach is inspired by [27], who provided a reduction from ϵ-GCircuit [19, 38] to approximate consensus halving. However, our construction deviates significantly from theirs due to several reasons.

Firstly, the reduction in [27] works *only* for approximate consensus halving. Specifically, some valuations used in that construction have the form of $1/\epsilon$, where ϵ is the approximation guarantee, so the construction is not well-defined when $\epsilon = 0$ as it is in our case. Many of the gate gadgets used in [27] cannot be used due to this issue, including the max gate, which is crucially used in that construction to ensure that intermediate values do not get too large. We provide our own implementations of the broken gates. Our gate gadgets only work when the inputs and outputs lie in the range $[0, 1]$, and so we must carefully construct circuits for which this is always the case. The second major difference is that the reduction in [27] does not provide any method of multiplying two variables, which is needed in our case. We construct a gadget to do this, based on a more primitive gadget for squaring a single variable.

Table 2 The special types of gates, their constraints and ranges of input.

Special Gate	Constraint	Ranges
$G_{()^2}(v_{in}, v_{out})$	$x[v_{out}] = (x[v_{in}])^2$	$x[v_{in}] \in [0,1]$
$G_{*2}^{[0,1]}(v_{in}, v_{out})$	$x[v_{out}] = x[v_{in}] \cdot 2$	$x[v_{in}] \in [0,1/2]$
$G_{-}^{[0,1]}(v_{in1}, v_{in2}, v_{out})$	$x[v_{out}] = \max\{x[v_{in1}] - x[v_{in2}], 0\}$	$x[v_{in1}], x[v_{in2}] \in [0,1]$

Special circuit. Our reduction from an arithmetic circuit to consensus halving will use a very particular subset of gates. Specifically, we will not use $G_{\min}$, $G_{\max}$, or G_*, and we will restrict $G_{*\zeta}$ so that ζ must lie in $(0,1]$. We do however introduce three new gates, shown in Table 2. The gate $G_{()^2}$ squares its input, the gate $G_{*2}^{[0,1]}$ multiplies its input by two, but requires that the input be in $[0,1/2]$, and the gate $G_{-}^{[0,1]}$ is a special minus gate that takes as inputs $a, b \in [0,1]$ and outputs $\max\{a-b,0\}$. We note that $G_{\min}$, $G_{\max}$, and G_* can be implemented in terms of our new gates according to the following identities.

$$\max\{a,b\} = \frac{a+b}{2} + \frac{|a-b|}{2} = \frac{a}{2} + \frac{b}{2} + \frac{1}{2}\max\{a-b,0\} + \frac{1}{2}\max\{b-a,0\},$$

$$\min\{a,b\} = \frac{a+b}{2} - \frac{|a-b|}{2} = \frac{a}{2} + \frac{b}{2} - \frac{1}{2}\max\{a-b,0\} - \frac{1}{2}\max\{b-a,0\},$$

$$a \cdot b = 2\left[\left(\frac{a}{2} + \frac{b}{2}\right)^2 - \left(\left(\frac{a}{2}\right)^2 + \left(\frac{b}{2}\right)^2\right)\right].$$

Also, a very important requirement of the special circuit is that both inputs of any G_+ gate are in $[0,1/2]$. To make sure of that, we downscale the inputs before reaching the gate, and upscale the output, using the fact that $a + b = (a/2 + b/2) \cdot 2$.

The reduction to consensus halving. The reduction follows the general outline of the reduction given in [27]. The construction is quite involved, and so we focus on the high-level picture here. Each gate is implemented by 4 agents, namely ad, mid, cen, ex in the consensus halving instance. The values computed by the gates are encoded by the positions of the cuts that are required in order to satisfy these agents. Agent ad performs the exact mathematical operation of the gate, and feeds the outcome in mid, who "trims" it in accordance with the gate's actual operation. Then mid feeds her outcome to cen and ex, who make a copy of mid's correct value of the gate, with "negative" and "positive" labels respectively. This value with the appropriate label will be input to other gates.

The most important agents are the ones that perform the mathematical operation of each gate, i.e. agents ad. Figure 1 shows the part of the valuation functions of these agents that perform the operation. Each figure shows a valuation function for one of the agents, meaning that the blue regions represent portions of the object that the agent desires. The agent's valuation for any particular interval is the integral of this function over that interval.

To understand the high-level picture of the construction, let us look at the construction for $G_{*\zeta}$. The precise valuation functions of the agents in the construction ensure that there is exactly one *input* cut in the region v_{in}^+. The leftmost piece due to that cut in that region will belong to A_+, while the rightmost will belong to A_-. It is also ensured that there is exactly one *output* cut in the region v_{out}^a, and that the first piece in that region will belong to A_- and the second will belong to A_+.

Suppose that gate g_i in the circuit is of type $G_{*\zeta}$ and we want to implement it through a Consensus Halving instance. If we treat v_{in}^+ and v_{out}^a in Figure 1 as representing $[0,1]$, then agent ad_i will take as input a cut at point $x \in v_{in}^+$. In order to be satisfied, ad_i will

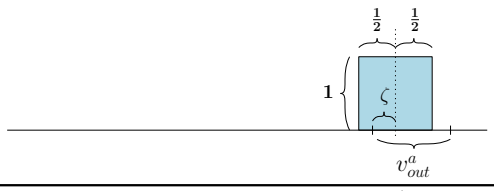
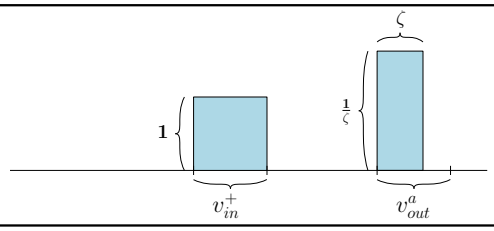
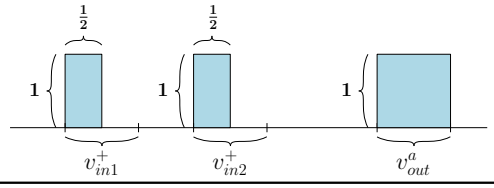
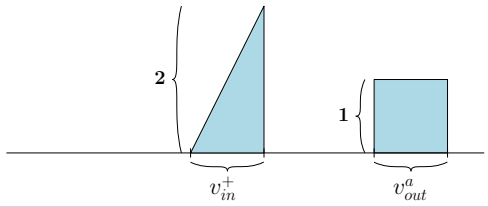
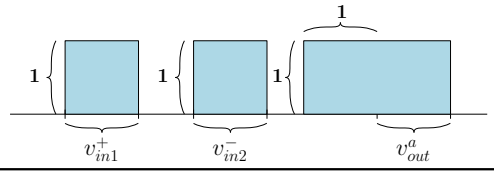
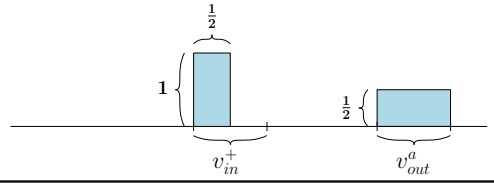
Gate	$\mathbf{G}_\pi(\mathbf{t})$	Valuation function
$\mathbf{G}_\zeta$	$\begin{cases} 1 & \textbf{if } t \in [v^a_{out,l} + \zeta - \frac{1}{2}, v^a_{out,l} + \zeta + \frac{1}{2}] \\ 0 & \textbf{otherwise} \end{cases}$	
$\mathbf{G}_{*\zeta}$	$\begin{cases} 1 & \textbf{if } t \in v^+_{in} \\ 1/\zeta & \textbf{if } t \in [v^a_{out,l}, v^a_{out,l} + \zeta] \\ 0 & \textbf{otherwise} \end{cases}$	
$\mathbf{G}_+$	$\begin{cases} 1 & \textbf{if } t \in [v^+_{in1,l}, v^+_{in1,l} + \frac{1}{2}] \\ 1 & \textbf{if } t \in [v^+_{in2,l}, v^+_{in2,l} + \frac{1}{2}] \\ 1 & \textbf{if } t \in v^a_{out} \\ 0 & \textbf{otherwise} \end{cases}$	
$\mathbf{G}_{()^2}$	$\begin{cases} 2(t - v^+_{in,l}) & \textbf{if } t \in v^+_{in} \\ 1 & \textbf{if } t \in v^a_{out} \\ 0 & \textbf{otherwise} \end{cases}$	
$\mathbf{G}^{[0,1]}_-$	$\begin{cases} 1 & \textbf{if } t \in v^+_{in1} \\ 1 & \textbf{if } t \in v^-_{in2} \\ 1 & \textbf{if } t \in [v^a_{out,l} - 1, v^a_{out,r}] \\ 0 & \textbf{otherwise} \end{cases}$	
$\mathbf{G}^{[0,1]}_{*\mathbf{2}}$	$\begin{cases} 1 & \textbf{if } t \in [v^+_{in,l}, v^+_{in,l} + \frac{1}{2}] \\ 1/2 & \textbf{if } t \in v^a_{out} \\ 0 & \textbf{otherwise} \end{cases}$	

Figure 1 Gates and their corresponding functions $G_\pi(t)$.

impose a cut at point $y \in v^a_{out}$, such that $F_i(A_+) = F_i(A_-)$, where: $F_i(A_+) = x + (\zeta - y)/\zeta$ and $F_i(A_-) = (1 - x) + y/\zeta$. Simple algebraic manipulation can be used to show that ad_i is satisfied only when $y = \zeta \cdot x$, as required.

We show that the same property holds for each of the gates in Figure 1. Two notable constructions are for the gates $G_{()^2}$ and $G^{[0,1]}_-$. For the gate $G_{()^2}$ the valuation function of agent ad is non-constant, which is needed to implement the non-linear squaring function.

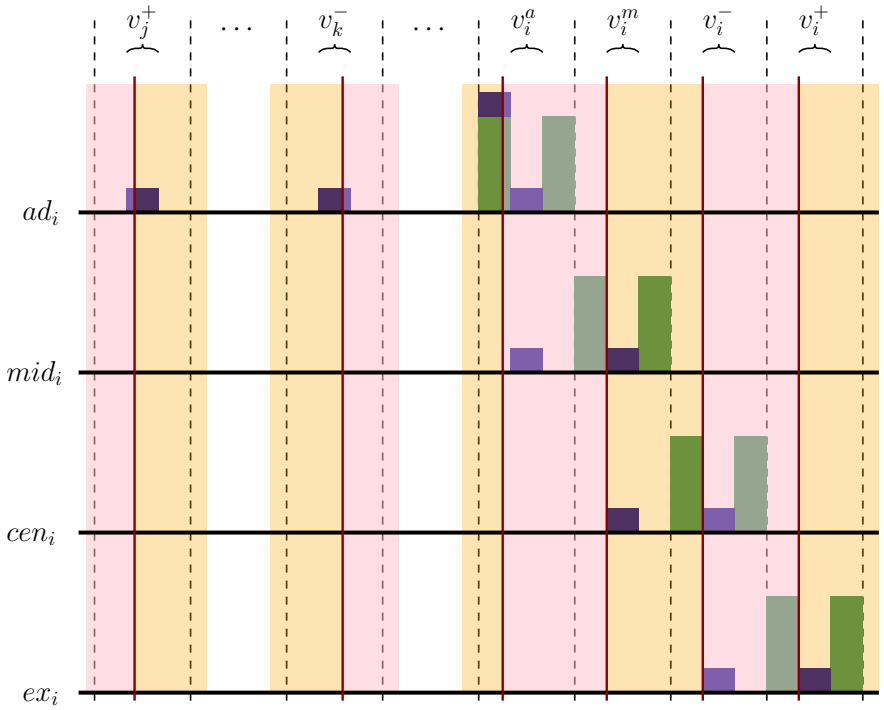


Figure 2 An example where the computation at the output $v_{out} := v_i$ of a $G_-^{[0,1]}$ gate with inputs $v_{in1} := v_j$ and $v_{in2} := v_k$ is simulated by the Consensus Halving instance. Here $x[v_j] = 1/4$ and $x[v_k] = 3/4$, hence $x[v_i] = 0$. The information about the values of the inputs is encoded by the cuts (red lines) in intervals v_j^+, and v_k^- imposed by agents ex_j and cen_k respectively. The blue and green shapes depict the area below the valuation function of each of the 4 agents. The pink regions have label "+" while the yellow have label "−". Agent ad_i performs the subtraction, by demanding that she is satisfied, and places a cut 1/10 to the left of the left endpoint of interval v_i^a. Then agent mid_i gets satisfied by placing a cut at exactly the left endpoint of interval v_i^m, thus encoding the value 0 which is the correct output value of the gate. Finally, agents cen_i, ex_i copy this value by enforcing similar cuts at the left endpoints of intervals v_i^- and v_i^+ respectively. The encoded values in the latter two intervals are the "negative" and "positive" version of $x[v_i]$.

For the gate $G_-^{[0,1]}$, note that the output region v_{out}^a only covers half of the possible output space. The idea is that if the result of $x[v_{in1}] - x[v_{in2}]$ is negative, then the output cut will lie before the output region, which will be interpreted as a zero output by agents mid, cen, ex in the construction. On the other hand, if the result is positive, the result will lie in the usual output range, and will be interpreted as a positive number. An example where $x[v_{in1}] = 1/4$ and $x[v_{in2}] = 3/4$ is shown in Figure 2.

Ultimately, this allows us to construct a consensus-halving instance that implements this circuit. This means that for any $x \in [0,1]^n$, we can encode x as a set of cuts, which then force cuts to be made at each gate gadget that encode the correct output for that gate. The full details of the construction are quite involved, but we are able to show the following result.

▶ **Lemma 10.** *Suppose that we are given an arithmetic circuit with the following properties.*

- *The circuit uses the gates* $G_\zeta, G_+, G_{*\zeta}, G_{()^2}, G_-^{[0,1]}, G_{*2}^{[0,1]}$.
- *Every* G_ζ *and* $G_{*\zeta}$ *has* $\zeta \in \mathbb{Q} \cap (0,1]$.
- *For every input* $x \in [0,1]^n$*, all intermediate values computed by the circuit lie in* $[0,1]$.

We can construct a consensus-halving instance that implements this circuit.

5.2 (n, n)-Consensus Halving is FIXP-hard

We show that (n, n)-Consensus Halving is FIXP-hard by reducing from the problem of finding a Nash equilibrium in a d-player game, which is known to be a FIXP-complete [26]. As shown in [26], this problem can be reduced to the Brouwer fixed point problem: given an arithmetic circuit computing a function $F : [0, 1]^n \to [0, 1]^n$, find a point $x \in [0, 1]^n$ such that $F(x) = x$. In a similar way to [27], we take this circuit and embed it into a consensus halving instance, with the outputs looped back to the inputs. Since Lemma 10 implies that our implementation of the circuit is correct, this means that any solution to the consensus halving problem must encode a point x satisfying $F(x) = x$.

One difficulty is that we must ensure that the arithmetic circuit that we build falls into the class permitted by Lemma 10. To do this, we carefully analyse the circuits produced in [26], and we modify them so that all of the preconditions of Lemma 10 hold.

▶ **Theorem 11.** (n, n)-Consensus Halving *is* FIXP*-hard.*

This theorem, along with Theorem 6 give the following corollary.

▶ **Corollary 12.** FIXP $\subseteq$ BU*.*

5.3 $(n, n-1)$-Consensus Halving is ETR-complete

We will show the ETR-hardness of $(n, n-1)$-Consensus Halving by reducing from the following problem Feasible, which is known to be ETR-complete [41].

▶ **Definition 13** (Feasible, Feasible$_{[0,1]}$)**.** *Let $p(x_1, \ldots, x_m)$ be a polynomial.* Feasible *asks whether there exists a point $(x_1, \ldots, x_m) \in \mathbb{R}^m$ that satisfies $p(x_1, \ldots, x_m) = 0$.* Feasible$_{[0,1]}$ *asks whether there exists a point $(x_1, \ldots, x_m) \in [0, 1]^m$ that satisfies p.*

The idea is to turn the polynomial into a circuit, and then embed that circuit into a consensus halving instance using Lemma 10. As before, the main difficulty is ensuring that the preconditions of Lemma 10 are satisfied. To do this, we must ensure that the the inputs to the circuit take values in $[0, 1]$, which is not the case if we reduce directly from Feasible. Instead, we first consider the problem Feasible$_{[0,1]}$, in which x is constrained to lie in $[0, 1]^n$ rather than $\mathbb{R}^n$, and we show the following result.

▶ **Lemma 14.** Feasible$_{[0,1]}$ *is* ETR*-complete.*

ETR$_{[0,1]}$ is the subclass of ETR in which variables are quantified over $[0, 1]^n$ rather than $\mathbb{R}^n$. The above lemma follows from the fact that ETR$_{[0,1]}$ = ETR, and the fact that Feasible$_{[0,1]}$ is ETR$_{[0,1]}$-hard. This equivalence of classes, together with the completeness of Feasible$_{[0,1]}$ may be of independent interest.

We then proceed to reduce Feasible$_{[0,1]}$ to $(n, n-1)$-Consensus Halving. We still don't quite meet the requirements of Lemma 10, because the intermediate terms may be outside $[0, 1]$. We resolve this by implementing a circuit $p^+(x)$ implementing only the positive terms of $p(x)$ downscaled appropriately, and a circuit $p^-(x)$ implementing the positive terms of $-p(x)$ again downscaled appropriately. The check agent is then satisfied if $p^+(x) = p^-(x)$, which can only occur when $p(x) = 0$.

There will be $n-1$ *choice* agents corresponding to the $(n-1)/4$ nodes of the circuit, who enforce that there is a cut for each of the nodes to the circuit, and together these cuts encode an input x to the polynomial. Each agent introduced by Lemma 10 has an associated cut that is forced by the construction used in that lemma, and these cuts compute the output of the associated gate.

So far, every agent has a corresponding cut that is forced by the construction. There is, however, one final *check* agent who has the following properties.

- If $p(x) = 0$, then the check agent agrees that A has been cut in half without an extra cut being made.
- If $p(x) \neq 0$, then the check agent requires one more cut to be made in order to be satisfied that A has been cut in half.

Hence, if there is a solution to FEASIBLE, then there is a solution to $\text{FEASIBLE}_{[0,1]}$, and there is a $(n-1)$-cut that solves the CONSENSUS HALVING instance. Otherwise there is no such solution.

▶ **Theorem 15.** $(n, n-1)$-CONSENSUS HALVING *is* ETR*-complete.*

6 A QPTAS for Consensus Halving with polynomial valuation functions

In this section we show that an approximate solution to the consensus halving problem can be found in quasi-polynomial time when each agent's valuation function is a single polynomial of constant or even polylogarithmic degree. We will do so by formulating the problem as a formula in the *approximate* existential theory of the reals, and then applying the approximation theorem proved in [20, 21].

Our result implies that these instances can be solved approximately using a polylogarithmic number of cuts. We note that this is one of the most general classes of instances for which we could hope to prove such a result: any instance in which n agents desire completely disjoint portions of the object can only be solved by an n-cut, and piecewise linear functions are capable of producing such a situation. So in a sense, we are exploiting the fact that this situation cannot arise when the agents have non-piecewise polynomial valuation functions.

▶ **Lemma 16.** *For every* CONSENSUS HALVING *instance with n agents, and every $\epsilon > 0$, if each agent's valuation function F_i is a single polynomial of degree at most $O(poly \log n)$, then there exists a k-cut, where $k := O(poly \log n)/\epsilon^4$, and pieces A_+ and A_- such that:*

- *every cut point is a multiple of $1/k = \frac{\epsilon^4}{O(poly \log n)}$;*
- *$|F_i(A_+) - F_i(A_-)| \leq \epsilon$, for every agent i.*

As a consequence, we can perform a brute force search over all possible k-cuts to find an approximate solution, which can be carried out in $n^{O(poly \log n/\epsilon^4)}$ time.

▶ **Theorem 17.** CONSENSUS HALVING *admits a QPTAS when the valuation function of every agent is a single polynomial of degree $O(poly \log n)$.*

References

1. Zachary Abel, Erik D Demaine, Martin L Demaine, Sarah Eisenstat, Jayson Lynch, and Tao B Schardl. Who needs crossings? Hardness of plane graph rigidity. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 51. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.
2. Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. The art gallery problem is ETR-complete. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 65–73. ACM, 2018.
3. James Aisenberg, Maria Luisa Bonet, and Sam Buss. 2-D Tucker is PPA complete. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:163, 2015.
4. Noga Alon. Splitting necklaces. *Advances in Mathematics*, 63(3):247–253, 1987.
5. Noga Alon and Douglas B West. The Borsuk-Ulam theorem and bisection of necklaces. *Proceedings of the American Mathematical Society*, 98(4):623–628, 1986.

6 Georgios Amanatidis, George Christodoulou, John Fearnley, Evangelos Markakis, Christos-Alexandros Psomas, and Eftychia Vakaliou. An improved envy-free cake cutting protocol for four agents. In *International Symposium on Algorithmic Game Theory*, pages 87–99. Springer, 2018.

7 Haris Aziz and Simon Mackenzie. A discrete and bounded envy-free cake cutting protocol for any number of agents. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 416–427. IEEE, 2016.

8 Haris Aziz and Simon Mackenzie. A discrete and bounded envy-free cake cutting protocol for four agents. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 454–464. ACM, 2016.

9 Daniel Bienstock. Some provably hard crossing number problems. *Discrete & Computational Geometry*, 6(3):443–459, 1991.

10 Vittorio Bilò and Marios Mavronicolas. The Complexity of Decision Problems about Nash Equilibria in Win-Lose Games. In *Proc. of SAGT*, pages 37–48, 2012.

11 Vittorio Bilò and Marios Mavronicolas. Complexity of rational and irrational Nash equilibria. *Theory of Computing Systems*, 54(3):491–527, 2014.

12 Vittorio Bilò and Marios Mavronicolas. A catalog of EXISTS-R-complete decision problems about Nash equilibria in multi-player games. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 47, 2016.

13 Vittorio Bilò and Marios Mavronicolas. Existential-R-complete decision problems about symmetric Nash equilibria in symmetric multi-player games. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 66, 2017.

14 Steven J Brams and D Marc Kilgour. Competitive fair division. *Journal of Political Economy*, 109(2):418–443, 2001.

15 Steven J Brams and Alan D Taylor. An envy-free cake division protocol. *The American Mathematical Monthly*, 102(1):9–18, 1995.

16 Steven J Brams and Alan D Taylor. *Fair Division: From cake-cutting to dispute resolution*. Cambridge University Press, 1996.

17 John Canny. Some Algebraic and Geometric Computations in PSPACE. In *Proc. of STOC*, pages 460–467, New York, NY, USA, 1988. ACM. doi:10.1145/62212.62257.

18 Jean Cardinal and Udo Hoffmann. Recognition and complexity of point visibility graphs. *Discrete & Computational Geometry*, 57(1):164–178, 2017.

19 Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibria. *Journal of the ACM (JACM)*, 56(3):14, 2009.

20 Argyrios Deligkas, John Fearnley, Themistoklis Melissourgos, and Paul G. Spirakis. Approximating the Existential Theory of the Reals. In George Christodoulou and Tobias Harks, editors, *Web and Internet Economics*, pages 126–139, Cham, 2018. Springer International Publishing.

21 Argyrios Deligkas, John Fearnley, Themistoklis Melissourgos, and Paul G. Spirakis. Approximating the Existential Theory of the Reals. *CoRR*, abs/1810.01393, 2018. arXiv:1810.01393.

22 Argyrios Deligkas, John Fearnley, Themistoklis Melissourgos, and Paul G. Spirakis. Computing Exact Solutions of Consensus Halving and the Borsuk-Ulam Theorem. *CoRR*, abs/1903.03101, 2019. arXiv:1903.03101.

23 Xiaotie Deng, Jack R Edmonds, Zhe Feng, Zhengyang Liu, Qi Qi, and Zeying Xu. Understanding PPA-completeness. In *Proceedings of the 31st Conference on Computational Complexity*, page 23. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.

24 Xiaotie Deng, Zhe Feng, and Rucha Kulkarni. Octahedral Tucker is PPA-complete. In *Electronic Colloquium on Computational Complexity Report TR17-118*, 2017.

25 Francis Edward Su. Rental harmony: Sperner's lemma in fair division. *The American mathematical monthly*, 106(10):930–942, 1999.

26 K. Etessami and M. Yannakakis. On the Complexity of Nash Equilibria and Other Fixed Points. *SIAM Journal on Computing*, 39(6):2531–2597, 2010. doi:10.1137/080720826.

27 Aris Filos-Ratsikas, Søren Kristoffer Stiil Frederiksen, Paul W. Goldberg, and Jie Zhang. Hardness Results for Consensus-Halving. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, pages 24:1–24:16, 2018.

28 Aris Filos-Ratsikas and Paul W. Goldberg. Consensus halving is PPA-complete. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 51–64, 2018.

29 Aris Filos-Ratsikas and Paul W Goldberg. The Complexity of Splitting Necklaces and Bisecting Ham Sandwiches. *arXiv preprint*, 2018. arXiv:1805.12559.

30 Katalin Friedl, Gábor Ivanyos, Miklos Santha, and Yves F Verhoeven. Locally 2-dimensional Sperner problems complete for the Polynomial Parity Argument classes. In *Italian Conference on Algorithms and Complexity*, pages 380–391. Springer, 2006.

31 Jugal Garg, Ruta Mehta, Vijay V Vazirani, and Sadra Yazdanbod. ETR-completeness for decision versions of multi-player (symmetric) Nash equilibria. *ACM Transactions on Economics and Computation (TEAC)*, 6(1):1, 2018.

32 Michelangelo Grigni. A Sperner lemma complete for PPA. *Information Processing Letters*, 77(5-6):255–259, 2001.

33 Claus-Jochen Haake, Matthias G Raith, and Francis Edward Su. Bidding for envy-freeness: A procedural approach to n-player fair-division problems. *Social Choice and Welfare*, 19(4):723–749, 2002.

34 Jiri Matousek. Intersection graphs of segments and EXISTS-R. *arXiv preprint*, 2014. arXiv:1406.2636.

35 Ruta Mehta. Constant rank bimatrix games are PPAD-hard. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 545–554, 2014. doi:10.1145/2591796.2591835.

36 Sergei Ovchinnikov. Max-min representation of piecewise linear functions. *Beiträge zur Algebra und Geometrie*, 43(1):297–302, 2002. URL: http://eudml.org/doc/225460.

37 Christos H Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994.

38 Aviad Rubinstein. Inapproximability of Nash equilibrium. *SIAM Journal on Computing*, 47(3):917–959, 2018.

39 Marcus Schaefer. Complexity of some geometric and topological problems. In *International Symposium on Graph Drawing*, pages 334–344. Springer, 2009.

40 Marcus Schaefer. Realizability of graphs and linkages. In *Thirty Essays on Geometric Graph Theory*, pages 461–482. Springer, 2013.

41 Marcus Schaefer and Daniel Stefankovic. Fixed Points, Nash Equilibria, and the Existential Theory of the Reals. *Theory Comput. Syst.*, 60(2):172–193, 2017. doi:10.1007/s00224-015-9662-0.

42 Yaroslav Shitov. A universality theorem for nonnegative matrix factorizations. *arXiv preprint*, 2016. arXiv:1606.09068.

43 Yaroslav Shitov. The complexity of positive semidefinite matrix factorization. *SIAM Journal on Optimization*, 27(3):1898–1909, 2017.

44 Forest W. Simmons and Francis Edward Su. Consensus-halving via theorems of Borsuk-Ulam and Tucker. *Mathematical Social Sciences*, 45(1):15–25, 2003.

Exploration of High-Dimensional Grids by Finite Automata

Stefan Dobrev
Institute of Mathematics, Slovak Academy of Sciences, Bratislava, Slovakia
http://www.ifi.savba.sk/~stefan/
stefan@ifi.savba.sk

Lata Narayanan
Department of CSSE, Concordia University, Montreal, Canada
https://users.encs.concordia.ca/~lata/
lata@cs.concordia.ca

Jaroslav Opatrny
Department of CSSE, Concordia University, Montreal, Canada
https://users.encs.concordia.ca/~opatrny/
opatrny@cs.concordia.ca

Denis Pankratov
Department of CSSE, Concordia University, Montreal, Canada
https://users.encs.concordia.ca/~denisp/
denisp@cs.concordia.ca

Abstract

We consider the problem of finding a treasure at an unknown point of an n-dimensional infinite grid, $n \geq 3$, by initially collocated finite automaton agents (scouts/robots). Recently, the problem has been well characterized for 2 dimensions for deterministic as well as randomized agents, both in synchronous and semi-synchronous models [12, 21]. It has been conjectured that $n+1$ randomized agents are necessary to solve this problem in the n-dimensional grid [17]. In this paper we disprove the conjecture in a strong sense: we show that *three* randomized synchronous agents suffice to explore an n-dimensional grid for *any* n. Our algorithm is optimal in terms of the number of the agents. Our key insight is that a constant number of finite automaton agents can, by their positions and movements, implement a stack, which can store the path being explored. We also show how to implement our algorithm using: four randomized semi-synchronous agents; four deterministic synchronous agents; or five deterministic semi-synchronous agents.

We give a different algorithm that uses 4 deterministic semi-synchronous agents for the 3-dimensional grid. This is provably optimal, and surprisingly, matches the result for 2 dimensions. For $n \geq 4$, the time complexity of the solutions mentioned above is exponential in distance D of the treasure from the starting point of the agents. We show that in the deterministic case, one additional agent brings the time down to a polynomial. Finally, we focus on algorithms that never venture much beyond the distance D. We describe an algorithm that uses $O(\sqrt{n})$ semi-synchronous deterministic agents that never go beyond $2D$, as well as show that any algorithm using 3 synchronous deterministic agents in 3 dimensions, if it exists, must travel beyond $\Omega(D^{3/2})$ from the origin.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Multi-agent systems, finite state machines, high-dimensional grids, robot exploration, randomized agents, semi-synchronous and synchronous agents

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.139

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Related Version A full version of the paper is available at https://arXiv.org/abs/1902.03693.

Funding Research supported by NSERC, Canada.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 139; pp. 139:1–139:16

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

1 Introduction

Motivated by the self-organizing behaviour of ants and other social insects, swarm robotics leverages the collective capability of a collection of extremely simple and inexpensive robots. Such robots have very limited computation and communication capabilities, and yet can collectively perform seemingly complex tasks such as: forage for food [14]; form patterns [26]; pull heavy objects [23]; and play *Für Elise* on the piano [15].

A series of recent papers [24, 22, 21, 12, 17] studies the conditions required for such primitive robots (also called agents or scouts) to search for a treasure placed at an unknown location in an infinite two-dimensional grid. In particular, they consider agents whose behaviour is controlled by a finite automaton (FA), who are equipped with a global compass, and can only communicate with other agents that are at the exact same grid location as themselves. Furthermore, this communication is limited to see the current state of other co-located agents. The primary question of interest is: *how many* such agents are needed to search for a treasure located at an unknown location in an infinite n-dimensional grid for $n \geq 2$? As shown in [12, 21] for $n = 2$, the answer depends on the computational power of the agents: whether or not they have access to random bits, the amount of memory they have, and whether or not they are synchronized. Note that for randomized algorithms, we require a finite mean hitting time for every node in the grid. The set of agents is *fully synchronous* if they operate by the same global clock; they are *semi-synchronous*[1] if in every time slot, a subset of adversarially scheduled agents is active. Full details of the agent models are given in Section 2.

The case of the 2-dimensional grid has been completely characterized. It has been shown that if the agents are deterministic and semi-synchronous, 4 agents are necessary [12] and sufficient [21]. If the agents are fully synchronous and deterministic, then 3 agents are necessary and sufficient [21]. In [17], the authors proved that 3 agents are necessary to search the 2-dimensional grid, even if they are fully synchronized and are randomized. They conjectured that in an n-dimensional grid, $n + 1$ agents would be necessary.

▶ **Conjecture 1** ([17]). *For $n \geq 3$, any search strategy on the n-dimensional infinite grid requires at least $n + 1$ agents.*

The main result of this paper is to disprove the above conjecture; we show that three randomized synchronous agents, or 5 deterministic semi-synchronous agents can explore any n-dimensional grid. These algorithms are completely different from previous algorithms for grid exploration, and are based on the key insight that a constant number of finite automata agents can, by their positions and movements, implement a stack that stores the path being explored.

1.1 Our results

Our main result is an algorithm for 3 randomized synchronous agents to explore an n-dimensional grid for any $n \geq 3$. This result is optimal, since 3 agents are necessary to explore even the 2-dimensional grid. Next we show how to "derandomize" the algorithm with the addition of one agent. If the agents are semi-synchronous, the algorithm can be implemented

[1] In some related literature [22, 17, 21] the same model was referred to as asynchronous. We follow the terminology of semi-synchronous of [12] and the vast literature on autonomous mobile robots to avoid confusion with a fully asynchronous model.

with the addition of one more agent, in both the randomized and deterministic cases. We also show that in the 3-dimensional grid, 4 deterministic semi-synchronous agents are sufficient for grid exploration.

The algorithms mentioned above have an exploration cost/time that is exponential in the volume of the smallest ball containing the treasure. In Section 5, we give an algorithm with exploration cost linear in the volume of the ball, which is similar to the algorithm for the 2-dimensional grid given in [21], but with an important modification that enables exploration of the 3-dimensional grid without increasing the number of agents. Our algorithm is optimal in the explored space and also in the number of agents, since 4 agents are necessary to explore even the 2-dimensional grid. In Section 4, we give a deterministic synchronous algorithm for exploring the n-dimensional grid that uses 5 agents and takes time polynomial in D, the distance from the origin to the treasure. A semi-synchronous implementation of this algorithm uses 6 agents. Table 1 shows our results.

Table 1 Exploration of an n dimensional infinite grid. Numbers marked with * indicate that the number of agents used or the exploration cost is optimal. We use c in the exploration cost to indicate a constant.

Model	Number of agents	Section	Exploration cost
Randomized Synchronous	3^*	Section 3.2	c^D
Randomized Semi-synchronous	4	Section 3.2	c^D
Deterministic Synchronous	4	Section 3.3	$O(2^{D+2n})$
	5	Section 4	$D^{O(n)}$
Deterministic Semi-synchronous	4^* $(n = 3)$	Section 5	$O(D^3)$ *
	5 $(n > 3)$	Section 3.3	$O(2^{3D+4n})$
	6 $(n > 3)$	Section 4	$D^{O(n)}$

In Section 6 we describe the following additional results. We give a lower bound of $\Omega(D^{3/2})$ on the distance from the origin that must be travelled by some agent in any 3-agent deterministic synchronous algorithm, and give an algorithm using $O(\sqrt{n})$ deterministic semi-synchronous agents in which no agent travels distance more than $2D$. Lastly, we extend our algorithms to agents without global compass. We show that one additional agent is sufficient in the semi-synchronous model, while two additional agents are sufficient in the synchronous model.

1.2 Related work

There is a lot of related literature on multi-agent systems and the exploration problem: from the early work on the cow-path problem to the more recent work on exploration of graphs, labyrinths, and grids by finite state agents [3, 6, 1, 2, 4, 5, 11, 16, 29, 32, 31, 34, 10, 19, 28, 8, 7, 13, 27, 9]. In this section we briefly mention the work that is most directly relevant to this paper.

The authors of [24] introduced the problem of k randomized mobile agents, starting from the same initial position, and searching for a treasure at an unknown location on the two-dimensional infinite grid. In their model, the agents are Turing machines, but cannot communicate at all. They show that if the agents have a constant approximation of k, the treasure can be found optimally in time $O(D + D^2/k)$, where D is the distance between the initial location and the treasure. The authors of [22] consider semi-synchronous and randomized FA agents and show that the same time complexity can be achieved. The relationship between the number of random bits available and the search time was studied in [33].

Emek et al. [21] posed the question of *how many* agents are required to find the treasure. They studied deterministic as well as randomized agents, synchronous as well as semi-synchronous agents, and FA agents, as well as agents that are controlled by a push-down automaton (PDA). They show that the problem can be solved by any of the following: 4 deterministic semi-synchronous FA agents; 3 deterministic synchronous agents; 3 randomized semi-synchronous FA agents; 1 deterministic FA together with 1 deterministic PDA agent; 1 randomized PDA agent. On the negative side they show that the problem cannot be solved by 2 deterministic (synchronous) FA agents; a single randomized FA agent; a deterministic PDA agent. Cohen et al. [17] prove that at least 2 FA agents are necessary to explore the one-dimensional grid and at least 3 FA agents are needed to explore the two-dimensional grid, thus proving the optimality of the FA-agent deterministic synchronous and randomized semi-synchronous algorithms in [21]. Recently it was shown that 3 deterministic semi-synchronous FA agents cannot perform exploration of the 2-dimensional grid [12], thus proving the optimality of the 4 FA-agent deterministic synchronous algorithm in [21].

A large body of work is devoted to the capabilities of autonomous mobile robots with very limited computational and communication abilities; see [25] for a comprehensive introduction. While we use some of that terminology in this paper, their robots are usually assumed to be identical, anonymous, and communication is limited to being able to "see" each other's positions, regardless of how far they are. In contrast, in our model, the robots follow different algorithms (this can be done by having different initial states of the same FSM), and only see other robots if they are at the same location, and they can exchange a message with the collocated robots. Equivalently, they can be assumed to see the current states of other robots at the same location. This is similar to the "robots with lights" model in the autonomous mobile robot literature [18].

2 Model and Notation

We use the same models (with the exception of Theorem 10 on agents without a global compass) as in [21, 12]. For completeness, we recall key definitions and introduce some notation in this section.

Our search domain is $\mathbb{Z}^n$ with the *Manhattan metric*, i.e., the distance between two points $p, p' \in \mathbb{Z}^n$ is defined as $||p-p'|| = \sum_{i=1}^{n} |p_i - p'_i|$. We refer to $\mathbb{Z}^n$ as the *n-dimensional integer grid* and its elements as *grid points, points, or cells.* A grid point $p = (p_1, p_2, \ldots p_i, \ldots, p_n)$ is *adjacent* to every grid point $(p_1, p_2, \ldots p'_i, \ldots, p_n)$, where $|p_i - p'_i| = 1$ for some i, $1 \leq i \leq n$. We assume that any two grid points cannot be distinguished from each other by an agent, and that includes the origin from which the search starts.

The search for the treasure in the grid is done using a fixed number of agents, each modelled by a finite automaton. These finite automata can be the same, except they typically have a different initial state. Two agents can exchange information with each other only when they occupy the same grid location at the same time. We assume that all agents have the same global n-dimensional compass. Initially, all agents are located in the same grid point, assumed without loss of generality to be the origin of the grid. The treasure is located at distance D from from the origin, where D is unknown to the agents.

Time is divided into discrete units. In each time unit an *active* agent performs a single *look-compute-move* cycle. In the look part of the cycle the agent sees the state of other agents located in its own grid point. In the compute part of the cycle the agent determines,

using its own state and those it sees, to which adjacent node to move to, if at all. The agent also determines its new state. Such a move is then executed in the move part of the cycle. When we consider randomized algorithms, we assume that an agent has access to an infinite one-way tape with i.i.d. random bits. We say that the system is *synchronous* if at each time unit all agents are active. We say that the system is *semi-synchronous* if at each time unit only a subset of agents, chosen by an adversarial scheduler, is active. The adversarial scheduler must schedule each agent infinitely often.

In addition to the question of whether $\mathbb{Z}^n$ can be fully explored by k agents, we are also interested in the efficiency of such exploration procedures. We refer to this measure of interest as *the exploration cost*. Intuitively, we measure how long it takes for k agents to visit all $\Theta(D^n)$ points in a sphere of radius D. In the synchronous model, this measure is simply the number of time units taken by the agents to complete the exploration. In the semi-synchronous case, because of adversarial scheduling, the exploration cost is defined as the total distance travelled by all robots to visit all points in a sphere of radius D. Since the number of robots is constant, we could as well define the exploration cost as the maximum path length travelled. Now that we have discussed this subtlety, we will abuse the terminology and use "exploration cost" and "time" interchangeably.

3 Exploration of n-dimensional Grids

A straightforward generalization of the algorithms for the exploration of 2D grids in [21] to n dimensions results in algorithms that use $\Omega(n)$ agents. Consider, for example, such a simple generalization of a randomized 2D algorithm. The basic idea of the $n+1$-agent randomized algorithm for n dimensions is to make an n-segment walk, starting from the origin, and walking the i-th segment along dimension i. The lengths of the segments are chosen randomly, and one agent per segment is used to mark its endpoint. This allows the agent to find the way back to the origin and start another random trial. In essence, this algorithm uses 2 agents per dimension to store in unary the distance travelled in this dimension, and by an appropriate arrangement we can reuse one of the agents in the successive dimension to bring the number of additional agents per dimension to 1.

The main idea of our approach is a realization that it is not necessary to use $n+1$ agents to store n numbers of segment lengths. Observe that segment lengths are stored and retrieved in this randomized algorithm in the first-in last-out order. Thus this algorithm can be realized if we can store the agent's movements in a stack. It turns out that we can use a *constant* number of agents, independent of the grid's dimension, to *implement a stack* in which the active agent, that does the exploration, stores its walk and subsequently uses to return to the origin. The active agent "carries" the stack along its walk, i.e., it always makes the agents representing the stack to shift by one position in the direction it moves before making that move itself in its walk.

3.1 The Stack Implementation

The format of data stored in the logical stack is the string $\alpha \in (0^*1)^n$, where 0 represents *continue walking in the current direction*, 1 represents *switch to the next dimension*.
The physical implementation of the stack stores this data by interpreting α^r (that is α reversed) as a binary number S and storing it in unary as a distance between two agents located in a row in the first dimension.

We employ the following agents:

- a: the *active* agent that is doing the exploration of the grid; in the semi-synchronous model this is the only agent moving around and manipulating the other agents,
- b: the *base* of the stack, from which measurements are taken, and representing the current logical location of the exploration,
- c: the *counter* agent; this is an auxiliary agent for implementing the stack operations in the semi-synchronous model,
- d: the *distance* agent; its distance from the base b stores the content of the stack in unary,
- e: the *extra* agent used in the deterministic algorithms to store an extra copy of the current stack value.

The basic stack operations we need to implement are *isEmpty()*, *push(v)* where $v \in \{0, 1\}$ and *pop()*. Operation *isEmpty()* simply returns whether b and d are collocated. Implementation of *push()* and *pop()* is model-dependent and given below.

3.1.1 Implementing Semi-Synchronous Stack

Algorithms 1 and 2 show the implementation of push and pop operations for the semi-synchronous stack. Notice that after each push/pop operation the agents b and c in these algorithms are not only collocated, but they actually return to the position they had before push/pop.

Algorithm 1 Semi-synchronous stack: push(v).

On entry: b and c collocated, a and d collocated at $b + Se_1$.
On exit: b and c collocated, a and d collocated at $b + (2S + v)e_1$.
procedure PUSH(v)
    a goes to b and brings c to d
    while b and d are not collocated **do**
        a goes to c, pushes it one step away from b and returns to d
        a pushes d one step closer to b
    end while
    d becomes c
    a goes to c and tells it to become d
    if v=1 **then**
        a pushes d one step away from b
    end if
end procedure

Algorithm 2 Semi-synchronous stack: pop().

On entry: b and c collocated, a and d collocated at $b + Se_1$.
On exit: b and c collocated, a and d collocated at $b + \lfloor S/2 \rfloor e_1$, returns $S \bmod 2 = 1$.
procedure POP
    while b and c are at distance more than 1 **do**
        a pushes d one step closer to b
        a goes to c and pushes it one step away from b
    end while
    $v = d$ is one step from c
    c and d switch roles
    a brings c to a and returns to d
    return v
end procedure

3.1.2 Implementing Synchronous Stack

In the synchronous model, we can synchronize the movements of agents to effectively multiply or divide the stack content by 2 without the need of the counter agent c, see Figure 1.

Algorithm 3 Synchronous stack: push(v).

1: On entry: a and d collocated at $b + Se_1$.
2: On exit: a and d collocated at $b + (2S + v)e_1$.
3: **procedure** PUSH(v)
4: a goes to b and then back towards d until they meet, walking at speed 1
5: d walks away from b at speed 1/3 (move, wait, wait, see Figure 1)
6: **if** v=1 **then**
7: a pushes d one step away from b
8: **end if**
9: **end procedure**

Algorithm 4 Synchronous stack: pop().

1: On entry: a and d collocated at $b + Se_1$.
2: On exit: a and d collocated at $b+\lfloor S/2 \rfloor e_1$, returns $S \bmod 2 = 1$.
3: **procedure** POP
4: a goes to b and then back towards d until they meet, walking at speed 1
5: d walks towards b at speed 1/3 (move, wait, wait, see Figure 1)
6: **if** a and d meet right after d's move **then**
7: return 1
8: **else**
9: return 0
10: **end if**
11: **end procedure**

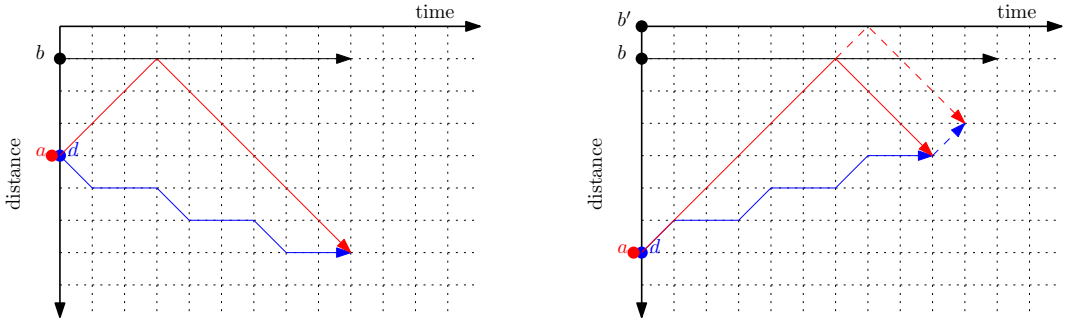


Figure 1 Multiply/divide operations by synchronous agents, shown in time-space diagrams. At left: multiplication by 2 by is done by agent a walking up to b and then down, while agent d walks down at speed 1/3. When a and d meet they have doubled the distance to b. At right: division by 2 is shown, in this case a and d walk towards b at speed 1 and 1/3 respectively; for even case a turns at b else at one node above b, and then walks back until meeting d.

3.2 The Randomized Algorithm

As already stated in the initial part of this section, the main idea of the algorithm is to use the stack to store the random choices during the walk, so that the agent can return to the origin. The agent a "carries" the stack along this walk so that the operations can be applied without the need to search for the stack.

In addition to the stack methods, it uses two new procedures. Procedure *random(p)* returns 1 with probability p, while *moveStack()* moves the whole stack one step in the direction specified. Note that since the whole stack is located on a single line, this can be accomplished by agent a walking to each of the other agents and instructing them to move one step in the specified direction.

The algorithm works in rounds, that we number $1, 2, 3, \ldots$, that correspond to the iteration numbers of the outer while loop. At the beginning of each round, the active robot picks a binary string $R \in \{-1, 1\}^n$ uniformly at random. This string indicates that the robot is

going to explore dimension i in direction R_i. Then for each dimension i from 1 to n, the active robot travels for $Z_i - 1$ steps in direction R_i, where Z_i is geometrically distributed with parameter p (to be determined later). Note that we want Z_i to represent the length of the string pushed onto the stack while moving in dimension i. Since the string pushed on the stack includes the "separator" between dimensions, we have the -1 term for the actual number of moves. We call the concatenation of all such moves over all dimensions *the logical path of the active robot*. If no treasure is found, the active robot uses the stack to retrace its logical path back to the origin by travelling $Z_n - 1$ steps in direction $-R_n e_n$ first, followed by $Z_{n-1} - 1$ steps in direction $-R_{n-1} e_{n-1}$, and so on. To compute the exploration cost of each round, we need a simple helper lemma.

Algorithm 5 Randomized Grid Exploration.

```
1:  while treasure not found do
2:      Pick a random n-bit string R ∈ {−1, 1}^n
3:      for i = 1 to n do
4:          while random(p) = 0 do
5:              push(0)
6:              moveStack(R_i e_i)
7:          end while
8:          if i < n then
9:              push(1)
10:         end if
11:     end for
12:     i = n
13:     while not empty() do
14:         while pop()= 0 do
15:             moveStack(−R_i e_i)
16:         end while
17:         i = i − 1
18:     end while
19: end while
```

▶ **Lemma 2.** *Let S be the maximal stack size during one iteration of the outer while loop of Algorithm 5. The overall cost of this iteration is $O(S^2)$ when implemented by semi-synchronous agents, and is $O(S)$ when implemented by synchronous agents.*

Proof. In the semi-synchronous model, each push() or pop() costs $O(X^2)$, where X is the actual stack size, as the active agent zig-zags between b and d. On the other hand, in the synchronous model, the cost of each operation is linear in the stack size. The cost of moving the stack is linear in both models.

As the stack size grows exponentially, and then reduces exponentially, the overall cost is determined by the cost when the stack is the largest, i.e. $O(S^2)$ and $O(S)$ for the semi-synchronous and synchronous models, respectively. ◀

Observe that during a given round the maximum size of the stack is $2^{Z_1+\cdots+Z_n}$. Thus the exploration cost of each round is at most

$$2(Z_1 + \cdots + Z_n)2^{\Theta(Z_1+\cdots+Z_n)}$$

where $2(Z_1 + \cdots + Z_n)$ is the bound on the overall length of the logical path (there and back) of the active robot, and by Lemma 2 each step of the active path costs $2^{\Theta(Z_1+\cdots+Z_n)}$, since we need to perform operations on the stack of size $2^{Z_1+\cdots+Z_n}$. Also note that

$$2(Z_1 + \cdots + Z_n)2^{\Theta(Z_1+\cdots+Z_n)} = 2^{\Theta(Z_1+\cdots+Z_n)}.$$

Let c be the constant in the Θ notation such that the exploration cost of a round is at most $2^{c(Z_1+\cdots+Z_n)}$.

For simplicity, we will assume that the active robot checks for the treasure only at the far end-point of the logical path in each round. This assumption might lead to a more pessimistic upper bound on the exploration cost than if we assumed that the active robot checks for treasure at each grid point that it visits. However, our assumption simplifies the calculations and is sufficient for our purposes.

▶ **Theorem 3.** *Algorithm 5 locates the treasure in the n-dimensional grid in finite expected time, using either 4 semi-synchronous or 3 synchronous agents.*

Proof. Consider the infinite sequence of random variables $(X_i)_{i=1}^{\infty}$, where X_i is the exploration cost of round i. Note that the X_i are independent and identically distributed. Consider the exploration cost of a particular round, e.g., X_1. Then we have $X_1 \leq 2^{c(Z_1+\cdots+Z_n)}$, where the Z_i and c are as defined above. Therefore:

$$\begin{aligned}
\mathbb{E}(X_1) &\leq \mathbb{E}\left(2^{c(Z_1+\cdots+Z_n)}\right) \\
&= \sum_{i_1=1}^{\infty}\sum_{i_2=1}^{\infty}\cdots\sum_{i_n=1}^{\infty} 2^{c(i_1+\cdots+i_n)} p^{i_1-1}(1-p)p^{i_2-1}(1-p)\cdots p^{i_n-1}(1-p) \\
&= \left(\sum_{i_1=1}^{\infty}(2^c p)^{i_1-1}2^c(1-p)\right)\left(\sum_{i_2=1}^{\infty}(2^c p)^{i_2-1}2^c(1-p)\right)\cdots\left(\sum_{i_n=1}^{\infty}(2^c p)^{i_n-1}2^c(1-p)\right) \\
&= 2^{cn}(1-p)^n\frac{1}{(1-2^c p)^n},
\end{aligned}$$

where the last step holds as long as $2^c p < 1$ that is $p < 1/2^c$.

Define a random variable T to be the minimum t such that the far end-point of X_t coincides with the treasure. That is, our exploration procedure terminates in round T, but not earlier. Suppose that the treasure is located at position $(k_1, \ldots, k_n)$ where $|k_1| + \cdots + |k_n| = D$. By the discussion immediately preceding the statement of this theorem, the probability that the treasure is found in a particular round is $\widehat{p} = 2^{-n}(1-p)^n p^{k_1}\cdots p^{k_n} = 2^{-n}(1-p)^n p^D$, where 2^{-n} is the probability of guessing correctly the signs of the k_i and $p^{k_i}(1-p)$ is the probability of travelling the correct number of steps in dimension i. Thus T is geometrically distributed with parameter $\widehat{p}$. Therefore, $\mathbb{E}(T) = 1/\widehat{p}$.

We are interested in bounding the overall exploration cost, that is, $\mathbb{E}(X_1 + \cdots + X_T)$. Since the X_i are i.i.d. and T is a stopping time, it follows by a generalization of Wald's equation [35] to stopping times that $\mathbb{E}(X_1 + \cdots + X_T) = \mathbb{E}(T)\mathbb{E}(X_1) \leq \frac{1}{\widehat{p}}2^{cn}(1-p)^n\frac{1}{(1-2^c p)^n} < \infty$. This holds as long as we choose $p < 1/2^c$. Since c is a constant, such a probabilistic coin can be implemented by a finite automaton. The statement of the theorem follows by the number of robots sufficient to implement stack operations in each of the models (synchronous vs. semi-synchronous). ◀

3.3 The Deterministic Algorithm

The main idea is to exhaustively go over all possible stack contents in increasing order, interpreting each stack as a specification of a walk. We also keep a backup of the initial stack content, and at the end of the walk we use the backup to return to the origin. The back-up stack is stored using an additional agent. The backup is needed, as reading the stack content during the walk destroys it. Note that after the outward walk, we do not logically reverse the stack; hence the return to the origin does not use the same path as the original walk. However, this is not a problem as the walks along different dimensions are commutative.

Finally, we should mention that some generated stacks do not necessarily have the correct format, some may contain too few or too many 1s. However, this is easy to handle by the algorithm: too few ones just means we walked without using all of the dimensions, which is still a perfectly valid walk. The excessive 1s are simply ignored by taking the first excessive 1 as a directive to end the walk and return to the origin.

Using essentially the same arguments as in Lemma 2 yields:

▶ **Lemma 4.** *The cost of procedure Walk is $O(S^2)$ and $O(S)$ in the semi-synchronous and synchronous models, respectively, where S is the size of the backup stack.*

▶ **Theorem 5.** *Algorithm 6 locates the treasure in the n-dimensional grid with: (a) 5 agents and the exploration cost of $O(2^{3D+4n})$ moves in the semi-synchronous model, and (b) 4 agents and the exploration cost of $O(2^{D+2n})$ in the synchronous model.*

Algorithm 6 Deterministic Grid Exploration.

```
 1: while treasure not found do
 2:     Increment the backup stack
 3:     for every n-bit string R ∈ {−1, 1}^n do
 4:         execute Walk(R, 1)
 5:         execute Walk(R, −1)
 6:     end for
 7: end while
 8:
 9: procedure Walk(R, s)
10:     Restore stack from backup
11:     i = 1
12:     while not empty() and i ≤ n do
13:         while pop()= 0 do
14:             moveStack(sR_i e_i)
15:         end while
16:         i = i + 1
17:     end while
18: end procedure
```

Proof. The number of agents and the correctness follows easily from the construction.

It remains to sum up the cost of all calls to procedure Walk. Note that each point in space uniquely specifies a valid (i.e. with precisely n 1's) stack. Hence, the valid stack for the treasure at distance D contains $D+n$ digits. Therefore, the overall cost of Algorithm 6 is $2^n \sum_{X=1}^{2^{D+n}} O(X^2) = O(2^n(2^{D+n})^3) = 2^{3D+4n}$ in the semi-synchronous model, and $O(2^{D+2n})$ in the synchronous model (the initial 2^n covers all choices for string R). ◀

4 Polynomial time solutions

While designing our exploration algorithms in the previous section, we concentrated on minimizing the number of agents used, and the resulting cost of these algorithms is exponential in the volume $V(D)$, the smallest ball containing the treasure. A natural question to ask is whether this is an unavoidable consequence of using only a constant number of agents in the exploration. In this section we show that this is not the case: a single additional agent is sufficient to bring the cost of exploration down to a polynomial in $V(D)$.

The main reason the cost of algorithms in the preceding section is exponential is the number of incorrect stack contents being considered: as D grows compared to the fixed n, ever larger proportion of stack contents does not have the correct format and they result in repeatedly reaching already explored vertices. To avoid this problem we will efficiently explore an n-dimensional cube q^n of side q centered at the origin. We use again the stack idea to trace the exploration of q^n. The logical stack content now consists of n numbers in q-ary alphabet, describing a location within this cube. However, in this case, we also need to store the scale q. As before, the stack implementation interprets the logical content as a q-ary number and stores it in unary[2]. Since q also needs to be stored on its own, this incurs the additional cost of one agent. However, this allows us to multiply and divide by q, which would not have been possible without the extra agent.

The stack is manipulated using the explicit commands: isDivisible() which checks the divisibility by q; push(0) which multiplies the stack content by q; pop() which divides the stack content by q; and increment() which increments the top of the stack.

4.1 Stack operations: semi-synchronous implementation

In addition to agents a, b and d, we use agent f to maintain the value of q by placing it at $b + qe_1$. Furthermore, two counter agents c_d and c_f are used. At the beginning of the stack operations, f and c_f are collocated, as are b and c_d, and a and d. The basic procedure is a traversal of the whole stack by agent a, manipulating the tokens according to the specific command.

In push(0) (i.e. multiplying the stack content by q), a pushes c_f towards b and c_d away from b. Whenever c_f reaches b, a transports it back to f as well as pushes d one step closer to b. The process terminates when d reaches b; subsequently c_d and d change roles. The detailed procedure is given in Algorithm 8 in [20]. .

In isDivisible(), a pushes c_f towards b and c_d towards d, until c_d reaches d. Whenever c_f arrives to b, it is transported back to f. isDivisible() returns true iff at the moment when c_d reaches d, c_f is at b (or f).

Operation pop() means dividing the stack by q. The process is essentially reverse of push() – in every iteration/traversal of the stack, c_f and d are pushed towards b. Whenever c_f reaches b, it is brought back to f and c_d is pushed away from b. When d reaches b, c_d and d exchange their roles.

The detailed pseudocode of isDivisible() and pop() are straightforward and omitted. It is easy to see that the total cost of each of the stack operations is bounded by $O(S^2)$ where S is the maximal size of the stack during the operation.

[2] This is similar to the simulation of PDAs by counter machines – see Chapter 8.5 in Hopcroft, Motwani, and Ullman text [30]; however, the details of our implementation are completely different.

4.2 Stack operations: Synchronous implementation

A straightforward application of the technique from Section 3 would need agents travelling at speed $\frac{1}{2q+1}$ (for multiply) and $\frac{q-1}{s+q}$ (for divide), which is impossible with finite state agents.

Instead, we take q to be a power of two and implement the operation of multiply, divide by q via repeated applications of multiplication by 2, division by 2, respectively. Thus in this case f is placed at distance $\log q$ from b, instead of placing it at distance of q from b. The counter c_f is used to count the number of multiplications/divisions already performed, while the counter c_d is not used at all, i.e. only agents a, b, d, f and c_f are needed. The operations of doubling and halving were already described in Section 3 and shown to take $O(S)$ time. Since these operations are performed $\log q$ times, the total time complexity of every stack operation is $O(S \log q)$.

4.3 Fast deterministic grid exploration

Our polytime deterministic grid exploration algorithm is described in Algorithm 7. Starting with $q = 2$, and for any fixed value of q, the algorithm generates and visits the addresses (n-tuples from a q-ary alphabet) in lexicographic order. Then the agent a moves to position $(-q, -q, \ldots, -q)$, doubles the value of q, and moves on to the next iteration. Agent a always drags the stack along as it performs the exploration. The procedure $explore(i)$ is a recursive procedure to generate n-tuples in lexicographic order; it is called with logical stack content an i-tuple $x0$. It then iteratively calls $explore(i+1)$ to visit the $(n-i)$-dimensional cube of side q with $(x, j, 0, \ldots, 0)$ as the origin, for j ranging from 0 to $q-1$.

Note that the algorithm as shown in Algorithm 7 is presented using recursive calls for convenience; however, i is maintained in the local state.

▶ **Theorem 6.** *Let $V(D)$ be the volume of the ball of diameter D in the n-dimensional grid. Algorithm 7 locates the treasure in the n-dimensional grid with: (a) 6 agents and the exploration cost of $O(V(D)^3)$ moves in the semi-synchronous model, and (b) 5 agents and the exploration cost of $O(V(D)^2 \log D)$ in the synchronous model.*

Proof. The number of agents and the correctness follows easily from the construction. It remains to sum up the cost of all stack operations on a stack of size S. As already described, the cost of each stack operations is $O(S^2)$ and $O(S \log q)$ in the semi-synchronous and synchronous models, respectively. The maximal stack size S is bound by q^n, which is also the number of points covered by the stack base during one iteration of the outer loop (i.e. for fixed q). This results in the overall cost of $O(S^3)$ and $O(S^2 \log q)$ in the semi-synchronous and synchronous models, respectively. As q grows exponentially, the overall cost is determined by the cost for the last value of q. Finally, it is known that $V(D) = \frac{2^n}{n!} D^n$. As $q < 4D$ (the treasure would had been found if $q \geq 2D$), we get that $S \leq (4D)^n = 2^n n! V(D)$, where n is a constant. This proves the theorem. ◀

5 Exploring 3-dimensional Grids using 4 Semi-Synchronous agents

Our algorithm for the $3D$ grids explores the *sphere* consisting of all points at distance q from the origin for $q = 1, 2, 3, \ldots$, until reaching the sphere containing the treasure. In the Manhattan metric, the points of such a sphere are located on 8 triangular faces of a regular octahedron whose edges contain $q+1$ grid vertices.

The key to our success is an algorithm for exploring one such triangle using four agents, in such a way that (i) the value of q is maintained by the distance between some of the agents while exploring a triangle, so that it can be used for the exploration of all triangles of the

octahedron, (ii) the exploration of all eight triangles can be done in a fixed order, and (iii) the value of q can be increased for the exploration of the larger sphere after the exploration of the sphere of radius q is finished.

The detailed description of this algorithm and the proof of the following theorem appear in the full arXiv version of this paper [20].

▶ **Theorem 7.** *Assume that the treasure is located in a 3D grid at distance D from the origin. Algorithm* Explore3Dgrid *finds the treasure using 4 semi-synchronous agents, with the exploration cost of $O(D^3)$. This is optimal as far as the number of semi-synchronous agents used, and up to a constant factor in the exploration cost.*

6 Additional Results

In this section, we list without proofs three additional results mentioned in Section 1. We point out the sections of the arXiv version of this paper [20] where the details are given.

▶ **Theorem 8.** *Finding a treasure at distance D in an n-dimensional grid can be achieved with $O(\sqrt{n})$ agents, exploration cost $O(D^{n+\sqrt{n}})$, and without agents venturing further than distance $2D$ away from the origin (Section 6.1 of [20]).*

▶ **Theorem 9.** *Consider 3 deterministic synchronous agents that run a protocol for exploring $\mathbb{Z}^3$. In order to visit all grid points in the ball of radius D the distance of some agent from the origin must have been $\Omega\left(D^{3/2}\right)$ (Section 6.2 of [20]) .*

▶ **Theorem 10.** *Every algorithm in this paper which assumes agents with a global compass can be extended to work with agents without it by using one additional agent in the semi-synchronous model, and two additional agents in the synchronous model (Section 7 of [20]).*

7 Conclusions and Open Questions

We studied the exploration of n-dimensional grids for $n \geq 3$ by finite state automata agents. We showed the surprising result that three randomized synchronous agents suffice to find a treasure in an n-dimensional grid for any n; this is optimal in the number of agents. Our strategy can also be implemented by four randomized semi-synchronous agents, or four deterministic synchronous agents, or five deterministic semi-synchronous agents. For the three-dimensional case, we gave a different algorithm for the deterministic semi-synchronous case that uses only 4 agents, and is optimal. Our algorithms for $n \geq 4$ require agents to travel far away from the origin, i.e., exponential in D distance away, while looking for a treasure which is located at distance D from the origin. We also considered the question of whether it is possible to design algorithms that use few agents and do not require travelling much further than distance D away from the origin in order to explore the entire ball of radius D around the origin. We answered the question positively by describing an algorithm that uses $O(\sqrt{n})$ semi-synchronous deterministic agents that never travel beyond $2D$ while exploring the ball of radius D. We also showed that 3 synchronous deterministic agents in 3 dimensions performing search, if such an algorithm exists, must travel $\Omega(D^{3/2})$ away from the origin.

Algorithm 7 Fast Deterministic Grid Exploration.

```
1: q = 2
2: push(0)
3: while treasure not found do
4:     explore(1)
5:     moveStack(−q ∑_{i=1}^{n} e_i)
6:     q = 2q
7: end while
8:
9: procedure EXPLORE(i)
10:    if i > n then
11:        return
12:    end if
13:    repeat
14:        push(0)
15:        explore(i + 1)
16:        increment()
17:        moveStack(e_i)
18:    until isDivisible()
19:    pop()
20:    moveStack(−q e_i)
21: end procedure
```

Many interesting questions about the exploration of the n-dimensional grids remain open. Is it possible for 4 deterministic semi-synchronous agents to explore an n-dimensional grid for $n \geq 4$? For $n \geq 3$, can exploration of an n-dimensional grid be achieved by 3 randomized semi-synchronous agents or deterministic synchronous agents? What is the minimum number of agents that achieve polynomial time exploration? What is the minimum number of agents such that the distance of the furthest visited node from the origin is limited to polynomial in D? Is it possible to save an agent in the case of synchronous unoriented grids?

References

1 R. A. Baeza-Yates, J. C. Culberson, and G. J. E. Rawlins. Searching with Uncertainty (Extended Abstract). In *Proceedings of the 1st Scandinavian Workshop on Algorithm Theory (SWAT 88)*, pages 176–189, 1988.

2 R. A. Baeza-Yates and R. Schott. Parallel Searching in the Plane. *Computational Geometry*, 5:143–154, 1995.

3 A. Beck. On the linear search problem. *Israel Journal of Mathematics*, 2(4):221–228, 1964.

4 A. Beck. More on the linear search problem. *Israel J. of Mathematics*, 3(2):61–70, 1965.

5 A. Beck and P. Warren. The return of the linear search problem. *Israel J. of Mathematics*, 14(2):169–183, 1973.

6 R. Bellman. An optimal search. *SIAM Review*, 5(3):274–274, 1963.

7 M. A. Bender and D. K. Slonim. The power of team exploration: two robots can learn unlabeled directed graphs. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 75–85, November 1994. doi:10.1109/SFCS.1994.365703.

8 Michael A. Bender, Antonio Fernández, Dana Ron, Amit Sahai, and Salil Vadhan. The Power of a Pebble: Exploring and Mapping Directed Graphs. *Information and Computation*, 176(1):1–21, 2002. doi:10.1006/inco.2001.3081.

9 M. Blum and W.J. Sakoda. On the capability of finite automata in 2 and 3 dimensional space. In *Proceedings of FOCS*, pages 147–161, 1977.

10 P. Bose, J.-L. De Carufel, and S. Durocher. Revisiting the Problem of Searching on a Line. In *ESA 2013, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, pages 205–216, 2013.

11 S. Brandt, K.-T. Forster, B. Richner, and R. Wattenhofer. Wireless Evacuation on m Rays with k Searchers. In *Proceedings of SIROCCO 2017, to appear*, 2017.

12 S. Brandt, J. Uitto, and R. Wattenhofer. A Tight Bound for Semi-Synchronous Collaborative Grid Exploration. In *32nd International Symposium on Distributed Computing (DISC)*, 2018.

13 P. Brass, F. Cabrera-Mora, A. Gasparri, and J. Xiao. Multirobot Tree and Graph Exploration. *IEEE Transactions on Robotics*, 27(4):707–717, August 2011. doi:10.1109/TRO.2011.2121170.

14 E. Castello, T. Yamamoto, F. d. Libera, W. Liu, A. Winfield, and Y. Nakamura. Adaptive foraging for simulated and real robotic swarms: the dynamical response threshold approach. *Swarm Intelligence*, 10(1):1–31, 2016.

15 S. Chopra and M. Egerstedt. Multi-Robot Routing for Servicing Spatio-Temporal Requests: A Musically Inspired Problem. In *IFAC Conference on analysis and design of hybrid systems*, 2012.

16 M. Chrobak, L. Gasieniec, T. Gorry, and R. Martin. Group Search on the Line. In *Proceedings of SOFSEM 2015: 41st International Conference on Current Trends in Theory and Practice of Computer Science*, pages 164–176, 2015.

17 L. Cohen, Emek Y, O. Louidor, and J. Uitto. Exploring an Infinite Space with Finite Memory Scouts. In *Proc. of the 28th SODA*, SODA '17, pages 207–224, 2017.

18 S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita. Autonomous mobile robots with lights. *Theoretical Computer Science*, 609(P1):171–184, 2016.

19 X. Deng and C.H. Papadimitriou. Exploring an unknown graph (Extended Abstract). In *Proceedings of FOCS*, 1990.

20 Stefan Dobrev, Lata Narayanan, Jaroslav Opatrny, and Denis Pankratov. Exploration of High-Dimensional Grids by Finite State Machines. *Computing Research Repository (CoRR)*, abs/1902.03693, 2019. arXiv:1902.03693.

21 Y. Emek, T. Langner, D. Stolz, J. Uitto, and R. Wattenhofer. How many ants does it take to find the food? *Theoretical Computer Science*, 608:255–267, 2015.

22 Y. Emek, T. Langner, J. Uitto, and R. Wattenhofer. Solving the ANTS problem with finite state machines. In *Proceedings of ICALP*, pages 471–482, 2014.

23 M.A. Estrada, S. Mintchev, D. Christensen, M.R. Cutkosky, and D. Floreano. Forceful Manipulation with Micro Air Vehicles. *Science Robotics*, 2018.

24 O. Feinerman, A. Korman, Z. Lotker, and J-S. Sereni. Collaborative Search in the Plane without Communication. In *Proceedings of PODC*, pages 77–86, 2013.

25 P. Flocchini, G. Prencipe, and N. Santoro. *Distributed computing by oblivious mobile robots (Synthesis Lectures on Distributed Computing Theory)*. Morgan & Claypool Publishers, 2016.

26 P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Arbitrary pattern formation by asynchronous, anonymous, oblivious robots. *Theoretical Computer Science*, 407(1–3):412–447, 2008.

27 Paola Flocchini, David Ilcinkas, Andrzej Pelc, and Nicola Santoro. Remembering Without Memory: Tree Exploration by Asynchronous Oblivious Robots. *Theoretical Computer Science*, 411(14-15):1583–1598, March 2010. doi:10.1016/j.tcs.2010.01.007.

28 Pierre Fraigniaud, David Ilcinkas, Guy Peer, Andrzej Pelc, and David Peleg. Graph Exploration by a Finite Automaton. *Theoretical Computer Science*, 345(2-3):331–344, 2005. URL: https://hal.archives-ouvertes.fr/hal-00341531.

29 S. K. Ghosh and R. Klein. Online algorithms for searching and exploration in the plane. *Computer Science Review*, 4(4):189–201, 2010.

30 John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.

31 L. Hua and E. K. P. Chong. Search on lines and graphs. In *Proceedings of the 48th IEEE Conference on Decision and Control, CDC 2009, China*, pages 5780–5785, 2009.

32 A. Jez and J. Lopuszanski. On the two-dimensional cow search problem. *Information Processing Letters*, 109(11):543–547, 2009.

33 C. Lenzen, N. A. Lynch, C. C. Newport, and T. Radeva. Trade-offs between selection complexity and performance when searching the plane without communication. In *ACM Symposium on Principles of Distributed Comp. (PODC 2014),*, pages 252–261, 2014.

34 A. López-Ortiz and G. Sweet. Parallel searching on a lattice. In *Proceedings of the 13th Canadian Conference on Computational Geometry (CCCG 2001)*, pages 125–128, 2001.

35 Abraham Wald. On Cumulative Sums of Random Variables. *Ann. Math. Statist.*, 15(3):283–296, September 1944. doi:10.1214/aoms/1177731235.

Deterministic Leader Election in Programmable Matter

Yuval Emek
Faculty of Industrial Engineering and Management, Technion – IIT, Haifa, Israel
yemek@technion.ac.il

Shay Kutten
Faculty of Industrial Engineering and Management, Technion – IIT, Haifa, Israel
kutten@ie.technion.ac.il

Ron Lavi
Faculty of Industrial Engineering and Management, Technion – IIT, Haifa, Israel
ronlavi@ie.technion.ac.il

William K. Moses Jr.[1]
Faculty of Industrial Engineering and Management, Technion – IIT, Haifa, Israel
wkmjr3@gmail.com

Abstract

Addressing a fundamental problem in programmable matter, we present the first deterministic algorithm to elect a unique leader in a system of connected amoebots assuming only that amoebots are initially contracted. Previous algorithms either used randomization, made various assumptions (shapes with no holes, or known shared chirality), or elected several co-leaders in some cases.

Some of the building blocks we introduce in constructing the algorithm are of interest by themselves, especially the procedure we present for reaching common chirality among the amoebots. Given the leader election and the chirality agreement building block, it is known that various tasks in programmable matter can be performed or improved.

The main idea of the new algorithm is the usage of the ability of the amoebots to move, which previous leader election algorithms have not used.

2012 ACM Subject Classification Theory of computation → Distributed computing models; Computing methodologies → Mobile agents; Theory of computation → Self-organization

Keywords and phrases programmable matter, geometric amoebot model, leader election

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.140

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Related Version A full version of the paper is available at `http://arxiv.org/abs/1905.00580`.

Funding *Yuval Emek*: The work of Y. Emek was supported in part by an Israeli Science Foundation grant number 1016/17.
Shay Kutten: The work of this author was supported in part by a grant from the Bi-national Science Foundation.
Ron Lavi: This research was supported by the ISF-NSFC joint research program (grant No. 2560/17).
William K. Moses Jr.: The work of this author was supported in part by a grant from the Israeli Ministry of Science.

[1] Corresponding author.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 140; pp. 140:1–140:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

The notion of programmable matter [19], and specifically amoebots [8, 7], envisions matter as composed of tiny weak robots called "particles". Multiple studies have addressed what these particles can achieve by cooperation, and how such weak entities can even cooperate. See e.g., coating of materials [12, 11, 4], bridge building [1], shape formation [9, 3, 10, 15, 5], and shape recovery [14]. An important primitive used often for coordinating such tasks is the election of a unique leader. Interestingly, all deterministic algorithms either elected multiple co-leaders in cases of symmetrical shapes of the matter, or relied on various assumptions on the particles, such as initially forming a specific shape (no holes), or initially having a common chirality.

1.1 Amoebot Model

Under the *amoebot model* [8, 7], each *particle* (an *amoebot*) occupies (alone) a different intersection (or *node*) of the lines of a triangular grid embedded in the plane, as seen in Figure 1.[2] The *degree* of a particle is the number of particles occupying neighboring nodes. A particle is either *contracted* (occupies one node) or *expanded* (occupies two neighboring nodes).

Each particle is *activated* infinitely often by an asynchronous scheduler to act. One *asynchronous round* is completed when each particle is activated at least once. The activation of a particle is atomic, i.e., it is completed before the next particle is activated. Each activation consists of 3 stages: (i) P reads the memories of adjacent particles, (ii) P performs some local computation and may update its own memory and/or the memories of its neighboring particles (sometimes, this is called "sending messages"), and (iii) P may move by either expanding or contracting.[3] Specifically, an expanded particle can contract to either one of the two nodes it occupies. While contracting out of node b, the particle can *pull* a contracted particle Q that occupies a node c that neighbors b; then Q becomes expanded and occupies both c and b. The expansion of P from node b into a neighboring node c is possible if P is contracted and c is not occupied by another contracted particle. After the expansion, P occupies both b (termed P's *tail*) and c (termed P's *head*). Suppose that P's move is an expansion into c, already occupied by a different particle Q who is expanded (Q occupies also a different node e). We then say that Q is *pushed*, Q becomes contracted and occupies only node e. For the sake of convenience, we may sometimes view the (occupied) nodes as the entities taking actions.

Each particle has constant size memory. In the *leader election (LE)* problem, each particle has one of three *LE statuses*: **C** (candidate, the initial state), **L** (leader), and **U** (unelected). and will permanently change its status to either **U** or **L** such that exactly one particle has status **L** by the end of the algorithm.

The particles are classified according to their *chirality* as *clockwise (CW) particles* or *counter-clockwise (CCW)* so that a CW (resp., CCW) particle numbers the *ports* corresponding to the 6 incident edges in (each of) the node(s) it occupies from 0 to 5 in increasing CW (resp., CCW) order; the edge from which this numbering starts is chosen arbitrarily (refer to Figure 2 for an illustration). We assume that the particle chirality classification is

[2] Because of space constraints, all figures are found in the full version.

[3] Note that P allocates memory for each of its ports and that is the memory that can be modified by adjacent particles. In other words, when P receives a message, it knows through which port the message was sent and by extension which particle sent it.

determined by a malicious adversary and that initially, a particle does not know whether its chirality (or a chirality of any other node) is CW or CCW.

The *configuration* of the particle system at any given time is comprised of the location and state of each particle; it is *contracted* if all particles are contracted. We follow the common practice (see [6]) and assume that the particles are, initially, in a contracted configuration. The algorithm terminates in a contracted configuration.

Define a graph $G(t)$, called the *shape*, induced on the grid by the nodes occupied by particles at time t and an edge connects two graph nodes if they represent two neighboring grid nodes. Following the common practice in the amoebot model literature (see [8]), it is required that the shape is connected at all times. Since the shape is a (finite) planar graph associated with a planar embedding, it partitions the plane into *faces* (see [16]), where exactly one of them is the *outer* face. The occupied nodes adjacent to the outer face are said to form the *outer boundary* of the shape. An inner face that includes at least one unoccupied node is called a *hole* in the shape (refer to Figure 3 for an example). The occupied nodes adjacent to a hole are said to form an *inner boundary*. The *length* of a boundary B, denoted L_B, is the number of nodes on B; those are *boundary nodes*, occupied by *boundary particles*. Define $L_{\max} = \max_B L_B$.

1.2 Related Work

Randomized algorithms, assuming common chirality in the initial configuration, are given in [13, 6]. A deterministic algorithm was presented in [15] to both elect a leader and obtain common chirality for the natural special case that the shape did not contain holes; multiple leaders could be elected in some cases (a constant number). They then used the leader(s) to coordinate shape transformation. The no-holes assumption is replaced in [2] by an assumption that the particles start with a common chirality. In a brief announcement, they outlined an interesting algorithm that still may end with several leaders in cases of high symmetry. The current paper adapts and uses a large part of the algorithm of [2] as a procedure. In [17], both common chirality and no-holes are assumed to elect a unique leader. That algorithm also assigns, to each particle, an identifier that is unique within a radius of k. Moreover, beside triangular grids, their algorithm can work also on the square and king grids.[4]

The type of asynchronous scheduler used affects the leader election results. Typically in the literature [13, 6, 17], the scheduler provides conflict resolution mechanisms for movement and communication such that particle activations can be analyzed sequentially, i.e., the activation of each particle is atomic. As the current paper demonstrates, it is *not* impossible to elect a leader deterministically (unless, perhaps, in models when a scheduler is allowed to schedule such particles simultaneously [2, 15]).

1.3 Technical Challenges and Ideas

Multiple ideas are combined here in order to address different cases. Consider, for example, a polygon with a hole. One approach in previous algorithms, assuming no holes, was to remove (from being candidates) boundary nodes repeatedly until only one remains. In the case of one hole (addressed by one of our subroutines), the present algorithm utilizes the ability of particles to move. Intuitively, they may move (eventually) to the center of the polygon, and the particle reaching the center first is the elected one. (Thanks to the sequential scheduler, only one can reach a certain node first).

[4] It is possible to adapt the current paper's leader election algorithm to run on king grids, but this would require some work, while it follows in a natural way for the algorithm of [17].

This, of course, requires our algorithm to perform various maneuvers, to identify the center and to make sure no additional holes remain. In particular, particles have to identify the outer boundary, move outward in order to gain a symmetric shape, and then move inward together so no additional holes are formed. Since multiple polygons may be moving at the same time, two polygons may "collide" and not manage to finish the maneuver. There, we use the idea of reset, to restart the algorithm for the new shape. We managed to upper bound the number of such resets.

Because of the existence of bridge (and semi-bridge particles) (to be defined in the next section, but intuitively particles whose removal disconnects the shape), solving for a single simple polygon is not enough. For example, consider the case that the shape is a long line (possibly connecting simple polygons). Here, we use the fact that there exists only one outer face (borrowing its detection from the algorithm of [2], with some necessary adaptations). The partial leaders of the simple polygons cooperate to define a tree that spans the simple polygons. Final leader election is then performed over the tree.

The assumption of common chirality is used throughout the paper. To remove this assumption and have particles agree on chirality, we use again the detection of the outer face. The particles on the outer boundary agree on chirality (this turned out to be easier for us than agreeing on a leader among them, using the local symmetry breaking provided by the scheduler). Then, the outer boundary particles coordinate and propagate this shared chirality to the other particles within the shape.

1.4 Our Contributions and Paper Organization

The current paper presents the first deterministic protocol that elects exactly one leader on any contracted configuration, even without assuming common chirality. For a comparison of this result to known ones, please see Table 1.

The building blocks may be of interest by themselves. The assumption of common chirality is removed last, in Section 5. Other building blocks are: maximal independent set (MIS) protocol, boundary detection, leader election on a convex polygon without sharp vertices, and leader election on a spanning tree. They are are given in Section 3. Additional definitions required to understand the paper are present in Section 2. The main protocol, $\texttt{Leader} - \texttt{Election} - \texttt{By} - \texttt{Moving}$ (before removing the common chirality assumption), is presented in Section 4 (together with some other small components, such as broadcast with termination detection, and reset).

Table 1 Table comparing the result on leader election to those of previous papers. "No holes" refers to whether the algorithm requires the graph to have no holes initially or not. "Multiple leaders" refers to whether the leader election algorithm may output multiple leaders in certain cases or always outputs a unique leader. The length of the largest boundary in the initial configuration is $L_{\max}$. The length of the outer boundary in the initial configuration is L. The number of particles in the configuration is denoted by n. The terms r and $mtree$ are unique to paper [17].

Paper	Common chirality	Randomness	No holes	Multiple leaders	Running time
[13]	Yes	Yes	No	No	$O(L_{\max})$ rounds on expectation
[6]	Yes	Yes	No	No	$O(L)$ rounds with high probability
[2]	Yes	No	No	Yes	Not analyzed in paper
[15]	No	No	Yes	Yes	$O(n)$ rounds
[17]	Yes	No	Yes	No	$2(r + mtree + 1)$ rounds
Current Paper	No	No	No	No	$O(Ln^2)$ rounds

2 Preliminaries: Shape and Boundaries

We need quite a few definitions related to the shape. A *local boundary* of a particle is an interval $i, i+1, \ldots, i+j \mod 6$ of its ports that lead to unoccupied grid nodes. Note that a contracted particle may have up to three local boundaries, each a part of some boundary of the shape. However, all three may be parts of the same boundary of the shape. Henceforth, we use only the term "boundary" even for local boundaries, when the context makes the usage clear. A *bridge particle* is a *contracted* boundary particle occupying a node b lying on i local boundaries (note that $1 \leq i \leq 3$), each of which is a part of the outer boundary, and having i occupied adjacent nodes in the grid. An example of bridge particles and semi-bridge particles (see next paragraph) with bridge edges is illustrated in Figure 4.

A *semi-bridge particle* is a *contracted* boundary particle occupying a node b lying on 2 outer boundaries, and having 3 or 4 occupied adjacent nodes in the grid. If b is occupied by a bridge or semi-bridge particle, c is an adjacent occupied node, and both sides of the edge (b, c) are on the outer boundary, then edge (b, c) is called a *bridge edge*.[5]

For a boundary node b occupied by particle P with chirality C and lying on boundary B, define b's *predecessor* node a and *successor* node c w.r.t. B and C as the previous occupied node and the next occupied node along B according to C, respectively (refer to Figure 5 for an illustration).[6] Note that node b admits such predecessor a and successor c for each boundary b lies on.

The *boundary count* of b w.r.t. B and C measures the deviation of the line segment formed by b and its successor from the line segment formed by b's predecessor and b w.r.t. B taking C into account. More formally, the boundary count of b w.r.t. B is a function of C and the angle $\angle abc$ that takes on one of the values $-1, 0, 1, 2$, or 3 (as illustrated in Figure 6).[7] Let i be the unique integer that satisfies $\angle abc = 180° - i * 60°$. Let x and y be the port numbers of b corresponding to edges (b, a) and (b, c), respectively. If $(x - y) \mod 6 = 4$, then the boundary count of b w.r.t. B is $-i$, else it is i. When the boundary referred to is clear from context, it is not mentioned when giving the boundary count for a node.

Consider an occupied node b on boundary B with boundary count w. The following definitions for b are all w.r.t. B. Node b is called a *vertex* when $w = -1, 1, 2$, or 3.[8] When $w = 2$, b is a *sharp* vertex. Vertex b is *concave* when $w = -1$ and *convex* when $w = 1$ or 2. A shape whose outer boundary vertices are all convex w.r.t. the outer boundary is a *convex polygon*. A shape consists of *two (or more) simple convex polygons sharing the same contracted semi-bridge particle(s) P (, Q, R, etc.)* when (i) P (, Q, R, etc.) has no adjacent bridge edges, (ii) the shape is disconnected by removing P (, Q, R, etc.), and (iii) all vertices other than those occupied by P (, Q, R, etc.) are convex vertices. Notice that the definition of convex polygon relates only to its outer boundary nodes. Specifically, no assumptions are made on the presence of holes within the shape.

[5] Note that a semi-bridge particle may have 3 adjacent occupied nodes and lie on 2 outer boundaries and 1 inner boundary. In this case, the particle still has 1 bridge edge.

[6] Throughout, we use w.r.t. to abbreviate "with respect to".

[7] Note that it is not possible to have a node with boundary count -2 or -3 w.r.t. some boundary. Also note that the boundary count and its application to calculating the count of a segment, defined and used in the full version, is similar to how [2] uses vertex labeling in deciding the count of a segment in their paper. The actual measurement of the boundary count is similar to how [13] measures the angles between the direction a token enters and exits an agent.

[8] Notice that the angle bisector of a vertex with boundary count 1 or -1 overlaps with a line of the triangular grid.

3 Building Blocks

Let us now present the four building blocks in brief, except Subsection 3.3 that is more detailed. The full version contains all missing details. The description uses some additional definitions. Each boundary particle P maintains a binary flag *seg_head* in each boundary node b that P occupies w.r.t. each boundary that b lies on. When P occupies a boundary node and has $seg\_head = true$ for that boundary, we say P is a *seg-head* for that boundary. Consider two seg-heads P_1 and P_2 on boundary B, occupying nodes b_1 and b_2 with predecessor nodes c_1 and c_2 and successor nodes d_1 and d_2 w.r.t. B, respectively. P_2 is the *previous (resp., next) seg-head* before (resp., after) P_1 iff the particles in successor (resp., predecessor) nodes from b_2 to b_1 w.r.t. B (excluding b_1 and b_2) have *seg_head* set to $false$.

Let P_2 be the next seg-head after P_1 w.r.t. B. P_1's *segment* is the sequence of successor nodes from b_1 to c_2 with *head* b_1 and *tail* c_2. It is said that P_1's segment is *before* P_2's segment or P_2's segment is *after* P_1's segment w.r.t. B. For the sake of convenience, when referring to a procedure/action initiated by the head of a segment involving the participation of the particles in that segment, we just say that a segment runs the procedure/performs the action. It is important to note that a particle P may participate in multiple segments simultaneously (one per boundary P lies on). The algorithm needs to be careful to prevent contradicting actions of such segments (for example, preventing one segment from expanding P into one node while another segment is trying to expand P into a different node).

3.1 MIS Selection

To perform Procedure `MIS − Selection`, P joins the MIS iff no neighbor of P has yet joined the MIS. The following trivial observation breaks with impossibility results in other models when the scheduler is not asynchronous.[9]

▶ **Observation 1.** *When run by particles, procedure* `MIS − Selection` *deterministically computes an MIS in one round.*

3.2 Boundary Detection

`Boundary − Detection` is a parameterized procedure run by boundary nodes with common chirality to tell each such node b, for each boundary B that b lies on, whether B is an inner or outer boundary. This procedure is a modification of the first phase of the algorithm presented in [2], specifically adapting their subroutine `StretchExpansion` to handle (1) inner boundaries and (2) an edge case that may not be needed (and is not addressed) in [2] but is needed here (see Figure 7).[10] These adaptations result in subroutines `Inner − Stretch − Expansion` and `Outer − Stretch − Expansion`, respectively. Due to space constraints, the entire modified boundary detection procedure and proof of the theorem are presented in the full version.

▶ **Theorem 1.** *When executed by contracted boundary particles, procedure* `Boundary − Detection` *terminates in $O(L_{\max}^2)$ rounds resulting in each boundary node b knowing, for each boundary B it is on, whether B is an inner or outer boundary. If b has $seg\_head = true$ w.r.t. boundary B, then b knows how many nodes k, $k \in \{1, 2, 3, 6\}$, are also segment heads w.r.t. B.*

[9] In particular, this procedure selects a leader in a ring of 3 particles.
[10] Recall that [2] is a brief announcement, so this edge case may be handled in the full version of their paper.

3.3 Leader Election on a Convex Polygon without Sharp Vertices

Procedure $\texttt{Convex}-\texttt{Polygon}-\texttt{Leader}-\texttt{Election}$ relies on 3 subroutines, described below. Note that the outer boundary nodes of a convex polygon without sharp vertices form a hexagon in the grid. Let b be a vertex, occupied by particle P, with successor node d w.r.t. the outer boundary. Define *P's side* as the side of the hexagon containing b and d.

Let LSLS stand for largest same length sides and SSLS stand for smallest same length sides. Every possible hexagon is isomorphic to one of the following four. See Figures 8-13.

1. Category 1: The hexagon has exactly either 1 LSLS or 1 SSLS.
2. Category 2: The hexagon has either 2 LSLS and 4 SSLS, 4 LSLS and 2 SSLS, or 2 LSLS, 2 SSLS, and 2 other same length sides.
3. Category 3: The hexagon has exactly 3 LSLS and 3 SSLS.
4. Category 4: The hexagon has 6 sides of the same length.

Let $P_0, P_1, \ldots P_{k-1}$ be the seg-heads on the outer boundary such that $P_{(i+1) \mod k}$ is the next seg-head after P_i. Subroutine $\texttt{Compare}-\texttt{Length}(\texttt{x})$ is initiated by a seg-head P_i to compare the length of P_i's segment with that of $P_{(i+x) \mod k}$'s segment. The procedure simulates the way a Turing machine would perform a similar task, where the segments would be segments of the machine's tape. Since this is a known method, the details are omitted (refer to [15] for an example). The proof of the following lemma can be found in the full version.

▶ **Lemma 2.** *Let x be a constant and assume that there are L nodes on the outer boundary. If the nodes from P_i's segment's head to $P_{(i+x) \mod k}$'s segment's tail run* $\texttt{Compare}-\texttt{Length}(\texttt{x})$*, then the subroutine terminates in $O(L^2)$ rounds, resulting in P_i knowing the size comparison between the two segments.*

Consider two parallel lines M_1 and M_2 of the grid. The *mid-line(s)* between M_1 and M_2 is the line(s) parallel to both M_1 and M_2 which is either equidistant from both M_1 and M_2 or not closer to one of the lines by more than a unit distance. Consider a category 2 hexagon where opposite outer boundary vertices b_1 and b_2, occupied by particles P_1 and P_2 respectively, have $seg\_head = true$ and the remaining nodes have $seg\_head = false$. There exist either 1 or 2 mid-lines between P_1's side and P_2's side. Let c_1 and c_2, occupied by particles Q_1 and Q_2 respectively, be nodes on P_1 and P_2's segments respectively lying on the mid-line (or on the closer mid-line to the head of the segment in the case of 2 mid-lines). The outer boundary particles run subroutine $\texttt{Mid}-\texttt{Line}$ to find c_1 and c_2 and subsequently Q_1 and Q_2 set $seg\_head = true$ and P_1 and P_2 set $seg\_head = false$. See Figures 10 and 11 for examples. A more detailed description of the subroutine is found in the full version. The following observation captures the running time of $\texttt{Mid}-\texttt{Line}$.

▶ **Observation 2.** *Let there be L outer boundary nodes on a category 2 hexagon with opposite vertices b_1 and b_2, occupied by particles P_1 and P_2 respectively, with $seg\_head = true$ and remaining nodes with $seg\_head = false$. Subroutine* $\texttt{Mid}-\texttt{Line}$*, run by the L nodes, terminates in $O(L^2)$ rounds, such that nodes c_1 and c_2, which are the closest nodes in P_1 and P_2's segments lying on mid-lines between P_1's side and P_2's side respectively, now have $seg\_head = true$ and b_1 and b_2 have $seg\_head = false$.*

Intuitively, when the segment heads are on the mid-line as promised by Observation 2, if they move towards the center, they can get next to each other and elect one of them as a leader. The following subroutine $\texttt{Snake}-\texttt{Movement}(\texttt{D},\texttt{x})$ (described in more detail in the full version), is used for election in hexagons of several types. Consider a path p of w nodes occupied by contracted particles with head node b occupied by particle P and tail

node c. P has $seg\_head = true$ w.r.t. the outer boundary and the remaining particles have $seg\_head = false$. See Figure 14 for an example. Particles in p (termed *snake* p) expand, so p becomes longer and its head P moves in direction D for distance $x \leq w$ (without breaking connectivity and while keeping the tail node of p fixed at c). P is the one expanding first, and the particles perform a sequence of expansions and contractions until reaching the desired total length of p. Note that x may not be a constant. Hence, this value is represented distributively on the particles of p by them simulating a tape of a Turing machine. (The value of x is also input to the subroutine and used for the computing in the same manner).

Note that only a segment on the outer boundary can perform this procedure, hence, snake p will not belong to two different segments giving it contradictory instructions to move. One thing that may happen is that the head of snake p reaches a particle Q not in snake p. If Q belongs to another snake p' then p stops. Otherwise, p continues moving in direction D simply by annexing Q who now becomes the head of the snake (that we still call snake p). The proof sketch of the following lemma can be found in the full version.

▶ **Lemma 3.** *Assume that L contracted particles of a snake run* `Snake − Movement(D, x)`*, where $x \leq L$. Then, the subroutine terminates in $O(x^2)$ rounds without breaking connectivity. On termination, either the snake head reached a node at distance x away from the head of the snake in direction D, or the next particle in direction D belongs to another snake.*

Now, procedure `Convex − Polygon − Leader − Election` is described. Note that illustrations expanding on the description are found in the full version. Initially, the 6 particles that occupy vertices on the outer boundary set $seg\_head = true$ while the remaining particles in the polygon set $seg\_head = false$. Each of these 6 particles initiates `Compare − Length(x)` for $1 \leq x \leq 6$, sends messages to the remaining 5 particles with the results of these comparisons, and determines which category hexagon it lies on.[11] The procedure follows one of the following four cases:

1. *Category 1 hexagon:* This case is trivial - the particle at the head of the smallest or largest side becomes the leader. See Figures 8 and 9.
2. *Category 2 hexagon:* If there are exactly 2 LSLS, then those sides' polygon vertices keep $seg\_head = true$ and the remaining vertices set $seg\_head = false$. Else there are 2 SSLS whose vertices keep $seg\_head = true$ while others set $seg\_head = false$. Call particles occupying vertices with $seg\_head = true$, P_1 and P_2, and denote the direction from the successor of P_1 to P_1 as D_1 (similarly denote D_2). Now, P_1 and P_2 initiate `Mid − Line` resulting in two new particles Q_1 and Q_2 setting $seg\_head = true$ and P_1 and P_2 setting $seg\_head = false$ (refer to Figures 10 and 11 for examples).
   The resulting segments of Q_1 and Q_2 form snakes p_1 and p_2 with lengths w_1 and w_2 respectively that run `Snake − Movement(D1, w1)` and `Snake − Movement(D2, w2)` in directions D_1 and D_2, respectively. In addition to the usual termination conditions when running `Snake − Movement(D1, w1)` and `Snake − Movement(D2, w2)`, the subroutines also terminate when the head of p_1 is adjacent to that of p_2. Then, the two heads run `MIS − Selection` and the particle that joins the MIS becomes the leader.
3. *Category 3 hexagon:* Let P_1, P_2, and P_3 occupy vertices b_1, b_2, and b_3 such that P_1, P_2, and P_3's sides are the 3 largest sides. The remaining particles set $seg\_head = false$. See Figure 12. Let D_1, D_2, and D_3 be the directions along the angle bisectors of b_1, b_2, and b_3 respectively toward the center of the hexagon. The two phase procedure followed by P_1's segment is now described. (P_2's and P_3's segments act similarly).

[11] With this information, a particle can compute, using a constant amount of space, the total order on the lengths of sides of the hexagon. Combined with information of which sides are equal in length, a particle can determine both the category of the hexagon it lies on and the type of its own side.

Notice that P_1's segment encompasses 1 SSLS and 1 LSLS with lengths x and y respectively. In phase 1 (simulating a Turing machine), the values of $f = \lfloor (y-x)/3 \rfloor$, $g = x + f$, and $q = (y-x) \mod 3$ are computed and stored in P_1's segment. If $q = 2$, g is incremented by 1. Now P_1 sends a message to the particle Q_1 located f nodes from the head of the segment, telling Q_1 to set $seg\_head = true$ and store D_1, f, g, and q. P_1 subsequently sets $seg\_head = false$. Q_1 is now the head of a segment. Similarly, some Q_2, Q_3 replace P_2, P_3 as heads of their segments. Now Q_1 sends a message along the outer boundary to Q_2 and Q_3 indicating that the first phase is over. Once Q_1 receives similar messages from Q_2 and Q_3, the second phase begins.

In phase two, Q_1's segment runs $\texttt{Snake} - \texttt{Movement}(\texttt{D}_1, \texttt{g})$. If $q = 0$, all three snakes move towards the same final node b. Let p be the snake such that its head particle R is the first to occupy b. R waits until the remaining two snakes reach it and then becomes the leader. If $q \neq 0$, the final nodes occupied by the heads of the three snakes form a triangle. Let R be a head of the snake that occupies one of the triangle's nodes. R waits until the other two triangle's nodes are occupied and then runs $\texttt{MIS} - \texttt{Selection}$. The particle chosen to be in the MIS becomes the leader.

4. *Category 4 hexagon:* All vertices have $seg\_head = true$ (e.g., Figure 13). The procedure here is a simplified version of the case of Category 3 hexagon. See the full version.

▶ **Theorem 4.** *Procedure* $\texttt{Convex} - \texttt{Polygon} - \texttt{Leader} - \texttt{Election}$ *run by contracted particles of a convex polygon without sharp edges results in exactly one leader being elected deterministically in* $O(L^2)$ *rounds, where* L *is the number of particles on the outer boundary.*

Proof Sketch. The readers can convince themselves that all types of hexagons have been accounted for in the four hexagon categories. Let us prove correctness for each category separately. The case of category 1 is trivial.

In a category 2 hexagon, Observation 2 guarantees that particles are chosen such that they lie on the same mid-line or adjacent mid-lines. The distance needed to be traveled by each segment until both heads are adjacent is $\leq L/2$. Since the segments divide the nodes of the outer boundary equally, each segment has enough contracted particles such that it is possible to traverse this distance by expanding every particle in the segment. Moreover, no two snakes can block each other before reaching that distance. $\texttt{MIS} - \texttt{Selection}$ is guaranteed to choose exactly one leader due to Observation 1.

For category 3, inscribe the hexagon in an equilateral triangle with vertices A, B, and D, centroid C, and F trisecting $\overline{AB}$, as seen in Figure 15. Observe that each side of the triangle is of length $2x + y$ and $|\overline{AF}| = |\overline{FC}|$. When $(y-x) \mod 3 = 0$, $\overline{FC}$ coincides with a grid line and all segments move towards C using `Snake-Movement()`. However, if $(y-x) \mod 3 \neq 0$, the segments move to nodes that form a triangle around the centroid, in which case, $\texttt{MIS} - \texttt{Selection}$ is run and Observation 1 guarantees a leader is selected. Note that no two snakes can block each other before reaching the meeting point.

The proof for category 4 is a simplified version of the proof for Category 3. Thus for all four types of hexagons, a leader is chosen. The running time is analyzed in the full version. ◀

3.4 Leader Election on a Spanning Tree

Procedure $\texttt{Spanning} - \texttt{Tree} - \texttt{Leader} - \texttt{Election}$ deterministically elects a unique leader when participating particles form a spanning tree and have common chirality. Note that this can also be performed by the algorithms of [15, 17]. We defer the description to the full version and give the following theorem without a proof.

▶ **Theorem 5.** *Procedure* `Spanning − Tree − Leader − Election` *run by particles forming a spanning tree of diameter x results in exactly one leader being elected deterministically in $O(x)$ rounds.*

4 Leader Election

An overview of deterministic algorithm `Leader − Election − By − Moving` for electing a unique leader is now given, assuming common chirality (an assumption removed later). Additional details and proofs appear in the full version.

The initial contracted configuration of n particles forms a connected shape $G(0)$ at the beginning of round 0 with all particles having status **C**. $H(t), K(t), F_1(t)$, and $F_2(t)$ are virtual graphs at the beginning of round t that are maintained by the particles distributively and are initially empty.[12] Note that the round number is subsequently dropped, as it is apparent from context.

The algorithm has six phases. Graph H is used throughout the algorithm for various purposes depending on the phase of the algorithm. Graph K is a subgraph of G that holds a spanning tree of all particles and is important for phase 6 of the algorithm. Graph F_1 is a forest of trees of all particles used throughout the algorithm. Graph F_2 is a forest of trees of a subset of the particles used only in phase 5 of the algorithm.

Each particle P maintains a phase counter in $[1 \dots 6]$ and appends its value to each message sent. If P receives a message from another particle Q in a different phase, P does not process Q's message until P is in the same phase as Q.[13]

1. *Initialization:* At the end of this phase, every particle is contracted and each boundary particle has identified the type (inner/outer) of each of its boundaries. Furthermore, graph H consists of a set of simple convex polygons, where two simple polygons may share the same semi-bridge particle.
   Each boundary particle runs `Boundary − Detection` for each boundary B it lies on to determine whether B is an inner or outer boundary. Once `Boundary − Detection` terminates, all particles not on the outer boundary set $seg\_head = false$. Thus, there are k, $k \in \{1, 2, 3, 6\}$, particles with $seg\_head = true$ located on the outer boundary. Call these seg-heads $P_1, P_2, \dots, P_k$. If $k = 1$, change P_1's status to **L** and broadcast (by simple flooding [18]) a $final\_terminate$ message to other particles to terminate the algorithm and change their statuses to **U**.
   Each particle that is not a bridge or semi-bridge adds itself and its edges to adjacent nodes to H. Otherwise, semi-bridge particles add themselves and their non-bridge edges to H. Note that all particles are contracted at the end of this phase.
2. *Spanning forest formation:* Each outer boundary node a becomes the root of a tree T and uses the standard broadcast-&-echo method [18] to recruit nodes to its tree. Each node b joins exactly one tree T. Termination detection of the phase is coordinated by seg-heads P_1 to P_k, after all the broadcast-&-echoes terminate. Thus a spanning forest F_1 of trees is formed with outer boundary particles as roots of the trees. Furthermore, each node knows its parent and children in the tree. Note that all particles remain contracted during the phase.

[12] For each graph, each particle maintains locally its own edges in the graph and whether it is in the graph or not. Each particle allots a constant amount of memory for each of the graphs G, H, K, F_1, F_2 and updates them as necessary when activated.

[13] It is trivial to return to a contracted configuration from the configuration the algorithm terminates in, so this "7th phase" is not described. Informally, it consists of particles that performed `Snake − Movement`() as part of `Convex − Polygon − Leader − Election` during phase 5 reversing their movements.

3. *Convexification:* The subgraph H, induced by removing bridge particles from the shape, is a collection of polygons. Each outer boundary particle P w.r.t. H that is a concave vertex and not a semi-bridge particle expands towards the outer boundary along P's angle bisector while coordinating the pulling of P's tree with it. P occupying node b and moving to node c completes one step of convexification when it has moved to node c and all particles in the tree rooted at P in F_1 are back in a contracted state. Convexification is performed repeatedly by particles until no more steps of convexification are possible (refer to Figure 16 for an example).

   At the same time, each seg-head P_i $(1 \leq i \leq k)$ continuously checks its segment for any concave vertices in H. If none are found, the k seg-heads coordinate to terminate this phase. All particles previously in H update their edges in H to reflect current connections to other particles. Bridge and semi-bridge particles add themselves and their bridge edges to virtual graph K. Note that all particles are contracted at the end of this phase.

   The phase as described so far may be stopped before convexification completes if two types of situations arise. *Type 1*: during the movement outward, an outer boundary particle that moved in some direction D to node b in one step of convexification finds out that the node adjacent to b in direction D is occupied. *Type 2*: a semi-bridge particle, bridge particle, or outer boundary particle stops being one. Both types reflect a change in the particles occupying the outer boundary, possibly resulting in particles previously with $seg\_head = true$ no longer lying on the outer boundary. The algorithm then resets to phase 1, as described in the full version (the reset procedure also makes sure that all the particles are reset to a contracted state).

4. *De-sharpification:* In this phase, certain particles remove themselves from H recursively until only convex polygons and two-node lines remain in H. Consider a particle P in H. If P is not a semi-bridge particle and is a sharp vertex w.r.t. the outer boundary in H, then P removes itself from H. If P is a semi-bridge particle and its occupied adjacent nodes are located at ports $x, x+1, x+3$, and $x+4$ (mod 6) for some positive integer value of x, then P removes itself from H.

   At the same time, each seg-head P_i $(1 \leq i \leq k)$ checks its segment continuously for any sharp vertices in H. If none are found, P_i coordinates with the other $k-1$ seg-heads to terminate the phase. The induced subgraph H at the end of the phase is a set containing just two types of polygons: (a) lines consisting of 2 nodes as well as (2) convex polygons without sharp vertices. Note that all the particles remain contracted at the end of this phase.

5. *Leader election on individual polygons and spanning tree formation:* This phase consists of two stages. In stage one, each convex polygon and each line in H elects a unique polygon leader using $\texttt{Convex} - \texttt{Polygon} - \texttt{Leader} - \texttt{Election}$ and $\texttt{MIS} - \texttt{Selection}$, respectively. In stage two, each particle P chosen as a polygon leader in stage one, acts as a root and forms a tree that spans its connected component of $G \setminus K$. The nodes in K that are reachable from P over $G \setminus K$ are leaves of P's tree. Call this forest of polygon leaders rooted trees F_2.

   The termination condition is somewhat long to describe; see the full version for details. Very informally - the polygons are connected by semi-bridge particles. Hence, when a polygon leader finished constructing its tree over the polygon, the semi-bridge particle(s) is notified. The seg-head particles P_i $(1 \leq i \leq k)$ check continuously the semi-bridge particles to know when the construction of F_2 is done.

   At the end of this phase, K is updated to contain all particles in graph G with edges restricted to bridge edges and edges of F_2. It is shown later in a lemma that K now forms a tree, spanning all the candidates (status **C** particles).

6. *Leader election on a spanning tree:* Each particle participates in $\texttt{Spanning}-\texttt{Tree}-\texttt{Leader}-\texttt{Election}$ on the graph K. Once a particle P changes its status to $\mathbf{L}$, P broadcasts a $final\_terminate$ message by flooding along K. This results in one particle, the leader, having status $\mathbf{L}$ and the remaining particles having status $\mathbf{U}$ when the algorithm terminates.

The following lemmas apply to the algorithm, with proofs deferred to the full version.

▶ **Lemma 6.** *Phase 1 terminates in $O(L_m^2)$ rounds, where L_m is the length of the largest boundary of the shape, resulting in each boundary particle knowing what type each of its boundaries is and k, $k \in \{1, 2, 3, 6\}$, particles, $P_1, P_2, \ldots, P_k$, lying on the outer boundary with $seg\_head = true$. Furthermore, at the end of the phase, H consists of a set of simple convex polygons, where two simple polygons may share the same semi-bridge particle. If $k = 1$, the algorithm terminates with one particle as leader in an additional $O(n)$ rounds.*

▶ **Lemma 7.** *Phase 2 terminates in $O(n)$ rounds, resulting in a disjoint forest of trees F_1 covering every particle.*

▶ **Lemma 8.** *Phase 3 terminates in $O(Ln)$ rounds, resulting in either a reset or a graph H containing a set of simple convex polygons, where two simple polygons may share the same semi-bridge particle.*

▶ **Lemma 9.** *There can be at most L resets occurring in phase 3, where L is the length of the outer boundary of the original shape.*

▶ **Lemma 10.** *Phase 4 takes $O(n)$ rounds to complete, resulting in H containing only a set of lines consisting of 2 nodes and convex polygons without sharp vertices.*

▶ **Lemma 11.** *Phase 5 terminates in $O(L^2+n)$ rounds resulting in K containing all particles and forming a spanning tree.*

▶ **Lemma 12.** *Phase 6 terminates in $O(n)$ rounds resulting in a unique leader with status $\boldsymbol{L}$ being chosen and all other nodes having status $\boldsymbol{U}$.*

Combining the above lemmas together, we get the desired results.

▶ **Theorem 13.** *Algorithm* $\texttt{Leader}-\texttt{Election}-\texttt{By}-\texttt{Moving}$*, run by n particles in a contracted configuration, elects a unique leader deterministically and terminates in $O(Ln^2)$ rounds, where L is the number of particles on the outer boundary of the original shape.*

Proof Sketch. From Lemmas 6, 7, and 8, the combined running time of one iteration of phases 1 to 3 is $O(n^2)$ rounds since $L_m = O(n)$. There can be at most $O(L)$ iterations of phases 1 to 3, by Lemma 9. Adding in the running times of phases 4 to 6 from Lemmas 10, 11, and 12, it is clear that the total running time of the algorithm is $O(Ln^2)$ rounds.

The correctness directly follows from Lemma 12. ◀

5 Chirality Agreement

In this section, procedure $\texttt{Chirality}-\texttt{Agreement}$ is described. Consider n contracted particles forming a connected shape with the length of maximum boundary being $L_{\max}$. The particles run $\texttt{Chirality}-\texttt{Agreement}$ and terminate in $O(L_{\max}^2 + n)$ rounds, resulting in all particles agreeing on the same chirality and forming the original shape.

Informally, after the boundary particles identify their boundaries, they first agree on chirality separately for each boundary they lie on. This is easier than leader election given the local symmetry breaking built into the model (atomicity of the scheduler). This is enough to allow the particles to identify the outer boundary similarly to the way it was done for the leader election. Finally, the chirality agreed upon for the outer boundary becomes the chirality of everyone. A detailed explanation of the procedure along with the proof of the following theorem can be found in the full version.

▶ **Theorem 14.** *Procedure* `Chirality − Agreement`*, run by n contracted particles forming a connected shape, terminates in $O(L_{\max}^2)$ rounds, where $L_{\max}$ is the length of the largest boundary in the shape, resulting in all particles having common chirality and retaining the original shape.*

6 Conclusion and Future Work

The results of this paper leave several lines of research open. First, the algorithms here require the particles to move for leader election and for chirality agreement. Is it possible to solve either problem deterministically in the given setting without requiring particles to move? Second, can one reduce the running time or provide a matching lower bound?

References

1. Marta Andrés Arroyo, Sarah Cannon, Joshua J. Daymude, Dana Randall, and Andréa W. Richa. A stochastic approach to shortcut bridging in programmable matter. *Natural Computing*, 17(4):723–741, 2018.
2. Rida A. Bazzi and Joseph L. Briones. Brief Announcement: Deterministic Leader Election in Self-organizing Particle Systems. In *International Symposium on Stabilizing, Safety, and Security of Distributed Systems*, pages 381–386. Springer, 2018.
3. Sarah Cannon, Joshua J. Daymude, Dana Randall, and Andréa W. Richa. A Markov chain algorithm for compression in self-organizing particle systems. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 279–288. ACM, 2016.
4. Joshua J. Daymude, Zahra Derakhshandeh, Robert Gmyr, Alexandra Porter, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. On the runtime of universal coating for programmable matter. *Natural Computing*, 17(1):81–96, 2018.
5. Joshua J. Daymude, Robert Gmyr, Kristian Hinnenthal, Irina Kostitsyna, Christian Scheideler, and Andréa W. Richa. Convex Hull Formation for Programmable Matter. *arXiv preprint*, 2018. `arXiv:1805.06149`.
6. Joshua J. Daymude, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Improved leader election for self-organizing programmable matter. In *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, pages 127–140. Springer, 2017.
7. Joshua J. Daymude, Kristian Hinnenthal, Andréa W. Richa, and Christian Scheideler. Computing by Programmable Particles. In *Distributed Computing by Mobile Entities*, pages 615–681. Springer, 2019.
8. Zahra Derakhshandeh, Shlomi Dolev, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Brief announcement: amoebot–a new model for programmable matter. In *Proceedings of the 26th ACM symposium on Parallelism in algorithms and architectures*, pages 220–222. ACM, 2014.
9. Zahra Derakhshandeh, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. An algorithmic framework for shape formation problems in self-organizing particle systems. In *Proceedings of the Second Annual International Conference on Nanoscale Computing and Communication*, page 21. ACM, 2015.

10 Zahra Derakhshandeh, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Universal shape formation for programmable matter. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 289–299. ACM, 2016.

11 Zahra Derakhshandeh, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Universal coating for programmable matter. *Theoretical Computer Science*, 671:56–68, 2017.

12 Zahra Derakhshandeh, Robert Gmyr, Andréa W. Richa, Christian Scheideler, Thim Strothmann, and Shimrit Tzur-David. Infinite object coating in the amoebot model. *arXiv preprint*, 2014. `arXiv:1411.2356`.

13 Zahra Derakhshandeh, Robert Gmyr, Thim Strothmann, Rida Bazzi, Andréa W. Richa, and Christian Scheideler. Leader election and shape formation with self-organizing programmable matter. In *International Workshop on DNA-Based Computers*, pages 117–132. Springer, 2015.

14 Giuseppe Antonio Di Luna, Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Giovanni Viglietta. Line recovery by programmable particles. In *19th International Conference on Distributed Computing and Networking, ICDCN 2018*, volume 133180, pages 1–10. Association for Computing Machinery, 2018.

15 Giuseppe Antonio Di Luna, Paola Flocchini, Nicola Santoro, Giovanni Viglietta, and Yukiko Yamauchi. Shape Formation by Programmable Particles. In *21st International Conference on Principles of Distributed Systems*, 2017.

16 Reinhard Diestel. Graph theory. 2005. *Grad. Texts in Math*, 101, 2005.

17 Nicolas Gastineau, Wahabou Abdou, Nader Mbarek, and Olivier Togni. Distributed leader election and computation of local identifiers for programmable matter. *arXiv preprint*, 2018. `arXiv:1807.10461`.

18 Adrian Segall. Distributed network protocols. *IEEE transactions on Information Theory*, 29(1):23–35, 1983.

19 Tommaso Toffoli and Norman Margolus. Programmable matter: concepts and realization. *Physica D: Nonlinear Phenomena*, 47(1-2):263–272, 1991.

Two Moves per Time Step Make a Difference

Thomas Erlebach
Department of Informatics, University of Leicester, Leicester, England
te17@leicester.ac.uk

Frank Kammer
THM, University of Applied Sciences Mittelhessen, Giessen, Germany
frank.kammer@mni.thm.de

Kelin Luo
School of Management, Xi'an Jiaotong University, Xianning West Road, Xi'an, China
luokelin@stu.xjtu.edu.cn

Andrej Sajenko
THM, University of Applied Sciences Mittelhessen, Giessen, Germany
andrej.sajenko@mni.thm.de

Jakob T. Spooner
Department of Informatics, University of Leicester, Leicester, England
jts21@leicester.ac.uk

Abstract

A temporal graph is a graph whose edge set can change over time. We only require that the edge set in each time step forms a connected graph. The temporal exploration problem asks for a temporal walk that starts at a given vertex, moves over at most one edge in each time step, visits all vertices, and reaches the last unvisited vertex as early as possible. We show in this paper that every temporal graph with n vertices can be explored in $O(n^{1.75})$ time steps provided that either the degree of the graph is bounded in each step or the temporal walk is allowed to make two moves per step. This result is interesting because it breaks the lower bound of $\Omega(n^2)$ steps that holds for the worst-case exploration time if only one move per time step is allowed and the graph in each step can have arbitrary degree. We complement this main result by a logarithmic inapproximability result and a proof that for sparse temporal graphs (i.e., temporal graphs with $O(n)$ edges in the underlying graph) making $O(1)$ moves per time step can improve the worst-case exploration time at most by a constant factor.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases Temporal Graph Exploration, Algorithmic Graph Theory, NP-Complete Problem

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.141

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Funding *Kelin Luo*: This work was partially supported by the China Postdoctoral Science Foundation (Grant No. 2016M592811), and the China Scholarship Council (Grant No. 201706280058).
Andrej Sajenko: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 379157101.

Acknowledgements We would like to thank an anonymous reviewer for a helpful suggestion regarding the presentation of the proof of one of the lemmas.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 141; pp. 141:1–141:14

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

1 Introduction

Temporal graphs, or time-varying graphs, are graphs whose edge set changes over time. Due to the prevalence of dynamic networks whose links change over time in many application settings (e.g., wireless mobile networks, transportation networks, social networks), the study of algorithmic aspects of temporal graphs has received increasing attention recently [14]. A temporal graph $\mathcal{G}$ with lifetime τ is a sequence of graphs $G_i = (V, E_i)$, for $i = 1, \ldots, \tau$, that all have the same vertex set, but possibly different edge sets. A particular problem of interest is the *temporal exploration problem* (TEXP) where an agent starts at a given vertex $s \in V$ and aims to visit all vertices in V as quickly as possible (i.e., minimizing the time step in which the last unvisited vertex is reached) while making one move (either stay at the current vertex or move to a neighboring vertex in the current graph) in each time step. The existence of such an exploration is guaranteed if the graph G_i in each time step is a connected graph [13], and so it has become customary to study TEXP for temporal graphs with this property [6, 7, 13]. We make the same assumption throughout this paper. The number of vertices of the temporal graph under consideration is denoted by n.

It is known that every temporal graph (under the assumption that the graph in each step is connected) can be explored in $O(n^2)$ steps and that there are temporal graphs that require $\Omega(n^2)$ steps to be explored [6]. The construction of temporal graphs that require $\Omega(n^2)$ steps for an exploration from [6] produces temporal graphs for which the graph in each step has a vertex of high degree (the graph in each step is a star, so one vertex has degree $n-1$) and in which the edge set changes in every step (the center of the star changes in each step). This poses the natural question whether a better upper bound on the worst-case exploration time holds if any of these properties is avoided, i.e., if the graph in each step has bounded degree or if the edge set of the graph changes only in every other step (which is, up to a factor of two, equivalent to allowing the agent to make two moves instead of one in each step). A first step towards answering this question was made in [7], where it was shown that $O(\log \Delta \cdot \frac{n^2}{\log n})$ steps[1] suffice for the exploration if the graph in each step has maximum degree at most Δ. For constant Δ, this proves that $O(\frac{n^2}{\log n})$ steps suffice.

In this paper, we present a substantial further improvement by showing that $O(n^{1.75})$ steps suffice for an exploration if either the graph in each step has bounded degree, or if the graph in each step has arbitrary degree but the agent can make two moves in each step (or, equivalently, if the agent can make one move per step but the graph changes only in every other step). Surprisingly, the improvement for both cases follows from the same analysis: The key insight is that the better bound on the number of time steps required for an exploration can be proved if the graph in each time step admits a spanning tree of bounded degree. The existence of such a spanning tree is obvious if the graph itself is connected and has bounded degree, and it follows in the model with two moves per step because the square of any connected graph has a spanning tree of bounded degree.

To complement our positive result, we also show that letting the agent make c moves per step, for any constant c, cannot improve the worst-case exploration time for any family of temporal graphs where the underlying graph (the union of the graphs in all steps) has $O(n)$ edges by more than a constant factor. Furthermore, we show that it is *NP*-hard to approximate the exploration time with approximation ratio better than $c \log n$ for some constant c, both for the case of bounded degree and the case of two moves per step.

[1] The conference paper [7] proves a weaker bound of $O(\Delta \log \Delta \cdot \frac{n^2}{\log n})$, but a simple change in one calculation shows that the factor Δ can be avoided.

The remainder of the paper is organized as follows. Section 1.1 discusses related work. Preliminaries are given in Section 2. The main results, showing that exploration in $O(n^{1.75})$ time steps is possible for the case of bounded degree and the case of two moves per step, are presented in Sections 3 and 4, respectively. The result that bounds the improvement obtainable by using c moves per step for sparse temporal graphs and the inapproximability results appear in Sections 5 and 6, respectively. Section 7 concludes the paper.

1.1 Related Work

Brodén et al. [3] studied a temporal analogue of the traveling salesperson problem (TSP) in which the graph is a complete graph in every step and the cost of every edge is either 1 or 2 in each time step, with each edge being allowed to change its cost at most k times over the graph's lifetime. They provided an approximation algorithm with approximation ratio $2 - \frac{2}{3k}$. Michail and Spirakis [13] studied this model as well, showing the general problem to be *APX*-hard and presenting a $(1.7 + \varepsilon)$-approximation algorithm. They also considered the temporal exploration problem and showed that it is *NP*-hard to decide if a temporal graph can be explored if no restrictions are placed on the graph in each step. They therefore suggested making the assumption that the graph is connected in each step, which has turned out to be a very useful model to study. They proved that it is *NP*-hard to approximate the temporal exploration problem under this assumption with ratio $2 - \varepsilon$. Erlebach et al. [6] strengthened this result and proved that approximation with ratio $n^{1-\varepsilon}$ for any $\varepsilon > 0$ is *NP*-hard. Moreover, they constructed a concrete family of temporal graphs for which exploration takes $\Omega(n^2)$ time. They also presented further results for special graph classes, including upper bounds of $O(n^{1.5}k^2 \log n)$ steps for underlying graphs of treewidth k and $O(n \log^3 n)$ steps for the case that the underlying graph is a $2 \times n$ grid. For the case that the underlying graph is a planar graph of maximum degree 4 and the graph in each step is a path, they proved that $\Omega(n \log n)$ steps can be necessary in the worst case. The temporal exploration problem for the special case where the underlying graph is a ring has been studied for the setting of T-interval-connectivity (the intersection of the graphs of any T consecutive time steps is connected) by Ilcinkas and Wade [11]. Decentralized algorithms for the exploration of temporal rings have been studied by Di Luna et al. [5]. Temporal exploration for the case where the underlying graph is a cactus has been studied by Ilcinkas et al. [10].

For surveys of other work on algorithmic aspects and different models of temporal or time-varying graphs we refer to [4, 12]. Examples of recent work include results on the design of temporal networks [1], on temporal (s, t)-separation problems [8, 15], and on temporal vertex cover with sliding time windows [2].

2 Preliminaries

▶ **Definition 2.1** (Temporal Graph). *We represent a temporal graph $\mathcal{G}$ with underlying graph $G = (V, E)$ using an ordered sequence of static graphs: $\mathcal{G} = \langle G_1, G_2, \ldots, G_\tau \rangle$. The subscripts $i \in 1, 2, \ldots, \tau$ indexing the graphs in the sequence are the discrete time steps 1 to τ, where τ is known as the lifetime of G. Each G_i represents the structure of G in time step i. More precisely, $G_i = (V, E_i)$ is a subgraph of G with $V(G_i) = V(G)$ and $E_i \subseteq E$ for all $1 \leq i \leq \tau$.*

▶ **Definition 2.2** (Temporal walk). *A temporal walk W through a temporal graph $\mathcal{G}$ is given as an alternating sequence of vertices and edge-time pairs*

$$W = v_0, (e_0, i_0), v_1, (e_1, i_1), v_2, \ldots, v_{k-1}, (e_{k-1}, i_{k-1}), v_k$$

that starts at vertex v_0 and ends at vertex v_k. Additionally, we require that $i_0 < i_1 < \ldots < i_{k-1}$, so that an agent following W can traverse at most one edge per time step. Each edge-time pair (e_j, i_j) denotes the traversal of edge $e_j = \{v_j, v_{j+1}\}$ at time step i_j. For such a traversal to be possible, e_j must be present in graph G_{i_j}, i.e., $e_j \in E_{i_j}$. We say that the walk W departs at time i_0 (or, at a time $i' \leq i_0$ if we imagine the walk to wait at v_0 from time i' to time i_0), and arrives at time $i_{k-1} + 1$.

We often view a temporal walk W, defined as above, as describing the movement of an agent that is initially located at v_0 and can, in each step, either stay at its current vertex or move to a neighboring vertex in the current graph.

▶ **Lemma 2.3** (Reachability Lemma [6]). *Let $\mathcal{G}$ be a temporal graph with vertex set V, and assume that $\mathcal{G}$ is connected in each step. Then an agent situated at any vertex $u \in V$ at any time $t \leq \tau - n$ can reach any other vertex $v \in V$ in at most $|V| - 1 = n - 1$ steps, i.e., by time step $t + n - 1$.*

▶ **Problem** (Temporal Exploration). *An instance of the* Temporal Exploration *(TEXP) problem is given by a pair $(\mathcal{G}, s)$, where $\mathcal{G} = \langle G_1, G_2, \ldots, G_\tau \rangle$ is an arbitrary temporal graph with lifetime $\tau \geq |V(G)|^2 = n^2$ (in order to ensure that there exist feasible solutions for any instance), and $s \in V(\mathcal{G})$ is a start vertex. The problem then asks for a temporal walk W that departs from vertex s and visits all vertices of $\mathcal{G}$ and minimizes the arrival time at the last unvisited vertex. We make the extra assumption that the graph is connected in each step; without this it could happen that there exists no valid exploration schedule.*

We end the section with an auxiliary lemma used in the next section.

▶ **Lemma 2.4.** *Let $T = (V, E)$ be a tree with maximum degree Δ and $U \subseteq V$. Then $\lfloor |U|/(\Delta + 1) \rfloor$ pairs of vertices in U can be found such that the paths between all pairs are pairwise vertex-disjoint.*

Proof. We root T at an arbitrary vertex and explain how to iteratively select pairs $\{u, v\} \subseteq U$. The procedure will ensure that, whenever we select a pair of vertices in U, there are at most $\Delta - 1$ other vertices in U that can no longer be paired up and have to be removed from U.

Let $u, v \in U$ be two vertices such that their lowest common ancestor $w = \mathrm{lca}(u, v)$ has the largest possible depth. Note that we may have $w = v$ or $w = u$. Since T has maximum degree Δ, w can have at most Δ children. There can be at most $\Delta + 1$ vertices from U in the subtree rooted at w: Each subtree rooted at a child of w contains at most one vertex from U (by our choice of w), and w itself may be in U. Select the pair $\{u, v\}$ and remove u, v and all other vertices that are in the subtree rooted at w from U, a total of at most $\Delta + 1$ vertices. It is easy to see that the paths between the pairs selected by this procedure are vertex-disjoint. ◀

3 Exploration of Degree-Bounded Temporal Graphs

In this section, our goal is to construct a temporal walk using $O(n^{1+\alpha})$ time steps on a temporal n-vertex graph $\mathcal{G}$ for some α with $0 < \alpha < 1$. We assume that the maximum degree of the graph G_i in each step is bounded by a constant $\Delta \in \mathbb{N}$. While constructing the temporal walk, we distinguish between vertices that already belong to our constructed temporal walk, which we call *seen vertices*, and the remaining vertices, called *unseen vertices*. We first show in Section 3.1 that $O(n^{1.8})$ time steps suffice, and then improve the analysis further to get $O(n^{1.75})$ time steps in Section 3.2.

3.1 Phases, Subphases, Labels and Forbidden Sets

Recall that the reachability lemma allows us to easily construct a temporal walk with arrival time $O(n^2)$ by picking an arbitrary order $s = v_1, v_2, \ldots, v_n$ of vertices and searching for a temporal walk from v_1 to v_2, from v_2 to v_3, and so on, each starting when the previous walk arrives and using $O(n)$ time steps. To find a better solution, we have to avoid fixing the order of the vertices without considering the given temporal graph. We divide our construction into so-called *phases*, each consisting of $O(n)$ time steps. Within each phase we construct several *temporal subwalks*, one ending at each unseen vertex. We say that a subwalk is better than another subwalk if it contains more unseen vertices. Among the constructed subwalks, we then choose a best subwalk and use it to extend the temporal walk constructed so far.

By leaving $O(n)$ time steps between two phases, we can easily connect the subwalks of two subsequent phases by the reachability lemma.

Let ℓ be the number of unseen vertices at the beginning of the current phase. The phase is split into *subphases* $1, 2, 3, \ldots$ where the goal of each subphase is to replace a constant fraction of the subwalks by subwalks that have at least one more unseen vertex. For every unseen vertex v we use a *label* $L(v)$ to store the (ordered sequence of the) unseen vertices of the subwalk ending at v. We also keep track of the subwalk that visits all the unseen vertices in $L(v)$, and we use $L(v)$ to refer to that subwalk if no confusion can arise.

We set $L(v) = v$ at the beginning of the phase. Moreover, we maintain the invariant that after the ith subphase, no subwalk has more than $i + 1$ unseen vertices, i.e., $|L(v)| \leq i + 1$ for all unseen vertices v. We cannot guarantee that there is an improvement w.r.t. the total number of unseen vertices of the subwalks in every time step. Instead, at the beginning of each subphase, we place labels on all unseen vertices. In each time step the idea is to propagate labels from a vertex to one of its neighbors and analyze the total improvement in the spread of the labels. Whenever a label reaches another unseen vertex w, a longer subwalk is found that may replace the label of w in the next subphase. A similar approach was used by Erlebach and Spooner [7].

To measure the improvement during the time steps, we additionally define for every vertex v a *home set* $H(v)$ consisting of labels of unseen vertices that can reach v within the time steps of the current subphase. At the beginning of a subphase, we set $H(v) = \{L(v)\}$ for every unseen vertex v, and $H(v) = \emptyset$ for the remaining vertices. For technical reasons, in subphase i, the size of each home set is bounded by $2i + 1$.

During the extension of a subwalk we have to avoid adding the same unseen vertices to a subwalk again. For this purpose, we store for every unseen vertex v a set of vertices whose subwalks cannot be extended by v (because they already contain v). More precisely, we store a *forbidden set* $F(v)$ of every unseen vertex v that consist of all unseen vertices w with $v \in L(w)$. Note that $v \in F(v)$ for every unseen vertex v. We next describe our first operation on the home sets.

Operation 1: If u and v are connected by an edge in the current time step, we are allowed to copy a label L from the home set $H(v)$ into the home set $H(u)$. In addition, we can delete a label L' from $H(u)$, i.e., we can replace a label L' by L in $H(u)$.

As one can easily see by induction, a label of an unseen vertex u can be part of a home set of a vertex v by the start of time step t only if there is a temporal walk from u to v with arrival time t in the current subphase.

In the following, we motivate a second operation. Let v and u be unseen vertices. If $L(u) \in H(w)$ where $w = v$ or w is a neighbor of v, $u \notin F(v)$ and $|L(u)| \geq |L(v)|$ holds, the subwalk $L(u)$ ending at u can be extended by v and this can be a subwalk ending at v that is better than the current subwalk $L(v)$. In this case, we can take the better subwalk for v,

i.e., we define $L_{\text{new}}(v) := L(u) \circ v$. Note that we have that $|L(u)| \leq i$ at the end of subphase $i-1$ by our invariant and hence it follows that $|L_{\text{new}}(v)| \leq i+1$. Thus, the invariant holds also in the next subphase.

For the remaining time steps of the current subphase, it would seem to be useful to start propagating the new label of v starting from v. However, then we would have several labels ending at v and this would increase the total size of the forbidden sets, which we cannot afford. Therefore, we set $L(v) := L_{\text{new}}(v)$ only at the end of the current subphase. For the remainder of the current subphase, we leave all $L(v)$ in the home sets unchanged. In particular, it is still possible that the label $L(v)$ reaches another unseen vertex w – which means that we may also be able to set $L_{\text{new}}(w) := L(v) \circ w$. Altogether, we get the following operation.

Operation 2: If a label $L(u)$ is in $H(x)$ for some vertex x in the closed neighborhood of an unseen vertex v and if $|L(v)| \leq |L(u)|$ as well as $u \notin F(v)$, then we can define $L_{\text{new}}(v) := L(u) \circ v$.

In the following we measure our progress by an increase in the potential function $\phi = |\{u \text{ unseen vertex} \mid L'(u) \neq L(u)\}| + \sum_{v \in V} \sum_{L \in H(v)} |L|$ where $L'(u) := L_{\text{new}}(u)$ if $L_{\text{new}}(u)$ is defined in the current subphase and $L'(u) := L(u)$ otherwise.

▶ **Lemma 3.1.** *Assume that we are in the ith subphase. Given a set of unseen vertices U of size k with $L(u) \in H(u)$, $L'(u) = L(u)$ and $|F(u)| \leq 2i$ for all $u \in U$, the operations from above allow us to modify the labels of the unseen vertices and the home sets of all vertices in such a way that ϕ increases by $\Omega(k/i)$ within one time step.*

Proof. We consider a spanning tree T in the graph of $\mathcal{G}$ in the time step under consideration. Since the graph has maximum degree at most Δ, also T obeys this degree bound.

Next we pair up the unseen vertices in such a way that they are connected by pairwise vertex-disjoint paths. However, we are not allowed to pair up vertices where one is in the forbidden set of the other. Therefore, we compute a set of unseen vertices $U' \subseteq U$ such that we can pair up the vertices in U' without taking the forbidden sets into consideration. To determine U', start with $U' = \emptyset$ and greedily iterate over the vertices $u \in U$. Whenever $F(u) \cap U' = \emptyset$ and $L(u) \cap U' = \emptyset$ holds, add u to U'. The latter condition guarantees $F(v) \cap (U' \cup \{u\}) = \emptyset$ for all v already in U'. Since each vertex u in U' can prevent the insertion of at most $|L(u) \setminus \{u\}| + |F(u) \setminus \{u\}| \overset{\text{invariant}}{\leq} 3i-1$ vertices into U', $|U'| = \Omega(k/i)$. Next, pair up the vertices of U' in such a way that, in T, the paths between every two pairs of vertices are pairwise vertex-disjoint. By Lemma 2.4, we obtain at least $\Omega(k/i)$ pairs since the current graph of $\mathcal{G}$ and thus T has constant degree Δ.

We next show that, for each of the chosen pairs of vertices u and v, ϕ increases by at least one. For this purpose, we focus on the path P between u and v in T, and let w be the neighbor of u in P. W.l.o.g. $|L(u)| \leq |L(v)|$ holds. We consider two cases.

Case 1. The home set of w contains a label $L(x)$ with $x \notin F(u)$ and $|L(x)| \geq |L(u)|$. Then we define $L_{\text{new}}(u) := L(x) \circ u$ using Operation 2 and remove u from U, but leave all occurrences of $L(u)$ in the home sets unchanged. In this way we get a potential increase since u newly satisfies the condition $L'(u) \neq L(u)$.

Case 2. Otherwise, we argue as follows. Note that, if $|H(w)| = 2i+1$ and $|L(x)| \geq |L(u)|$ for all $L(x) \in H(w)$, then we are in Case 1 since $|F(u)| \leq 2i$ for all $u \in U$. Thus, either $|H(w)| < 2i+1$ or $H(w)$ contains a label $L(x)$ that is strictly shorter than $|L(u)|$. Moreover, since $u, v \in U'$, we have $v \notin F(u)$. Since we are not in Case 1, $L(v) \notin H(w)$.

Let f be the function that removes from a set all labels that are shorter than $|L(v)|$. Note that $|f(H(w))| \leq 2i$ because either $|H(w)| \leq 2i$ or $H(w)$ contains a label that is strictly shorter than $|L(u)|$ and thus also strictly shorter than $|L(v)|$.
Now, observe that it is not possible that, for each pair of subsequent vertices x (possibly w) and y (possibly v) on the path from w to v, where x is closer to w than y, $f(H(x)) \supseteq f(H(y))$ holds – this would be a contradiction to $L(v) \notin f(H(w))$ and $L(v) \in f(H(v))$. Therefore, there must be a pair of consecutive vertices x and y on the path, with x closer to w, such that $f(H(x))$ is not a superset of $f(H(y))$. Among all such pairs of vertices, choose the one such that x is closest to w. As we have $|f(H(w))| \leq 2i$, it follows that $|f(H(x))| \leq 2i$. By Operation 1 we copy a label $L \in f(H(y)) \setminus f(H(x))$ from $H(y)$ to $H(x)$. Possibly, we additionally have to remove a shorter label L' from $H(x)$ in order to ensure that $|H(x)|$ remains bounded by $2i + 1$. If $|H(x)| = 2i + 1$, then L' must exist since $H(x)$ has at most $2i$ labels being $\geq |L(v)|$ long. Thus, ϕ increases. In case $x \in U$, we have to be careful that we do not remove $L(x)$ from $H(x)$ in this operation. Therefore, in that case we do not replace $L(x)$ in $H(x)$ by a longer label unless all other labels in $H(x)$ are longer than $L(x)$. If this is the case and we replace $L(x)$ in $H(x)$, then $H(x)$ now contains $2i + 1$ labels longer than $L(x)$ and, since $|F(x)| \leq 2i$, one of these is $L(v')$ for some $v' \notin F(x)$ and we can apply Operation 2 and set $L_{\text{new}}(x) = L(v') \circ x$ and remove x from U. ◀

To guarantee the condition on the size of the forbidden sets in the lemma above, we do not add all unseen vertices u with $L'(u) = L(u)$ into the set U. Instead, we initially define $U = \{u \text{ unseen vertex} \mid L'(u) = L(u) \text{ and } |F(u)| \leq 2i\}$. Since the total label length is at most $i\ell$ by our invariant, there can be at most $\ell/2$ unseen vertices u with $|F(u)| > 2i$. As long as we have $L'(u) \neq L(u)$ for at most $\ell/4$ unseen vertices u in the current subphase, we have $|U| \geq \ell - \ell/2 - \ell/4 = \ell/4$.

Within each time step, the set of vertices u with $L'(u) = L(u)$ shrinks whenever we apply Operation 2. We apply the lemma only while at most $\ell/4$ vertices have been removed from U, so this does not cause a problem.

▶ **Lemma 3.2.** *Given a set of unseen vertices U of size ℓ in the beginning of the ith subphase, $\Theta(ni^3/\ell)$ time steps allow us to increase the length of at least $\ell/4$ subpaths (i.e., labels of unseen vertices) by one.*

Proof. By Lemma 3.1, $\phi \geq \ell/4 + i(2i + 1)n$ after $\Theta(i/4 + i^2(2i + 1)(n/\ell)) = \Theta(ni^3/\ell)$ time steps (or we set L_{new} for at least $\ell/4$ vertices even earlier). This can happen only if $L'(u) \neq L(u)$ for at least $\ell/4$ unseen vertices u since we have in the home sets at most $(2i+1)$ labels of length i for every vertex. ◀

We stop the construction of the temporal walk in phases when fewer than n^α unseen vertices remain for some α with $0 < \alpha < 1$ and finish the temporal walk by adding the remaining n^α unseen vertices in $O(n^{1+\alpha})$ time steps (by visiting them in arbitrary order using the reachability lemma). Therefore, we can assume for the next lemma that the number of unseen vertices is $\ell \geq n^\alpha$.

▶ **Lemma 3.3.** *After $O(n)$ time steps within one phase, a temporal subwalk consisting of $\Theta(\ell^{1/4}) = \Omega(n^{\alpha/4})$ unseen vertices has been found.*

Proof. By Lemma 3.2, we spend $\Theta(ni^3/\ell)$ time steps in subphase i. Since we have $O(n)$ time steps, the number of subphases x within one phase is bounded by the following equation: $\Theta((n/\ell)(1^3+2^3+\ldots+x^3)) \subseteq O(n)$. This means that we can have $x = \Theta(\ell^{1/4})$ subphases. ◀

By choosing $\alpha = 4/5$, we have to run $(n - n^\alpha)/n^{\alpha/4} = \Theta(n^{4/5})$ phases until n^α vertices remain. Thus, this part uses $\Theta(n^{9/5})$ time steps, which is also true for the construction of the rest of the temporal walk.

▶ **Theorem 3.4.** *Let $\mathcal{G}$ be a temporal graph with n vertices that is connected and has constant degree in every time step. Then, $\mathcal{G}$ admits a temporal walk that visits all vertices and arrives at the last unvisited vertex after $O(n^{9/5}) = O(n^{1.8})$ time steps.*

3.2 A Tighter Analysis

For the analysis in the previous section, we have applied Lemma 3.3 with the pessimistic assumption that the number of unseen vertices at the start of each phase is only n^α (giving subwalks visiting $\Theta(n^{\alpha/4}) = \Theta(n^{1/5})$ unvisited vertices). There are clearly more unseen vertices in the earlier phases (e.g., $n-1$ unseen vertices at the start of the first phase, thus admitting a subwalk visiting $\Theta(n^{1/4})$ vertices by Lemma 3.3). To get a tighter bound, we analyze the required number of phases more carefully. Let γ, $0 < \gamma < 1$, be the constant hidden in the bound $\Theta(\ell^{1/4})$ from Lemma 3.3, i.e., each phase visits at least $\gamma \cdot \ell^{1/4}$ unseen vertices if there are ℓ unseen vertices at the start of the phase. An upper bound $f(t)$ on the number of unseen vertices after t phases is now given by the following equation system:

$$\begin{aligned} f(0) &= n \\ f(t) &= f(t-1) - \gamma \cdot (f(t-1))^{1/4}, \text{ for } t \geq 1 \end{aligned} \tag{1}$$

The following claim establishes a closed formula for an upper bound on $f(t)$.

▷ Claim 3.5. $f(t) \leq (n^{3/4} - \frac{3}{4}\gamma t)^{4/3}$ for all $t \geq 0$.

Proof. We prove the claim by induction. For the base of the induction, note that for $t = 0$ we have $f(0) = n = (n^{3/4} - \frac{3}{4}\gamma \cdot 0)^{4/3}$. For the induction step, assume that the claim holds for $t-1$. We want to show that it also holds for t. By the induction hypothesis and using the fact that the function $g(x) = x - \gamma x^{1/4}$ is monotone increasing for $x \geq 1$ (even for $x \geq (\gamma/4)^{4/5}$) we get:

$$\begin{aligned} f(t) &= f(t-1) - \gamma \cdot (f(t-1))^{1/4} \\ &\leq (n^{3/4} - \tfrac{3}{4}\gamma(t-1))^{4/3} - \gamma \cdot \sqrt[4]{(n^{3/4} - \tfrac{3}{4}\gamma(t-1))^{4/3}} \\ &= (n^{3/4} - \tfrac{3}{4}\gamma t + \tfrac{3}{4}\gamma)^{4/3} - \gamma \cdot (n^{3/4} - \tfrac{3}{4}\gamma t + \tfrac{3}{4}\gamma)^{1/3} \end{aligned}$$

We need to show that the right-hand side is bounded by $(n^{3/4} - \frac{3}{4}\gamma t)^{4/3}$. Setting $y = n^{3/4} - \frac{3}{4}\gamma t$, this is equivalent to:

$$\begin{aligned} & (y + \tfrac{3}{4}\gamma)^{4/3} - \gamma \cdot (y + \tfrac{3}{4}\gamma)^{1/3} \leq y^{4/3} \\ \Leftrightarrow \quad & (y + \tfrac{3}{4}\gamma)^{4/3} - y^{4/3} \leq \gamma \cdot (y + \tfrac{3}{4}\gamma)^{1/3} \end{aligned}$$

With $h(x) = x^{4/3}$, the latter inequality is equivalent to

$$h(y + \tfrac{3}{4}\gamma) - h(y) \leq \tfrac{3}{4}\gamma \cdot h'(y + \tfrac{3}{4}\gamma),$$

which holds because the function $h(x)$ is convex. Hence, the inductive step is complete. ◁

Equation (1) is valid as long as the number of unseen vertices is sufficiently large. The value of t for which $f(t)$ becomes smaller than $n^{3/4}$ is clearly smaller than $\frac{4}{3\gamma}n^{3/4}$. Hence, $O(n^{3/4})$ phases of length $O(n)$, a total of $O(n^{1.75})$ time steps, suffice to reduce the number of unseen vertices to a value below $n^{3/4}$, and the remaining unseen vertices can be visited in $O(n^{1.75})$ steps via the reachability lemma.

▶ **Theorem 3.6.** *Let $\mathcal{G}$ be a temporal graph with n vertices and constant degree that is connected in every time step. Then, $\mathcal{G}$ has a temporal walk that visits all vertices and uses $O(n^{7/4}) = O(n^{1.75})$ time steps.*

4 Two Moves per Time Step in a Graph of Unbounded Degree

Erlebach, Hoffmann and Kammer [6] showed that there are temporal graphs of unbounded degree where a temporal walk visiting all vertices needs $\Theta(n^2)$ time steps. We show in this section that we can break this lower bound not only if the graph in each step has bounded degree, but also if we allow up to *two moves* per time step, i.e., in each time step we are allowed to move from a vertex v to a neighbor w of v and then to a neighbor of w.

We handle the two moves in a temporal graph $\mathcal{G} = \langle G_1, G_2, \ldots, G_\tau \rangle$ for some $\tau \in \mathbb{N}$ by adding additional edges between each pair of vertices of distance 2 in each graph G_i $(i = 1, \ldots, \tau)$ to obtain a graph G_i^2, the so-called *square graph* of G_i. Afterwards, we can simply use the "one-move-per-time-step approach" in $\mathcal{G}^2 = \langle G_1^2, G_2^2, \ldots, G_\tau^2 \rangle$.

▶ **Lemma 4.1.** *The square of a connected graph has a spanning tree with maximum degree 3.*

Proof. Let G be a connected graph and G^2 the square of G. Compute a spanning tree T in G. This spanning tree T is also a spanning tree in G^2. Now color all edges of T black. Note that, because the children of every vertex of T are directly connected in G^2, we are allowed to connect them in T and still have a subgraph of G^2.

We next reduce the degree of each vertex in T. First, delete all black edges from each vertex v to all its children except one. Starting from that child, we then connect the children of v by a red path, see also Fig. 1. After applying these changes to each vertex v we can observe the following: Each vertex v (except the root) was either the first child in T or not. In the former case, v has a red edge and the remaining black edges can be to a parent and to at most one child. In the second case, v has at most two red edges to siblings and possible one black edge to a child. Hence, in both cases v has maximum degree 3. ◀

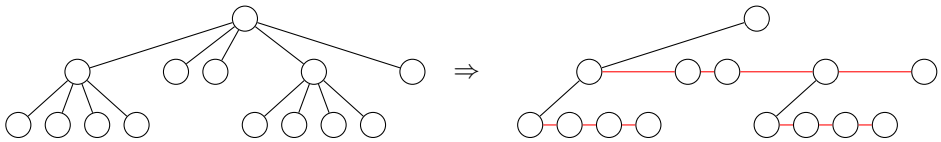


Figure 1 Left: A spanning tree T of a graph G. Right: The transformation of T into a spanning tree in G^2 of maximum degree 3 as shown in Lemma 4.1.

By using the lemma above for the construction of a spanning tree in the proof of Lemma 3.1, we get the following theorem.

▶ **Theorem 4.2.** *Let $\mathcal{G}$ be a temporal graph with n vertices and unbounded degree that is connected in every time step. Then $\mathcal{G}$ has a temporal walk that visits all vertices and uses $O(n^{7/4}) = O(n^{1.75})$ time steps if we allow at least two moves per time step.*

5 Bounded Benefit of c Moves per Step for Sparse Graphs

In the previous section we showed that allowing more than one move per time step enables us to break the existing lower bound of $\Omega(n^2)$ steps on general temporal graphs [6, Theorem 3.5]. We next show that there are certain temporal graph classes where c moves per time step, for an arbitrary integer $c = O(1)$, cannot decrease the worst-case bound on the exploration time by more than a constant factor.

Let $\mathcal{G}$ be a temporal graph with underlying graph $G = (V, E)$. The vertices of G are called *original vertices*. Denote the graph in time step i by G_i. For odd ℓ, we define an operation called *edge-path transformation of length* ℓ that produces a temporal graph $\mathcal{G}'$ with underlying graph G' and graph G'_i in time step i as follows: To construct G' from the underlying graph $G = (V, E)$ of $\mathcal{G}$, the transformation replaces each edge $\{u, v\} \in E$ by a path $\pi_{u,v} = u, a_1, a_2, \ldots, a_{\ell-1}, v$ of ℓ edges, where each a_i is a new *artificial vertex*. For an artificial vertex a_i, we define $o(a_i)$ to be the nearest original vertex, i.e., $o(a_i) = u$ if $i < \ell/2$ and $o(a_i) = v$ if $i > \ell/2$. For an original vertex w, we set $o(w) = w$. If $\{u, v\}$ is present in G_i, the whole path $\pi_{u,v}$ is present in G'_i. If $\{u, v\}$ is not present in G_i, the path $\pi_{u,v}$ without its *middle edge* $\{a_{\lfloor \ell/2 \rfloor}, a_{\lceil \ell/2 \rceil}\}$ is present in G'_i.

A temporal graph with $n \in \mathbb{N}$ vertices is called *sparse* if its underlying graph has $O(n)$ edges. We say that a function $f : \mathbb{N} \to \mathbb{N}$ is *nice* if $f(n/b) = \Omega(f(n))$ holds for any constant $b \geq 1$. For example, it is easy to see that all functions of the form $f(n) = n^g (\log n)^h$ for constant g, h are nice.

▶ **Theorem 5.1.** *Let $c = O(1)$ and let $\mathcal{G}^*$ be a class of sparse temporal graphs such that $\mathcal{G}^*$ is closed under the edge-path transformation of the odd length $\ell \in \{c, c+1\}$. If there is a lower bound on the number of steps for TEXP on $\mathcal{G}^*$ for one move per time step, given as a nice function of the number of vertices, then the same lower bound, up to a constant factor, also applies to TEXP on $\mathcal{G}^*$ if c moves per time step are allowed.*

Proof. Let $\mathcal{G}'$ be the temporal graph obtained from a temporal graph $\mathcal{G} \in \mathcal{G}^*$ by the edge-path transformation of length ℓ. We claim the following: If a temporal walk W' visiting all vertices in $\mathcal{G}'$ uses k time steps with $\leq \ell$ moves per time step, then there is a temporal walk W visiting all vertices in $\mathcal{G}$ consisting of k time steps with one move per time step.

To construct W, we iterate over the time steps and analyze the moves in W' within the current time step. Before and after each time step, we ensure the following invariants. (1) If W' is at a vertex a, then W is at $o(a)$. (2) All original vertices already visited by W' have also been visited by W.

Let a and a' be the (possibly artificial) vertices at which W' is located just before and just after the current time step, respectively. Let $u = o(a)$. Note that W is located at u just before the current step by (1). We let W move from u to $o(a')$ in the current step. Note that, if W' visits during the current step an original vertex that has not yet been visited by W, then that vertex must be $o(a')$. This is because W' starts on the far side of the middle edge on the path to that original vertex, and thus must end on the near side of the middle edge on the same or another path incident with that original vertex. It is easy to see that both invariants hold again after the time step.

Assume there is a lower bound on the worst-case exploration time for $\mathcal{G}^*$ given by a nice function f of the number of vertices. Let $\mathcal{G} \in \mathcal{G}^*$ be a worst case instance where TEXP requires at least $k = f(n)$ steps. Let $\mathcal{G}'$ be obtained from $\mathcal{G}$ by the edge-path-transformation of length ℓ. Using the claim above, we can conclude that TEXP requires at least k steps in $\mathcal{G}'$ even if $\ell \geq c$ moves per time step are allowed. Since $\mathcal{G}' \in \mathcal{G}^*$ and since $\mathcal{G}$ and $\mathcal{G}'$ have the same number of vertices and edges up to constant factors, we have that $k = f(n'/b)$ for some constant b, where n' is the number of vertices of $\mathcal{G}'$. As f is a nice function, we get a lower bound of $\Omega(f(n'))$ for the case where c moves per time step are allowed. ◀

As an application of the theorem above, we can conclude the following from the known lower bound of $\Omega(n \log n)$ steps for sparse temporal graphs in the one-move-per-step model (Theorem 4.1 in [6]).

▶ **Corollary 5.2.** *For a temporal n-vertex graph $\mathcal{G}$ whose underlying graph is planar with maximum degree 4, an optimal exploration can take $\Omega(n \log n)$ steps even if $c = O(1)$ moves per time step are allowed.*

6 Inapproximability Result

▶ **Theorem 6.1.** *It is NP-hard to approximate TEXP with two moves per step on an n'-vertex temporal graph with ratio smaller than $b \log n'$ for some constant $b > 0$.*

Proof. We give a reduction from the Hamiltonian s-t-path problem, which is *NP*-complete even if the input graph is planar and has maximum degree 3 as shown by Garey, Johnson, and Tarjan [9]. Let an instance of the Hamiltonian s-t-path problem be given by a graph G with n vertices and $s, t \in V(G)$. Assume without loss of generality that $n = 2k$ for some $k \in \mathbb{N}$ with $k \geq 2$ (otherwise, simply add a new leaf t' adjacent to t and consider the Hamiltonian s-t'-path problem in the new graph).

We now construct an n'-vertex temporal graph $\mathcal{G}'$ in two *phases*, for some n' specified later. First, we specify the vertex set and all edges that will be present during at least one step of the first phase.

Underlying graph during the first phase. The construction is illustrated in Fig. 2. Create $2n^c$ copies of G (for an arbitrary constant $c \geq 2$). Form two groups of these copies, each of size n^c, calling the first group T and the second group B. Let $T(i)$ be the *i-th T-copy* of G and $B(j)$ the *j-th B-copy* of G. For all $i \in \{1, \ldots, n^c - 1\}$, connect vertex $t \in V(T(i))$ and $s \in V(T(i+1))$ by a *quick link* $e_t(i, i+1)$. Create similar quick links $e_b(i, i+1)$ between the B-copies. Between $t \in V(T(n^c))$ and $s \in V(B(1))$, let there be a *super quick link*.

Moreover, build a path P consisting of further vertices $v_t(1), \ldots, v_t(12n^{c+1})$. Use the first (last) quarter of the path to connect each vertex s in the T-copies (B-copies) of G to the path such that the minimal distance of two such vertices s on the path is at least $3n$. More precisely, for $k \in \{0, \ldots, n^c - 1\}$, connect vertex $v_t(3kn+1)$ with $s \in V(T(k+1))$, and connect $v_t(9n^{c+1} + 3kn + 1)$ with $s \in V(B(n^c - k))$. In total, $\mathcal{G}'$ has $n' = O(n^{c+1})$ vertices.

Temporal realization of the first phase. Let the start vertex be $s \in V(T(1))$. Let the steps $t \in \{1, 2, \ldots, n^{c+1}\}$ constitute the first phase of $\mathcal{G}'$'s lifetime. During this phase, the edges of all $2n^c$ copies of G, the path of length $12n^{c+1}$ and the connections between the T-copies or B-copies and the path P described in the previous paragraph always exist. Additionally, let the edge $e_t(i, i+1)$ be present only in step $in/2$, for all $i \in \{1, \ldots, n^c - 1\}$. Let the super quick link be present only in step $n^{c+1}/2$. Let the edge $e_b(i, i+1)$ be present only in step $(n^{c+1} + in)/2$, again for all $i \in \{1, \ldots, n^c - 1\}$.

It is not hard to see that, if G has a Hamiltonian s-t-path, then a temporal walk with two moves per step can visit all vertices in all copies of G within the first phase: In each copy of G, it uses the $n/2$ time steps to follow the $n - 1$ edges of the Hamiltonian s-t-path and then the quick link (or super quick link) to move to the vertex s of the next copy.

Assume now that G does not have a Hamiltonian s-t-path. If a temporal walk W with two moves per step does not use the super quick link, then none of the vertices in the B-copies have been visited in the first phase.

Otherwise, W must have used all quick links connecting the T-copies. We claim that in any two consecutive T-copies, W has missed at least one vertex. To see this, note that an s-t walk in G consists of at least n edges if there is no Hamiltonian s-t-path. Thus, for the walk to arrive at the ith T-copy via a quick link in step $(i-1)n/2$, visit all vertices of the copy, and leave via a quick link in step $in/2$, it must have used the quick links as the first

move in step $(i-1)n/2$ and as the second move in step $in/2$, respectively. But then it is not possible for the walk to visit all vertices of the $(i+1)$th copy and leave it via a quick link.

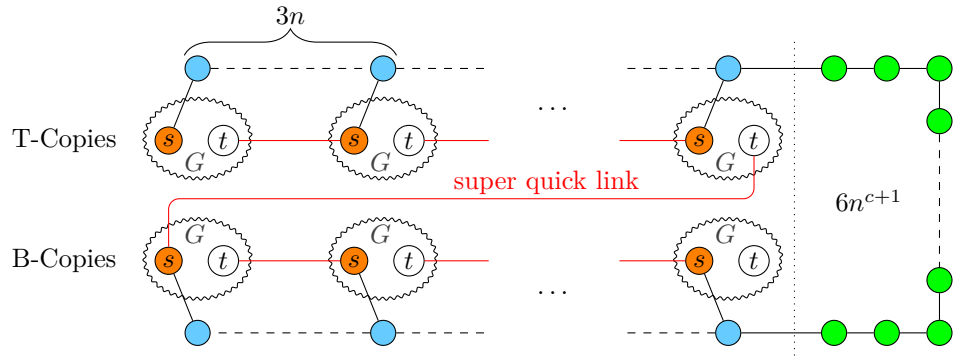


Figure 2 The underlying graph during the first phase. A red line is a quick link.

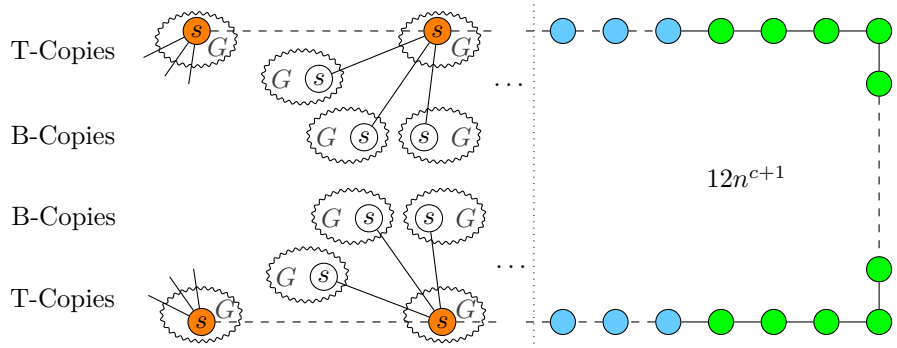


Figure 3 Construction of the temporal graph in the second phase. The connections between the orange vertices differ in the time steps and result from the connections between the labeled vertices in the temporal graph in Fig. 3 in [6].

The second phase. For the whole second phase, the path P is present. Thus, all vertices of P can be visited in $O(n')$ time steps. Therefore, if G has a Hamiltonian s-t-path, $\mathcal{G}'$ can be explored in $O(n')$ time steps.

Erlebach et al. [6, Theorem 4.1] show that TEXP takes $\Omega(N \log N)$ steps on a planar N-vertex temporal graph. More precisely, the proof of their theorem shows that, for arbitrary parameters $r, m \in \mathbb{N}$, there is a planar temporal graph $\mathcal{H}$ with $r+m$ vertices, consisting of r vertices that we call *path vertices* and m vertices that we call *target vertices*, in which $\Omega(r \log m)$ time steps are necessary for visiting all m target vertices. In the temporal graph $\mathcal{H}$, the r path vertices are connected in the form of a path Q in all time steps. The lower bound of $\Omega(r \log m)$ steps for visiting the m target vertices also holds for the model with two moves per step if the length of the rounds in the construction of $\mathcal{H}$ is divided by two. Let $\mathcal{H}'$ be $\mathcal{H}$ with this modification of the length of the rounds.

The idea is now to take in the second phase of our construction the temporal graph $\mathcal{H}'$ for parameters $r = 12n^{c+1}$ and $m = n^c/2$. However, we must construct the temporal graph with the vertices used in Phase 1. Therefore, the second phase of our temporal graph $\mathcal{G}'$ is constructed as follows – see also Fig. 3: Each of the m target vertices v in $\mathcal{H}'$ is identified with a vertex s of a T-copy $T(2i)$ such that no two such vertices s are identified with the same vertex v. Furthermore, the vertex s of each T-copy $T(2i-1)$ is made adjacent to the

vertex s of $T(2i)$. In addition, connect exactly the vertices s of two B-copies to each such vertex v. Our path P becomes the path Q. As in Phase 1, the edges in the copies of G are present in all time steps. Note that the only edges that change during the time steps of the second phase are the edges between the m target vertices of $\mathcal{H}'$. Such an edge exists in the ith time step within Phase 2 exactly if it exists in $\mathcal{H}'$ in the ith time step.

Analysis. Assume that G does not have a Hamiltonian s-t-path. Since we have an unvisited vertex in each pair of consecutive T-copies or in each B-copy after Phase 1, we must visit all m target vertices of $\mathcal{H}'$ to reach the unvisited vertices. Thus, we have to spend $\Omega(r \log m) = \Omega(n^{c+1} \log n^c) = \Omega(n' \log n')$ time steps to explore all vertices of $\mathcal{G}'$, where the last equality follows from $r = \Theta(n')$ and $m = n^c/2 = \Theta((n')^{c/c+1})$.

Since establishing whether $\mathcal{G}'$ can be explored in its entirety in just $O(n')$ steps or requires $\Omega(n' \log n')$ steps also decides whether there exists a Hamiltonian s-t-path in G, we get that it is NP-hard to approximate TEXP with two moves per step with approximation ratio smaller than $b \log n'$ for some constant $b > 0$. ◀

Using a simple variation of the proof of Theorem 6.1, we also get:

▶ **Corollary 6.2.** *It is NP-hard to approximate TEXP on n-vertex temporal graphs whose underlying graph has bounded degree with ratio smaller than $c \log n$ for some constant $c > 0$.*

7 Conclusion

In this paper we have shown that temporal graphs can be explored in $O(n^{1.75})$ time steps if the graph in each step has bounded degree or if two moves per step are allowed. We remark that our proofs are constructive and also yield polynomial-time algorithms to compute such exploration schedules. Moreover, we have shown that TEXP is NP-hard to approximate with ratio better than $b \log n$ for some constant b for the considered cases (bounded degree or two moves per step). The main open problem for these cases is to further reduce the gap between the lower bound of $\Omega(n \log n)$ steps (proved in [6] for bounded degree and in Corollary 5.2 for two moves per step) and the upper bound proved in this paper.

References

1. Eleni C. Akrida, Leszek Gasieniec, George B. Mertzios, and Paul G. Spirakis. The Complexity of Optimal Design of Temporally Connected Graphs. *Theory of Computing Systems*, 61(3):907–944, 2017. doi:10.1007/s00224-017-9757-x.
2. Eleni C. Akrida, George B. Mertzios, Paul G. Spirakis, and Viktor Zamaraev. Temporal Vertex Cover with a Sliding Time Window. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *LIPIcs*, pages 148:1–148:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.148.
3. Björn Brodén, Mikael Hammar, and Bengt J. Nilsson. Online and Offline Algorithms for the Time-Dependent TSP with Time Zones. *Algorithmica*, 39(4):299–319, 2004. doi:10.1007/s00453-004-1088-z.
4. Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012. doi:10.1080/17445760.2012.668546.
5. Giuseppe Antonio Di Luna, Stefan Dobrev, Paola Flocchini, and Nicola Santoro. Live Exploration of Dynamic Rings. In *Proceedings of the 36th IEEE International Conference on Distributed Computing Systems (ICDCS 2016)*, pages 570–579. IEEE Computer Society, 2016. doi:10.1109/ICDCS.2016.59.

6 Thomas Erlebach, Michael Hoffmann, and Frank Kammer. On Temporal Graph Exploration. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP 2015), Part I*, volume 9134 of *LNCS*, pages 444–455. Springer, 2015. doi:10.1007/978-3-662-47672-7_36.

7 Thomas Erlebach and Jakob T. Spooner. Faster Exploration of Degree-Bounded Temporal Graphs. In *Proceedings of the 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*, volume 117 of *LIPIcs*, pages 36:1–36:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.MFCS.2018.36.

8 Till Fluschnik, Hendrik Molter, Rolf Niedermeier, and Philipp Zschoche. Temporal Graph Classes: A View Through Temporal Separators. In *Proceedings of the 44th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2018)*, volume 11159 of *LNCS*, pages 216–227. Springer, 2018. doi:10.1007/978-3-030-00256-5_18.

9 M. R. Garey, David S. Johnson, and Robert Endre Tarjan. The Planar Hamiltonian Circuit Problem is NP-Complete. *SIAM Journal on Computing*, 5(4):704–714, 1976. doi:10.1137/0205049.

10 David Ilcinkas, Ralf Klasing, and Ahmed Mouhamadou Wade. Exploration of Constantly Connected Dynamic Graphs Based on Cactuses. In *Proceedings of the 21st International Colloquium on Structural Information and Communication Complexity (SIROCCO 2014)*, volume 8576 of *LNCS*, pages 250–262. Springer, 2014. doi:10.1007/978-3-319-09620-9_20.

11 David Ilcinkas and Ahmed Mouhamadou Wade. Exploration of the T-Interval-Connected Dynamic Graphs: the Case of the Ring. *Theory of Computing Systems*, 62(5):1144–1160, 2018. doi:10.1007/s00224-017-9796-3.

12 Othon Michail. An Introduction to Temporal Graphs: An Algorithmic Perspective. *Internet Mathematics*, 12(4):239–280, 2016. doi:10.1080/15427951.2016.1177801.

13 Othon Michail and Paul G. Spirakis. Traveling salesman problems in temporal graphs. *Theoretical Computer Science*, 634:1–23, 2016. doi:10.1016/j.tcs.2016.04.006.

14 Othon Michail and Paul G. Spirakis. Elements of the theory of dynamic networks. *Communications of the ACM*, 61(2):72, 2018. doi:10.1145/3156693.

15 Philipp Zschoche, Till Fluschnik, Hendrik Molter, and Rolf Niedermeier. The Complexity of Finding Small Separators in Temporal Graphs. In *Proceedings of the 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*, volume 117 of *LIPIcs*, pages 45:1–45:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.MFCS.2018.45.

Distributed Arboricity-Dependent Graph Coloring via All-to-All Communication

Mohsen Ghaffari
ETH Zurich, Switzerland
ghaffari@inf.ethz.ch

Ali Sayyadi
Sharif University of Technology, Iran
ali.sayyadi.98@gmail.com

Abstract

We present a constant-time randomized distributed algorithms in the congested clique model that computes an $O(\alpha)$-vertex-coloring, with high probability. Here, α denotes the arboricity of the graph, which is, roughly speaking, the edge-density of the densest subgraph. Congested clique is a well-studied model of synchronous message passing for distributed computing with all-to-all communication: per round each node can send one $O(\log n)$-bit message algorithm to each other node. Our $O(1)$-round algorithm settles the randomized round complexity of the $O(\alpha)$-coloring problem. We also explain that a similar method can provide a constant-time randomized algorithm for decomposing the graph into $O(\alpha)$ edge-disjoint forests, so long as $\alpha \leq n^{1-o(1)}$.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Mathematics of computing → Graph algorithms

Keywords and phrases Distributed Computing, Message Passing Algorithms, Graph Coloring, Arboricity, Congested Clique Model, Randomized Algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.142

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Funding The first author acknowledges the support of the Swiss National Foundation, under project number 200021_184735. The second author is grateful for the support of the Student Summer Research Fellowship (SSRF) program at the Computer Science department of ETH Zurich; this project was initiated during an undergraduate internship in the summer of 2018.

1 Introduction and Related Work

This paper settles the randomized complexity of the graph coloring problem with a number of colors linear in its arboricity, in the congested clique model of distributed computing: we show that constant time suffices. Before formally stating our result, let us start by reviewing the model and the problem statement, as well as state of the art.

1.1 Background: Model, Problem Statement, and State of the Art

The Congested Clique Model. We work with the congested clique model of distributed computing, which was introduced by Lotker et al. [18] and has received extensive attention over the past seven years[1]. There are n processors in the system. We are also given an n-node graph $G = (V, E)$, which has one node corresponding to each processor. This graph

[1] There are many paper, and it is well-beyond the scope of this paper to survey them all. As a prominent example, see the beautiful line of developments for the minimum spanning tree problem [18, 12, 11, 14], which also settled its complexity to be $O(1)$ rounds [14].

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 142; pp. 142:1–142:14


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

can abstract various things. For instance, it might be abstracting the dependencies in a *contention resolution* setting, as follows: processors that are adjacent in the graph are dependent and cannot perform their tasks simultaneously, e.g., they should not transmit their wireless messages simultaneously or they should not read/write to the same memory location simultaneously, or many other similar settings of accessing shared resources. We assume that the graph G is known to the processors in a distributed fashion: each processor/node knows its own neighboring processors/nodes in G. In the congested clique model, we assume that processors can communicate in synchronous rounds, in an all-to-all fashion. Per round, each processor can send one $O(\log n)$-bit message to each other processor. At the end, each processor should know its own part of the output, e.g., the color of its own node.

A discussion about the model, in contrast with other distributed models. Two other well-studied, and in fact more classic, models of distributed computing are CONGEST and LOCAL [23]. In these models, the communication network among the processors is the same as the graph G for which they want to solve the graph problem. In the CONGEST model, per round, each node can send one $O(\log n)$ bit message to each of its neighbors (in the graph, which abstract both the problem and the communication network). The LOCAL model is the variant where there is no upper bound on the message sizes.

These models CONGEST and LOCAL are perfectly suitable for settings such as computer networks, where we want to solve a graph problem about the communication network. This was in fact the original motive of these models. Moreover, there has been a host of beautiful and powerful algorithmic and impossibility results developed in the context of these two models. For some other applications, these models may be less suitable. In particular, when the graph problem is a logical one – as in the abstraction used in the above above example to capture contentions among processor – such a coincidence of communication graph and problem graph may look strange. This is perhaps a partial explanation for the surge of recent interest in distributed algorithms in the congested clique model. Some other reasons are more theoretical and have to do with setting the limitation of locality aside, see the introduction of [11] for a more extensive discussion of this aspect.

The congested clique model is more permissive than CONGEST and we can easily simulate CONGEST-model algorithms in congested clique. But that would not be enough for us, as there is no constant-algorithm for our problem in CONGEST. In fact, in the CONGEST and LOCAL models, the problem has an $\Omega(\log n)$ lower bound [17]. We build on the techniques developed in the CONGEST and LOCAL model, but we also present and use several other algorithmic ideas which allow us to go much further and solve the problem in constant time.

Problem Statement – Arboricity-Dependent Graph Coloring. The graph coloring problem is a well-studied problem with a wide range of applications. It asks for coloring the vertices with few colors in a way that no two adjacent nodes have the same color. For instance, in the context of the contention resolution example mentioned above, this would allow the processors to perform their tasks in a few time slots, without any two dependent processors working concurrently.

Our objective is to compute a coloring whose number of colors is $O(\alpha)$, where α denotes the *arboricity* of the graph. We recall that arboricity is a measure of sparsity of the graph. For a graph $G = (V, E)$, its arboricity is defined as the maximum edge-density $\lceil \frac{E(V')}{|V'|-1} \rceil$ among all induced subgraphs on vertices $V' \subseteq V$ with $|V'| > 1$. Alternatively, by a beautiful result of Nash-Williams[19], an equivalent formulation can be given as the minimum number of edge-disjoint forests into which we can decompose the edges of G. We assume that α is given as an input parameter and the input graph has arboricity at most α.

Any graph with arboricity α admits a coloring with 2α colors and this bound is sharp. For the former, note that we can arrange vertices as a_1 to a_n such that each v_i has at most $2\alpha - 1$ neighbors v_j with a higher index $j > i$. For the latter, a graph consisting of disjoint cliques, each having 2α vertices, gives the sharpness example.

Following [3, 5], we set out target to be obtaining a coloring close to this existentially optimal bound; ideally we would want a coloring with $(2+\epsilon)\alpha$ colors for a small constant $\epsilon > 0$, but we relax it further to $O(\alpha)$ colors, for this paper. Note that $\alpha \leq \Delta$, where Δ denotes the maximum degree of the graph. Hence, coloring with $O(\alpha)$ colors also directly gives a coloring with $O(\Delta)$ colors. However, α can be much smaller than Δ; take for instance a star graph where $\Delta = n - 1$ and $\alpha = 1$. For many graph problems in practice, α is indeed much smaller than Δ: even though they might have a few nodes of high degree, typically they do not contain very dense subgraph, at least not as dense as average degree of Δ.

State-of-the-Art – LOCAL and CONGEST. As it will become clear soon, there is a large body of work on distributed algorithms for graph coloring, and other related problems, and it is well beyond the scope of this paper to review them all. For a survey of the results prior to 2013 on algorithms in the LOCAL and CONGEST models, we refer the reader to the *Distributed Graph Coloring* book by Barenboim and Elkin [3]. We will just mention the best known result in randomized and deterministic settings:

In the LOCAL model, the best known randomized upper bound for $(\Delta+1)$-coloring problem is $2^{O(\sqrt{\log\log n})}$ [7] and the best known deterministic upper bound for $(\Delta+1)$-coloring is $2^{O(\sqrt{\log n})}$[20]. Only if we relax the number of colors to $\Delta^{1+\epsilon}$ for some constant $\epsilon > 0$, a polylogarithmic-time – and particularly $O(\log\Delta\log n)$-round – algorithm is known, thanks a breakthrough of Barenboim and Elkin [2]. For the CONGEST model, the best known randomized algorithm for $(\Delta+1)$-coloring runs in $O(\log\Delta) + 2^{O(\sqrt{\log\log n})}$ rounds [9]. If we parameterize the algorithm's complexity by Δ, the best known algorithm runs in $\tilde{O}(\sqrt{\Delta}) + O(\log^* n)$ rounds [4].

Regarding coloring dependent on α, Linial [17] showed that any coloring with $O(\alpha)$ colors, or even $poly(\alpha)$ colors, needs at least $\Omega(\log n)$ rounds, even in the LOCAL model. Barenboim and Elkin[1] gave a deterministic algorithm that gives a $\Theta(\alpha^2)$ coloring in $O(\log n)$ rounds, and an $O(\alpha^{1+\epsilon})$ coloring in $O(\log\alpha\log n)$ rounds. Ghaffari and Lymouri [10] gave a randomized algorithm that computes a $(2+\epsilon)\alpha$ colors, for constant $\epsilon > 0$, if $\alpha = \tilde{\Omega}(\log n)$, an $O(\alpha\log\alpha)$-coloring in $O(\log n)$ rounds, and an $O(\alpha)$-coloring running in $\tilde{O}(\log n)$ rounds.

State-of-the-Art – CONGESTED-CLIQUE. There has been a sequence of improvements, for coloring with a number of colors dependent on the maximum degree Δ, as we overview next: Hegeman and Pemmaraju [12] gave algorithms for $O(\Delta)$-coloring in the CONGESTED-CLIQUE model, which run in $O(1)$ rounds if $\Delta \geq \Theta(\log^4 n)$ and in $O(\log\log n)$ rounds otherwise. For $(\Delta+1)$ coloring problem, recently Parter [21] designed the first sublogarithmic-time $(\Delta+1)$ coloring algorithm for CONGESTED-CLIQUE, which runs in $O(\log\log\Delta\log^*\Delta)$ rounds. Parter and Su [22] improved the bound to $O(\log^*\Delta)$ round. Finally, the round complexity of $(\Delta+1)$-coloring was settled very recently $O(1)$, by a recent work of Chang, Fischer, Ghaffari, Uitto, and Zheng [6].

For coloring with the number of colors dependent on the arboricity α, we are aware only one prior work in the CONGESTED-CLIQUE: Barenboim and Khazanov[5] gave a deterministic $O(\alpha)$-coloring which runs in α^ϵ rounds, for a constant $\epsilon > 0$, and an $O(\alpha^{2+\epsilon})$-coloring, which runs in $O(\log^* n)$ rounds. For randomized algorithms, we are not aware of any faster algorithm than these deterministic ones, or the randomized algorithms of the CONGEST model (which need at least $\Omega(\log n)$ rounds).

1.2 Our Results

Coloring. We settle the randomized complexity of $O(\alpha)$-coloring in the congested clique model, by showing that constant rounds suffice.

▶ **Theorem 1.** *There is a randomized distributed algorithm that computes an $O(\alpha)$-coloring in constant rounds of the congested clique model, with high probability.*

Forest Decomposition. Our techniques also allow us to compute a decomposition of the graph into $O(\alpha)$ edge-disjoint forests in constant time, so long as $\alpha \leq n^{1-o(1)}$. The best known previous algorithm for this problem was by Barenboim and Khazanov [5] and it uses $O(\log \alpha)$ rounds. The statement of our result appears in the following theorem.

▶ **Theorem 2.** *There is a randomized distributed algorithm that computes a decomposition of the graph into $O(\alpha)$ edge-disjoint forests in constant rounds of the congested clique model, whenever $\alpha \leq n^{1-O(\frac{1}{\sqrt{\log n}})}$, with high probability.*

1.3 Technical Overview

On a high-level, our result makes use of a number of technical ingredients, and puts them together – along with several smaller ideas – in a way that results in the claimed constant-round algorithm for $O(\alpha)$ coloring. We start with a brief list of these technical ingredients:

1. The iterative peeling algorithm of Barenboim and Elkin [1], which partitions vertices into successive layers, along with an orientation from lower layers to higher layers, such that each node has out-degree at most $(2+\epsilon)\alpha$.
2. A topology-gathering procedure which allows us to exponentially speed up some LOCAL model algorithms, on sparse graphs and up to a small locality bound [16, 8].
3. An opportunistic topology-gathering idea which allows us to speed up some LOCAL model algorithms to just $O(1)$ rounds of the CONGESTED-CLIQUE model. Variants of this idea have been used by [14, 6].
4. Some ad-hoc but simple random sampling ideas which allow us to use a much sparser subgraph to perform an approximate peeling about the original graph.
5. A simple random partition idea which allows us to break the case of graphs with large arboricity to several graphs with smaller arboricity, or sometimes graphs that are almost small aboricity, in the sense that they have some number of extra edges.
6. A simple randomized coloring algorithm where per round each node chooses a random color from its palette and keeps it, if none of the neighbors (or sometimes neighbors in certain subsets) chose that color. Variants of this simple idea are standard in distributed coloring algorithms, perhaps the first appearance was in the work of [13].
7. A routing method of Lenzen [15], which by now has become a staple in CONGESTED-CLIQUE algorithms. It is worth noting that the topology gathering ingredients mentioned in items 2 and 3 above also make use of this routing method.

From a bird's eye viewpoint, our algorithm works as follows: we use the random partitioning of item (5) to break the graph into several graphs of much lower arboricity, plus some additional edges. We use the oppurtunistic topology-gathering of item (3) to learn some sufficent local topology in each of these subgraphs, so that we can locally simulate a non-constant number of iterations of the peeling algorithm of item (1) and then apply a

suitable variant of the randomized coloring algorithm of item (6). This will be only a partial coloring, but the number of nodes that remain uncolored will be considerably smaller. Then, we can apply the sampling idea of item (5) and gather the sampled sparse graph, using the routing method of (7), so that we can solve the problem in a centralized fashion, and make even more progress in reducing the number of nodes that remain uncolored. This number at the end will be so small that we can gather all the induced edges by the remaining nodes, using the routing of item (7), and solve the problem in a centralized way. There are several subtleties in this rough outline; we leave the details to the technical sections.

Roadmap. We start with a warm up in Section 2 where we present an $O(\log\log\log n)$ round algorithm for forest-decomposition. Of course, that $O(\log\log\log n)$ complexity is much higher than our claimed bound of $O(1)$ round complexity, but we can introduce a number of the key algorithmic ideas that we use in the context of this simpler but slower algorithm. Then, in Section 3, we present our $O(1)$-round algorithm for $O(\alpha)$-coloring.

2 Warm up: Forest Decomposition in $O(\log\log\log n)$ rounds

In this section, as a warm up, we describe an algorithm for $(2+O(\epsilon))\alpha$ forest decomposition for graphs with $\alpha \leq n^{1-O(\frac{(\log\log n)^2}{\log n})}$, in $O(\log\log\log n)$ rounds of the congested clique model, for any desirably small constant $\epsilon > 0$. This allows us to introduce some of the basics of our end-results, in a simpler setting.

Studying $\alpha = O(\log n)$ Suffices. Before starting our forest-decomposition algorithm, we comment that almost without loss of generality, we can assume that $\alpha = O(\log n/\epsilon^2)$. The reason is as follows: suppose that $\alpha = \Omega(\log n)$. Partition the edges into $k = \epsilon^2\alpha/(10\log n)$ parts $E_1, \ldots, E_k$, randomly. Then, as we show in a simple lemma below, we can see that each group is a subgraph with arboricity $\alpha' = (1+\epsilon)\alpha/k \leq \Theta(\log n)$, with high probability. We describe an algorithm below, which when applied to each of these subgraphs, it decomposes each such subgraph into $(2+\epsilon)\alpha'$ edge-disjoint forests. Thus, overall, the graph is decomposed into $k \cdot (2+\epsilon)\alpha' = (2+\epsilon)\alpha$ forests. We will remark at the end of this section why we have sufficient communication capacity to perform this algorithm on all the subgraphs in parallel.

▶ **Lemma 3.** *If we randomly partition edges of a graph with arboricity $\alpha = \Omega(\log n/\epsilon^2)$ into $k = \epsilon^2\alpha/(10\log n)$ parts $E_1, \ldots, E_k$, then each part is a subgraph with arboricity at most $\alpha' = (1+\epsilon)\alpha/k$, with high probability.*

Proof. We focus on one part E_i. Consider an arbitrary subset of vertices $V' \leq V$, where $|V'| \geq 2$. The expected number of edges of V' that would be put in E_i at most $E(V')/k \leq (|V|'-1)\alpha/k$ edges. By a basic Chernoff bound, the probability that there are more than $(|V|'-1)\alpha/k(1+\epsilon)$ edges is at most $2exp(-(|V|'-1)\alpha/(3k)) \leq exp(-(|V'|\log n)$. Now, we can use a union bound over all possible subset of size $|V'|$, for which there are at most $\binom{n}{|V'|} \ll 2^{|V'|\log n}$ possibilities, and conclude that the edge density of each of them is at most α', with probability $1 - exp(-\Theta(|V'|\log n)) \geq 1 - 1/poly(n)$. Finally, a union bound over all the n possibilities for the value $|V'|$ and at most $\alpha \leq n$ possibilities for index i, in part E_i, concludes the proof. ◀

We will next describe our forest decomposition algorithm for graphs with arboricity $\alpha = O(\log n)$. The algorithm we present in this section will provide a stronger structure, called *H-partition* by Barenboim and Elkin [5], defined as follows: The set V of nodes will

be partitioned into $L = O(\log n)$ disjoint sets V_1, V_2, ..., V_L, which we will call *layers* of the H-partition, such that for each $i \in \{1, 2, \ldots, L\}$, each node $v \in V_i$ has at most $(2+\epsilon)\alpha$ neighbors in $\cup_{j=i}^{L} V_j$.

Given such an H-partition, one can easily obtain a forest-decomposition, i.e., decompose the graph into edge-disjoint forests $F_1, \ldots F_K$, where $K = (2+\epsilon)\alpha$, as follows: (A) orient each edge $\{v, u\}$ between two disinct sets V_i and V_j where $i \neq j$ from the lower index one to the higher indexed one, i.e., from V_i to V_j iff $i < j$. (B) orient each edge $\{v, u\}$ where both endpoints are in the same layer V_i from the lower ID node to the higher ID node. Then, clearly each node has out-degree at most K. Each node v puts each of its outgoing edges in a distinct one of F_1 to F_k. Hence, each F_i is a directed acyclic graph with out-degree at most one. Thus, if we ignore the directions, F_i is a forest. See section 5.1 of the book by Barenboim and Elkin[3], for an elaborate proof.

Our H-partition Algorithm. We build our $O(\log\log\log n)$-round H-partition (and therefore forest decomposition) algorithm in two steps. First, in Section 2.1, we present a slower algorithm that computes H-partition in $O(\log\log n)$ rounds, with high probability. Then, in Section 2.2, we explain a topology-gathering idea (stated in Lemma 9), which allows us to improve this round complexity to $O(\log\log\log n)$, thus proving the following theorem.

▶ **Theorem 4.** *There is a randomized distributed algorithm that for any n-node graph with arboricity $\alpha = O(\log n)$ computes H-partition with parameter $(2+\epsilon)\alpha$ in $O(\log\log\log n)$ rounds with high probability.*

2.1 H-partition in $O(\log\log n)$

We now present our slow $O(\log\log n)$ round algorithm for H-partition. This algorithm relies on two key ingredients: (I) an iterative peeling algorithm of Barenboim and Elkin [5], which on its own would need $R = \Theta(\log n)$ rounds to build the H-partition. (II) A random edge sampling idea which will allow us to compress the last $R - O(\log\log n) = O(\log n)$ rounds of the peeling algorithm into just $O(1)$ rounds of the congested clique.

High-Level Outline. First by running $O(\log\log n)$ iterations of the peeling algorithm of Barenboim and Elkin [5], we reduce the number of nodes to $O(\frac{n}{\log^2 n})$. Then, we use an randomized edge sampling process which allows us to generate a graph with just $O(n)$ edges, using which we can perform the rest of the peeling rounds. Since this graph has just $O(n)$ edges, we can deliver it to a single node, using Lenzen's routing method [15], and thus perform all these remaining peeling rounds in a centralized fashion there, without having to use more rounds of communication in the congested clique model.

Part 1 – Slow Iterative Peeling. First, we perform $O(\log\log n/\epsilon)$ iterations. In iteration i, we remove all nodes whose degree in the remaining graph is at most $(2+\epsilon)\alpha$ and we put them in layer i. Then, we proceed to the next iteration. A more formal description can be found in Algorithm 1.

▶ **Lemma 5.** *After $2\log\log n \cdot \frac{2}{\epsilon}$ iterations of peeling, at most $\frac{n}{\log^2 n}$ nodes remain.*

Proof. After each iteration, the number of remaining nodes reduces by a factor of $\frac{2}{2+\epsilon}$. This is because any remaining graph has arboricity at most α, which means it has average degree at most 2α, and thus only $\frac{2}{2+\epsilon}$ fraction of its nodes can have degree higher than $(2+\epsilon)\alpha$ and remain for the next iteration. Hence, after $2\log\log n \cdot \frac{2}{\epsilon}$ iterations, the number of remaining nodes is at most $(\frac{2}{2+\epsilon})^{\lceil 2\log\log n \cdot \frac{2}{\epsilon} \rceil} n \leq \frac{n}{\log^2 n}$. ◀

Algorithm 1 Computing a partial H-partition for a graph G with arboricity α.

```
1: procedure H-PARTITION((α, ϵ))
2:     An algorithm for each vertex v of G:
3:     i = 1
4:     while i ≤ ⌈(4/ϵ) log log n⌉ do
5:         if v is active and has at most (2 + ϵ)α active neighbors then
6:             make v inactive
7:             add v to H_i
8:             send the message "inactive" and "v joined H_i" to all the neighbors
9:         for each received "inactive" message do
10:            mark the sender neighbor as inactive
11:        i = i + 1
```

Part 2 – Speeding up Peeling via Random Edge Sampling. Next we focus on the remaining nodes and we find an H-partition of them in constant rounds of the congested clique. Let N denote the number of remaining nodes, and notice that we know $N \leq \frac{n}{\log^2 n}$. Consider $O(\log n)$ independent sampling process where in each, each edge of the graph induced by the remaining nodes is sampled with probability $p = min(\frac{1000 \log n}{\epsilon^2 a}, 1)$. Let G_i be the graph with sampled edges in i^{th} process. We will show in Lemma 6 that the total number of sampled edges is $O(n)$, with high probability. Thus, using Lenzen's routing method [15], we can collect all sampled edges in one node in $O(1)$ rounds. Then, we devise an iterative processing algorithm to compute H-partition of the remaining nodes, by processing only these sampled edges (instead of working on the base graph). We note that this iterative process will happen in a centralized fashion in the node that holds all the sampled edges, and thus it needs no communication. At the end, we can report to each node of the graph the peeling iteration in which it was removed. The peeling algorithm based on the randomly sampled edges is described in Algorithm 2. We note that for each iteration, this algorithm uses a different randomly sampled subgraph G_i, and these samplings are performed independent of each other. We will show that even though we are working on a randomly sampled subgraph, our peeling process will still produce a correct H-partition, in the following senses: On one hand, in Lemma 7, we show that in each iteration, all nodes of degree less than $(2 + \frac{\epsilon}{2})\alpha$ in the remaining graph will be removed, with high probability. Hence, we will be done after $O(\log n)$ iterations of peeling. On the other hand, in Lemma 8, we show that any node that gets removed in an iteration has degree at most $(2 + 2\epsilon)\alpha$, with high probability, among the nodes that had remained for that iteration. Hence, each node has at most $(2+2\epsilon)\alpha$ neighbors in its own layer or the future layers. We next present these lemmas and their proofs.

We first show that the number of sampled edges is small enough to allow us to deliver them to one node, via Lenzen's routing.

▶ **Lemma 6.** *The total number of sampled edges is at most $O(n)$, with high probability.*

Proof. The number of remaining nodes is $N \leq \frac{n}{\log^2 n}$. We have $O(\log n)$ independent sampling process, where in each process, each edge is sampled with probability $p = min(\frac{1000 \log n}{\epsilon^2 a}, 1)$. Hence, the expected number of sampled edges is $O(n)$. Moreover, since the samplings are independent (across different edges and different runs of the process), we can apply the Chernoff bound and conclude that with high probability, at most $O(n)$ edges are sampled. ◀

Now, we argue that in each iteration of peeling on the sampled subgraph, we will remove all nodes of relatively small degree in the base graph (among remaining nodes).

Algorithm 2 Computing H-partition for the remaining nodes by processing the sampled subgraphs G_i for $i \in 1, \ldots, \Theta(\log n)$.

```
1: procedure PROCESSING SAMPLED GRAPHS(G_i)
2:     i = 1
3:     while i ≤ Θ(log n) do
4:         for for each vertex v of G_i do
5:             if v has at most (2 + ε)αp neighbors in G_i then
6:                 add v to H_{i+T}
                                ▷ T=Θ(log log n) is the number of iterations in Algorithm 1
7:                 delete v from all G_k for k ∈ {i + 1, ..., Θ(log n)}
8:         i = i + 1
```

▶ **Lemma 7.** *In each iteration of working with the sampled subgraphs, all nodes of degree less than* $(2 + \frac{\epsilon}{2})\alpha$ *in the remaining base graph will be removed, with high probability.*

Proof. Consider iteration i. Let G' be the subgraph of the base graph induced by the nodes that remain in iteration i. Consider a remaining node v and suppose that its degree $d_{G'}(v) \leq (2 + \frac{\epsilon}{2})\alpha$. We argue that such a node will be remove in this iteration, with high probability. Then, in the sample subgraph for process i , each of these edges is sampled with probability p. Thus, the expected number of the sampled edges of v is $(2 + \frac{\epsilon}{2})\alpha p$. Since the sampling for this process is independent of the randomness of the previous processes, and since different edges are sampled independently, we can apply a Chernoff bound and conclude that the number of sampled edges of v is at most $(2 + \epsilon)\alpha p$, with high probability. Hence, node v gets removed in this iteration. ◀

As mentioned before, Lemma 7 allows us to argue that after each iteration the number of nodes reduces by a factor of $\frac{2}{2+\frac{\epsilon}{2}}$ with high probability. Thus, after $O(\log n)$ iterations, the number of nodes that did not find their proper partition is below 1, i.e., no node remains. We also need to ensure that the peeling performs the correct thing in that any node that gets removed in some iteration has degree at most $(2 + 2\epsilon)\alpha$ in the base graph, among the remaining nodes.

▶ **Lemma 8.** *In each iteration of working with the sampled subgraphs, any node that gets removed has degree at most* $(2 + 2\epsilon)\alpha$ *in the base graph among the nodes that remained for that iteration, with high probability.*

Proof. Consider iteration i. Let G' be the subgraph of the base graph induced by the nodes that remain in iteration i. Consider a remaining node v and suppose that its degree $d_{G'}(v) \geq (2 + 2\epsilon)\alpha$. We argue that such a node will be not be removed in this iteration, with high probability. In the sample subgraph for process i , each of these edges is sampled with probability p. Thus, the expected number of the sampled edges of v is at least $(2 + 2\epsilon)\alpha p$. Since the sampling for this process is independent of the randomness of the previous processes, and since different edges are sampled independently, we can apply a Chernoff bound and conclude that the number of sampled edges of v is strictly exceeds $(2 + \epsilon)\alpha p$, with high probability. Hence, node v does not get removed in this iteration. ◀

To conclude, as argued above Lemma 7 implies that we are done in $O(\log n)$ iterations of peeling, and Lemma 8 shows that each node that gets removed in each iteration of peeling has degree at most $(2 + 2\epsilon)\alpha$ among the nodes that had remained for that iteration. That is,

the nodes peeled in different iterations gives a partitioning of the nodes so that each node in a given layer has at most $(2+2\epsilon)\alpha$ neighbors in that layer or the future ones. This gives us an H-partition with out-degree $(2+2\epsilon)\alpha$.

2.2 H-partition in $O(\log\log\log n)$ rounds

In this section, we explain how to speed up the $O(\log\log n)$-round algorithm described in the previous subsection to run in $O(\log\log\log n)$ rounds.

Recall that our previous algorithm had two main part, a slow $O(\log\log n)$ round for iterative peeling, and a fast $O(1)$ round algorithm where all the other peelings happen in constant rounds of the congested clique, by processing a sparse randomly sampled subgraph of the remaining graph. Here, we focus on the first part and improve its complexity to $O(\log\log\log n)$-round, using a topology-gathering idea.

Notice that the first part of the algorithm did not need any all-to-all communication. It was simply iterations of peeling in the graph. Hence, if a node v knew all of the topology within its R hops, it could simulate this peeling process for R rounds. We can learn this R-hop topology much faster than R rounds in the congested clique model, thanks to the all-to-all communication, but subject to some constraint: it is only possible if the topology to be learned is relatively small. In particular, we next discuss a special case of such a topology learning on graphs where the maximum degree is small. Later, we will discuss how to incorporate this maximum degree limitation into our peeling algorithm.

First, we start with a generic topology-gathering lemma, for small-degree graphs. Variants of such a statement have been used in prior work, see [16, 8], for instance.

▶ **Lemma 9.** *Suppose that F is graph with n nodes and maximum degree at most $\log^{100} n$. Then we can make each node $v \in F$ learns its R-hop neighborhood in F for $R = O(\log\log n)$, in $O(\log R) = O(\log\log\log n)$ rounds in congested clique.*

Proof. We use an induction to prove that for any $k \leq O(\log\log\log n)$, in $O(k)$ rounds, we can make the nodes learn the graph $F' = F^{2^k}$, which is formed by connecting each two nodes that are within distance 2^k. After that, each node can directly receive all the edges of all of its F' neighbors and thus know the whole topology within its 2^k-hop neighborhood, using Lenzen's routing[15]. The base case of induction where $k = 1$ is trivial, as that is knowing the graph F itself. Suppose that assume that $O(k)$ rounds have passed and each node already knows its neighbors in the graph F^{2^k}. Now, the number of neighbors in this graph is at most $(\log^{100} n)^{(2^k)} \leq 2^{O((\log\log n)^2)} \ll \sqrt{n}$. Make each node send the names of each of its neighbors in F^{2^k} to each of its neighbors in F^{2^k}. That is at most n messages. Hence, this information can be routing in $O(1)$ round using Lenzen's routing. Then, each node knows its neighbors of neighbors in F^{2^k}, which means it can know it knows all of its neighbors in $F^{2\cdot 2^k} = F^{2^{k+1}}$. ◀

Applying Topology-Gathering to Speed Up Peeling. If we could apply the above topology gathering to our $O(\log\log n)$ round peeling algorithm, we would compress it to $O(\log\log\log n)$ rounds. However, this is not immediately possible. Our original graph may have nodes of very large degree, much larger than $poly(\log n)$, despite its small arboricity $\alpha = O(\log n)$. This means we cannot directly apply the topology gathering idea of the above lemma. However, fortunately, our graph cannot have too many nodes of large degree (exactly because of its small arboricity). This allows us to freeze those few high degree nodes, and still use a peeling process on the rest of the graph to reduce the number of remaining nodes (including the frozen ones) to just $O(\frac{n}{\log^2 n})$.

In particular, freeze each node whose degree is greater than $\log^{100} n$. Notice that since the graph has arboricity $\alpha = O(\log n)$, as we justified this assumption at the beginning of this section and in Lemma 3, the number of frozen nodes is at most $\frac{2n\alpha}{\log^{100}(n)} \leq \frac{2n}{\log^{99} n}$. Now, we apply $O(\log \log n)$ iterations of the peeling algorithm on the remaining nodes, but sped up to $O(\log \log \log n)$ rounds of the congested clique model, using the topology-gathering approach of Lemma 9. The number of nodes that remain from this peeling is $O(n/\log^2 n)$. Hence, even including the at most $\frac{2n}{\log^{99} n}$ frozen nodes, the number of remaining nodes is $O(n/\log^2 n)$. That means, we can now apply the algorithm of the second part, described in the previous subsection, to finish all of the peeling of the H-partition in $O(1)$ rounds of the congested clique (via working on the sampled subgraphs).

Running the Algorithm in Different Parts with Arboricity $O(\log n)$

At the beginning of the section, we explained that we partition graphs with arboricity higher than $\Theta(\log n)$ into many $\alpha/\Theta(\log n)$ subgraphs, each with arboricity $\Theta(\log n)$, and we then decompose each subgraph separately, using the procedure described above. It remains to explain why we can run the algorithm on all these subgraphs simultaneously. Notice that the local communications performed for peeling are in the edges of the subgraph, and therefore those can be performed in parallel. When performing a local topology gathering, each node needs to send or receive $2^{O((\log \log n)^2)}$ bits of information. Hence, even over all the $\alpha/\Theta(\log n)$ subgraphs, the total information that each each node needs to send or receive is $O(n)$, as we have assumed $\alpha \leq n^{1-O(\frac{(\log \log n)^2}{\log n})}$. After that, for the final step of the algorithm, we just need to bring a graph of size $O(n)$ to some leaders node and solve the problem of remaining nodes there, for each subgraph. It suffices to use different such leaders for different subgraphs.

3 Constant-Time Coloring

In this section, we provide our proof of Theorem 1, thus showing a randomized distributed algorithm that colors any graph of arboricity at most α using $O(\alpha)$ colors, w.h.p.

Proof of Theorem 1. The proof has several steps. We start with a high-level outline: we will first argue that it suffices to work with graphs of arboricity at most $O(\log n)$, because higher-arboricity graphs can be broken to this case easily, via randomness. Second, we set aside nodes of degree higher than α^{10}, to color them only later using some other fresh $O(\alpha)$ colors. Then, we explain how we partition any graph with arboricity $\alpha = O(\log n)$ to β/α subgraphs, each being a graph with arboricity almost $\beta = poly(\log \alpha)$ plus a small number of extra edges. Then, we explain how we color almost the entirety of each of these subgraphs, using different colors for different subgraphs. For this partial coloring, we will first explain a slow algorithm in the CONGEST model and then discuss how to simulate it in $O(1)$ rounds of the CONGESTED-CLIQUE model. The nodes that remain uncolored will be so few that we can gather the topology induced by them and color them using some 2α extra colors. We next explain these steps separated by paragraph titles, to reflect the above outline.

Studying $\alpha = O(\log n)$ Suffices. We focus on the case where $\alpha = O(\log n)$. Coloring of graphs with $\alpha = \Omega(\log n)$ arboricity can be transformed easily to this $O(\log n)$-arboricity case: place each node in a randomly chosen one of $k = \alpha/\Theta(\log n)$ parts. Each part will induce a subgraph with arboricity $\Theta(\log n)$, with high probability. The reason is that the original graph with arboricity α admits an orientation with out-degree 2α. Fix such an orientation. We do not need to know this orientation but use it for our analysis, only. When we randomly

partition nodes, we expect each node's out-degree in its own part to be $2\alpha/k = \Theta(\log n)$. Hence, by applying a Chernoff bound, each node's outdegree in this imagined orientation is at most $3\alpha/k$. Thus, each of the subgraphs admits an orientation with out-degree at most $3\alpha/k$, which means it has arboricity at most $3\alpha/k$.

We color each of these parts using $\Theta(\log n)$ separate colors, all in parallel. For each part, we apply the algorithm for graphs with arboricity at most $O(\log n)$, which we describe next.

Setting aside nodes with degree $\Omega(\alpha^{10})$. As a preparation step, we set aside all nodes which have degree at least α^{10}. We note that a graph of arboricity α can have at most $2n/\alpha^9$ such nodes, simply because it has at most $n\alpha$ edges. This number of left-over nodes is small enough that allows us to deal with coloring these nodes later, using some fresh colors.

Partitioning to lower-arboricity subgraphs, plus few extra edges. We now have a graph $G = (V, E)$ with at most n nodes, arboricity $\alpha = O(\log n)$, and maximum degree at most α^{10}. We randomly partition V into $k = \frac{\alpha}{\log^3 \alpha}$ parts $V_1, V_2, \ldots, V_k$, by placing each node in a random part V_i. In Claim 10, we show that, with high probability, the subgraph $G[V_i]$ induced by the nodes in each part V_i is a graph with arboricity $(2+\epsilon)\log^3 \alpha$, plus at most $n/(\alpha)^{10}$ extra edges, for an arbitrarily small constant $\epsilon > 0$. We partially color each part $G[V_i]$ separately, using $O(\log^3 \alpha)$ different colors. This partial coloring will leave some $O(n/\alpha^5)$ nodes uncolored per part. In fact, the partial coloring will be done in two steps, one leaves at most $O(n/\log^3 \alpha)$ uncolored nodes, and the second reduces the number of uncolored nodes to $O(n/\alpha^5)$. At the very end, we will then gather the subgraph induced by all these remaining nodes over all parts – which has at most $k \cdot O(n/\alpha^5) \cdot \alpha = O(n)$ edges – in one node and color them using 2α extra colors.

▷ Claim 10. The subgraph $G[V_i]$ induced by each part V_i is a graph with arboricity $(2+\epsilon)\log^3 \alpha$, plus at most $n/(\alpha)^{10}$ extra edges, for an arbitrarily small constant $\epsilon > 0$.

Proof of Claim 10. Consider a hypothetical orientation of the graph with out-degree 2α and call a node v bad if in the subgraph $G[V_i]$ to which v belongs, its outdegree exceeds $2\alpha/k(1+\epsilon) = (2+\epsilon)\log^3 \alpha$. Notice that the probability of a node being bad is exponentially small in its expected outdegree, i.e., it is at most $exp(-\Theta(\alpha/k)) \leq exp(-\log^2 \alpha)$. Hence, the expected number of nodes that are bad is at most $nexp(-\log^2 \alpha)$. We would like to say that, with high probability, the number of bad nodes is at most $n \cdot exp(-\log^2 \alpha)$. We cannot directly apply the Chernoff bound, because the events of different nodes being bad are not independent of each other. However, the events are independent for any two nodes that do not share a common neighbor. Since we are now on a graph with maximum degree at most α^{10}, as we have set aside nodes of higher degree, we can infer that the even of each node being bad depends on the events of at most $d = \alpha^{20}$ many other nodes – all those within 2 hops. Hence, we can apply the extension of Chernoff to the setting with a bounded degree of dependency[24]. In particular, we get that the probability that the number of bad nodes exceeds $\mu = n \cdot exp(-\log^2 \alpha)$ by more than a constant factor is at most $\Theta(d) \cdot exp(-\Theta(\mu/d)) = \Theta(\alpha^{10}) \cdot exp(-\Theta(n \cdot exp(-\log^2 \alpha)/\alpha^{20})) \ll 1/poly(n)$. Hence, with high probability, the number of bad nodes is less than $\Theta(n \cdot exp(-\log^2 \alpha))$. A node that is bad introduced at most α^5 bad edges, in $G[V_i]$, because it is incident on at most α^{10} edges. Thus, except for at most $n \cdot exp(-\log^2 \alpha) \cdot \alpha^{10} \ll n/(\alpha)^{10}$ edges incident to bad nodes, all other edges can be oriented with out-degree $(2+\epsilon)\log^3 \alpha$. Hence, the subgraph is graph with arboricity $\beta = (2+\epsilon)\log^3 \alpha = O((\log\log n)^3)$, plus at most $n/(\alpha)^{10}$ extra edges. ◁

Outline for First Partial Coloring of Each Part. We now focus on one subgraph $H = G[V_i]$ which is a subgraph with arboricity $\beta = (2+\epsilon)\log^3 \alpha = O((\log\log n)^3)$, plus at most $n/(\alpha)^{10}$ extra edges. We next describe an algorithm $\mathcal{A}$ that runs in $O(\log^2 \beta)$ rounds of the CONGEST model and colors all but $O(n/\beta)$ many of the nodes of H, using $O(\beta)$ colors. Then, we explain how we simulate $\mathcal{A}$ in just $O(1)$ rounds of the CONGEST-CLIQUE model.

A CONGEST Algorithm for Partial Coloring of One Part. We focus on one part $H = G[V_i]$. First, we put aside all nodes of degree greater than β^2. The number of such nodes is $O(n/\beta)$. Then, we attempt to color most of the rest, in $O(\log^2 \beta)$ rounds, as follows. First, for $\ell = 40\log\beta$ iterations, in each iteration j, we remove nodes of degree at most 2.1β and put them in *layer* L_j. Since the graph has arboricity β, in each iteration at least a constant fraction of nodes get removed. Thus, We can show that the number of remaining nodes after $40\log\beta$ iterations is $O(n/(\beta)^5)$. Then, we attempt to color the nodes in layers L_ℓ, $L_{\ell-1}, \ldots L_1$ using 2.2β colors. We spend one phase to color each layer layer, as follows.

We now describe each phase, which is simply $10\log\beta$ repetitions of a simple randomized coloring attempt. In each round of the j^{th} phase, each node of layer L_j picks a color at random from 2.2β colors and keeps it if it is different than the colors of its neighbors in layers $L_{j'}$ for $j' \geq j$. A node that did not choose such a color remains uncolored in this round. After $10\log\beta$ rounds, all nodes of L_j that remain uncolored are removed. Per round, each node gets colored with at least $0.1\beta/2.2\beta = 0.02$, regardless of the choices of its other up to 2.1β neighbors. Hence, the probability of a node remaining uncolored after $10\log\beta$ rounds is at most β^5. Moreover, the events of different nodes remaining uncolored is independent, because in the above argument we only relied on the randomness in the color choices of the node itself. We can conclude that, with high probability, the number of these nodes in L_j is $O(n/(\beta)^2)$. We then proceed to the next phase. Over all the phases, the number of nodes in L_j that remain uncolored is at most $O(n\log\beta/(\beta)^2)$. Moreover, the number of nodes that were not in any of the layers is at most $O(n/(\beta)^2)$. Hence, overall, this process leaves at most $O(n/\beta)$ nodes of H uncolored.

A small final note about algorithm $\mathcal{A}$ is that each node uses at most $O(\log^2 \beta) \ll O(\log n)$ bits of randomness, $O(\log\beta)$ many bits in each round where it picks a random color from $\{1, 2, \ldots, 2.2\beta\}$. We will use this fact about the amount of randomness, later, when we speed up the algorithm in the CONGESTED-CLIQUE model.

CONGESTED-CLIQUE Algorithm for Partial Coloring of One Part. We focus on one part $H = G[V_i]$ and show how we mimic the algorithm described above, in just $O(1)$ rounds of CONGESTED-CLIQUE. As before, we put aside all nodes of degree greater than β^2. We next try to simulate $\mathcal{A}$, using a randomized information gathering approach.

We now use an opportunistic information gathering to compress the round complexity in CONGESTED-CLIQUE to $O(1)$. Concretely, we make each remaining node v send a message to all other nodes, where the message contains the name of v, its degree, and the at most $O(\log n)$-bits of randomness that v uses in algorithm $\mathcal{A}$. Besides the above, each v also sends each of its edges to each other node, but only randomly: each edge is sent to each node with probability $1/\beta^2$, all independent of each other. Notice that the total number of edges that are to be sent to each node is at most $O(n)\cdot\beta^2/\beta^2 = O(n)$, with high probability.

We say a node u successfully received the *relevant ball* of node v – with respect to algorithm $\mathcal{A}$ – if node u received all the edges incident on all nodes w within distance $\Theta(\log^2\beta)$ of v. The number of such edges is at most $\beta^{\Theta(\log^2\beta)} \leq 2^{\Theta(\log^3\beta)}$. The probability of each of them being sent to u is $1/\beta^2$. Thus, the probability that they are all sent to u is $(1/\beta^2)^{2^{\Theta(\log^3\beta)}} \geq 2^{-2\log\beta\cdot 2^{\Theta(\log^3\beta)}} \geq 2^{-2^{\Theta(\log^3\beta)}} \gg 2^{-\sqrt{\log n}}$, given that $\log\beta = \Theta(\log\log\log n)$. Since each

node u successfully receives the ball of v with probability at least $2^{-\sqrt{\log n}}$, and as there are n possibilities for u, with high probability, there is at least one node u that successfully receives the ball of v. Notice that node u realizes this successful event as it knows the degree of each node and it can distinguish whether it has received all of the edges or not. Once u has received the relevant ball of node v, node u can simulate the CONGEST-model algorithm $\mathcal{A}$ and inform node v about the status of node v at the end of $\mathcal{A}$. This status is simply whether v is colored or not, and if yes, with which color.

Notice the small subtlety that there might be many nodes like u that receive the ball. However, since they all use the same randomness to simulate the behavior of each node in this ball (the randomness received from that node), they all get the same output for v.

Second Step of Partial Coloring. Finally, we go back to those $O(n/\beta)$ nodes that remain uncolored and we color enough of them that the number of uncolored nodes reduces to $O(n/\alpha^5)$. Given that H is a graph with arboricity at most β plus some $O(n/\alpha^{10})$ edges, the number of edges induced by the remaining $O(n/\beta)$ nodes is at most $O(n)$. That means we can move all these edges to one node . Then, we run a process similar to algorithm $\mathcal{A}$ on these remaining nodes, but in a centralized fashion. In particular, in $O(\log \alpha)$ iterations, we remove nodes of degree at most 2.1β. Then, we color these removed nodes greedily from the last layer to the beginning, using 2.2β extra colors. The only nodes that are not colored are those that are not in any of these $O(\log \alpha)$ layers. Since each layer of peeling and coloring removes at least a $0.1/2.2$ fraction of nodes (similar to the previous arguments), after $O(\log \alpha)$ such iterations, the number of nodes that remain uncolored is at most $O(n/\alpha^5)$ many.

Final Step, Coloring the Remaining Nodes. Finally, we handle these remaining nodes of different parts. In each of the parts $G[V_i]$, we have $O(n/\alpha^5)$ remaining nodes. Moreover, we had some $O(n/\alpha^9)$ nodes that we set aside at the very beginning, because they had degree greater than α^{10}. We collect the subgraph induced by all remaining nodes into one node, which has no more than $O(n/\alpha^4)$ edges, and we color them using 2α new colors. ◀

References

1. Leonid Barenboim and Michael Elkin. Sublogarithmic distributed MIS algorithm for sparse graphs using Nash-Williams decomposition. *Distributed Computing*, 22(5-6):363–379, 2010.
2. Leonid Barenboim and Michael Elkin. Deterministic distributed vertex coloring in polylogarithmic time. *Journal of the ACM (JACM)*, 58(5):23, 2011.
3. Leonid Barenboim and Michael Elkin. Distributed Graph Coloring: Fundamentals and Recent Developments. *Synthesis Lectures on Distributed Computing Theory*, 4(1):1–171, 2013.
4. Leonid Barenboim, Michael Elkin, and Uri Goldenberg. Locally-Iterative Distributed $(\Delta+1)$:-Coloring below Szegedy-Vishwanathan Barrier, and Applications to Self-Stabilization and to Restricted-Bandwidth Models. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 437–446. ACM, 2018.
5. Leonid Barenboim and Victor Khazanov. Distributed Symmetry-Breaking Algorithms for Congested Cliques. In *International Computer Science Symposium in Russia*, pages 41–52. Springer, 2018.
6. Yi-Jun Chang, Manuela Fischer, Mohsen Ghaffari, Jara Uitto, and Yufan Zheng. The Complexity of $(\Delta + 1)$ Coloring in Congested Clique, Massively Parallel Computation, and Centralized Local Computation. *Manuscript*, 2019.
7. Yi-Jun Chang, Wenzheng Li, and Seth Pettie. An Optimal Distributed $(\Delta + 1)$-coloring Algorithm? In *Proc. of the Symp. on Theory of Comp. (STOC)*, pages 445–456, 2018.
8. Mohsen Ghaffari. Distributed MIS via all-to-all communication. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 141–149. ACM, 2017.

9 Mohsen Ghaffari. Distributed Maximal Independent Set using Small Messages. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 805–820. SIAM, 2019.

10 Mohsen Ghaffari and Christiana Lymouri. Simple and Near-Optimal Distributed Coloring for Sparse Graphs. In *31st International Symposium on Distributed Computing (DISC 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

11 Mohsen Ghaffari and Merav Parter. MST in log-star rounds of congested clique. In *Proc. of the Symp. on Princ. of Dist. Comp. (PODC)*, pages 19–28, 2016.

12 James W. Hegeman and Sriram V. Pemmaraju. Lessons from the Congested Clique Applied to MapReduce. *Theor. Comput. Sci.*, 608(P3):268–281, December 2015.

13 Öjvind Johansson. Simple distributed $\Delta+$ 1-coloring of graphs. *Information Processing Letters*, 70(5):229–232, 1999.

14 Tomasz Jurdziński and Krzysztof Nowicki. MST in O (1) rounds of congested clique. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2620–2632. SIAM, 2018.

15 Christoph Lenzen. Optimal deterministic routing and sorting on the congested clique. In *Proceedings of the 2013 ACM symposium on Principles of distributed computing*, pages 42–50. ACM, 2013.

16 Christoph Lenzen and Roger Wattenhofer. Brief announcement: Exponential speed-up of local algorithms using non-local communication. In *Proceedings of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 295–296. ACM, 2010.

17 Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing (SICOMP)*, 21(1):193–201, 1992.

18 Zvi Lotker, Boaz Patt-Shamir, Elan Pavlov, and David Peleg. Minimum-weight spanning tree construction in O (log log n) communication rounds. *SIAM Journal on Computing*, 35(1):120–131, 2005.

19 C St JA Nash-Williams. Decomposition of finite graphs into forests. *Journal of the London Mathematical Society*, 1(1):12–12, 1964.

20 Alessandro Panconesi and Aravind Srinivasan. Improved distributed algorithms for coloring and network decomposition problems. In *Proc. of the Symp. on Theory of Comp. (STOC)*, pages 581–592. ACM, 1992.

21 Merav Parter. (Δ+1) coloring in the congested clique model. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, 2018.

22 Merav Parter and Hsin-Hao Su. $(\Delta + 1)$ coloring in $O(\log^* \Delta)$ congested-clique rounds. In *Proceedings of the International Symposium on Distributed Computing (DISC)*, 2018.

23 David Peleg. *Distributed Computing: A Locality-sensitive Approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

24 Sriram V Pemmaraju. Equitable coloring extends Chernoff-Hoeffding bounds. In *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, pages 285–296. Springer, 2001.

Exploiting Hopsets: Improved Distance Oracles for Graphs of Constant Highway Dimension and Beyond

Siddharth Gupta
Ben-Gurion University of the Negev, Israel
siddhart@post.bgu.ac.il

Adrian Kosowski
Inria, Paris, France
adrian.kosowski@inria.fr

Laurent Viennot
Inria, Paris, France
Laurent.Viennot@inria.fr

Abstract

For fixed $h \geq 2$, we consider the task of adding to a graph G a set of weighted shortcut edges on the same vertex set, such that the length of a shortest h-hop path between any pair of vertices in the augmented graph is exactly the same as the original distance between these vertices in G. A set of shortcut edges with this property is called an *exact h-hopset* and may be applied in processing distance queries on graph G. In particular, a 2-hopset directly corresponds to a distributed distance oracle known as a *hub labeling*. In this work, we explore centralized distance oracles based on 3-hopsets and display their advantages in several practical scenarios. In particular, for graphs of constant highway dimension, and more generally for graphs of constant skeleton dimension, we show that 3-hopsets require *exponentially* fewer shortcuts per node than any previously described distance oracle, and also offer a speedup in query time when compared to simple oracles based on a direct application of 2-hopsets. Finally, we consider the problem of computing minimum-size h-hopset (for any $h \geq 2$) for a given graph G, showing a polylogarithmic-factor approximation for the case of unique shortest path graphs. When $h = 3$, for a given bound on the space used by the distance oracle, we provide a construction of hopset achieving polylog approximation both for space and query time compared to the optimal 3-hopset oracle given the space bound.

2012 ACM Subject Classification Theory of computation → Shortest paths; Theory of computation → Graph algorithms analysis; Theory of computation → Data structures design and analysis

Keywords and phrases Hopsets, Distance Oracles, Graph Algorithms, Data Structures

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.143

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Related Version A full version of the paper is available at `https://arxiv.org/abs/1803.06977`.

Funding *Siddharth Gupta*: Supported in part by the Zuckerman STEM Leadership Program.
Adrian Kosowski: Supported by ANR project DESCARTES (ANR-16-CE40-0023).
Laurent Viennot: Supported by ANR project DISTANCIA (ANR-17-CE40-0015) and ANR project MULTIMOD (ANR-17-CE22-0016).

1 Introduction

An exact h-hopset for a weighted graph G is a weighted edge set, whose addition to the graph guarantees that every pair of vertices has a path between them with at most h edges (*hops*) and whose length is exactly the length of shortest path between the vertices.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 143; pp. 143:1–143:15

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

The concept of a hopset was first explicitly described by Cohen [19] in its *approximate* setting, in which the length of h-hop path between a pair of vertices in the hopset should approximate the length of the shortest path in G. Hopsets were introduced in the context of parallel computation of approximate shortest paths. In this paper, we study hopsets in their exact version, with the general objective of optimizing exact shortest path queries.

Data structures which allow for querying distance between any pair of vertices of a graph have been intensively studied under the name of *distance oracles*. The efficiency of an exact distance oracle is typically measured by the interplay between the *space* requirement of the representation of the data structure and its *decoding time*. It is a well-established empirical fact that many real-world networks admit efficient (i.e., low-space and fast) distance oracles [6, 22]. A key example here concerns transportation networks, and specifically road networks, which are empirically known [34, 32, 5] to be augmentable by carefully tailored sets of shortcut edges, allowing for shortest-path computation. These sets of shortcuts may be hopsets (as is the case for the hub-labeling approach which effectively implements a 2-hopset), but may also be considered in some related (and frequently more involved) framework, such as contraction hierarchies [31] or transit-node routing [11].

An interesting theoretical insight due to Abraham et al. [3, 4, 5] provides theoretical bounds on the number of shortcuts required in all of the above-mentioned frameworks. They introduce a parameter describing the structure of shortest paths within ball neighborhoods of a graph, called *highway dimension* $\tilde{h}$. They also express the number of shortcuts that need to be added for each node so as to achieve shortest-path queries in a graph of n nodes with weighted diameter D as a polynomial of $\tilde{h}$, $\log n$, and $\log D$; this approach has been extended in subsequent work [2, 37]. The value of $\tilde{h}$ is known to be small in practice (e.g., typically $\tilde{h} < 100$ for continental-sized road networks [4]), and does indeed appear to be inherently linked to the size of the required shortcut sets. In fact, empirical tests have suggested that the (average) number of necessary shortcuts per node is in fact very close to $\tilde{h}$, laying open the question of whether the additional dependence of the number of shortcuts on logarithmic factors in n and D may be an artifact of the theoretical analysis of the oracles, which for each node require a separate shortcut for every "scale" of distance.

1.1 Results and Organization of the Paper

Our main result is to provide strong evidence that the dependence of the number of shortcuts on such logarithmic factors in n and D is indeed not essential, and we design a simple distance oracle based on a 3-hopset in which the number of shortcuts per node depends only on $\tilde{h}$, $\log\log n$, and the logarithm of the average edge length. This result is in fact shown in the framework of a strictly broader class of graphs, namely, graphs with a bounded value of a parameter known as *skeleton dimension* k ($k \leq \tilde{h}$), describing the width of the shortest-path tree of a node after pruning all branches at a constant fraction α of their depth. Considering various ranges of fraction α for increasing distance ranges was a novel key step for improving over [37, 36] from a 2-hopset construction to a 3-hopset construction.

From a general perspective, our connection between h-hopsets and distance oracles is original and offers new perspectives for studying the trade-off between size and query time of distance oracles. To exemplify this, we provide a construction of h-hopsets for graphs of treewidth t following a classical approach in pre-processing product queries on trees [7, 16]. For 3-hopsets, we obtain a distance oracle with quadratic dependency in t which improves over the construction of [15] (which has cubic dependency) for $t = \omega(\log^2 \log n)$. The space and time-bounds of oracles based on 3-hopsets are presented in Table 1, and compared with the corresponding parameters of oracles based on 2-hopsets. For the case of constant skeleton dimension or constant treewidth, we remark that using a 3-hopset instead of a 2-hopset reduces the number of shortcuts per node from $O(\log n)$ to $O(\log\log n)$ while achieving a query time of $O(\log^2 \log n)$.

Table 1 Comparison of distance oracles based on 2-hopsets (hub labeling [19, 28, 37]) and 3-hopsets (this paper). Size represents the number of shortcut edges in the hopset, i.e., the number of $O(\log n)$-bitsize words when measuring oracle size. The main results concern skeleton dimension and are stated in simplified form, assuming average edge length at most $O(\text{poly}\log n)$, with expected query times given for both types of oracles.

Distance oracle	Treewidth t		Skeleton dimension k	
	Size	*Time*	*Size*	*Time*
2-hopset (hubs):	$n \cdot O(t \log n)$	$O(t + \log\log n)$	$n \cdot O(k \log n)$	$O(k \log n)$
3-hopset:	$n \cdot O(t \log\log n)$	$O(t^2 \log^2 \log n)$	$n \cdot O(k \log k \log\log n)$	$O(k^2 \log^2 k \log^2 \log n)$

A classical assumption (applied, e.g., in almost all literature on transportation networks) resides in the uniqueness of shortest paths. It can be made without loss of generality by slightly perturbing the weights of the edges or by using appropriate tie break rules. In this context of *unique shortest path graph* (USP) graphs where there is a unique shortest path P_{uv} between any two nodes u and v, we propose an LP-based approximation algorithm for constructing h-hopsets with size within a polylog factor from optimal. Our construction can be seen as a non-trivial generalization of the prehub labeling introduced in [9] from 2 to more hops. In the case $h = 3$, we further extend our approach to provide an algorithm which constructs distance oracles in USP graphs based on 3-hopsets, with (approximate) optimality guarantees on size and query time. The form of guarantees we obtain is again novel: for a given size bound $\mathcal{S}$ of 3-hopset based oracle, we construct an oracle with size larger than $\mathcal{S}$ by at most a polylog factor which has average query time within a polylog factor of the performance achieved by the best oracle with size $\mathcal{S}$.

The rest of the paper is organized as follows. In Section 2, we introduce the necessary notions related to h-hopset and give a general approach for how a h-hopset can be used as a distance oracle, focusing on the special case of $h = 3$. In Section 3, we provide our first main result, using 3-hopsets to obtain improved (smaller and faster) distance oracles in graphs with bounded skeleton dimension. In Section 4, we present our second main result about approximating h-hopsets and constructing 3-hopset based oracles in USP graphs. Finally, Section 5, we show how to construct efficient h-hopsets and 3-hopset based oracles for bounded treewidth graphs. We provide full details of omitted and sketched proofs in the full version [33].

Our work is presented in the context of weighted undirected graphs, but all results can easily be extended to weighted directed graphs.

1.2 Other Related Work

Hopsets. Exact hopsets were implicitly constructed in the context of single-source shortest paths parallel computation [43, 35, 18, 40]. Such works study the work versus time trade-offs of such computation. Cohen [19] explicitly introduced the notion of (h, ϵ)-hopset of G as set H of weighted edges such that paths of at most h hops in $G \cup H$ have length within $(1 + \epsilon)$ of the corresponding shortest path in G. The parameter h is called the *hopbound*. For any graph G and $\epsilon, \epsilon' > 0$, she proposed a construction of $(O(\text{poly}\log n), \epsilon)$-hopset of G with size $O(n^{1+\epsilon'})$. More recently, Elkin et al. [24] proposed the construction of $(O(\epsilon^{-1}\log\kappa)^{\log\kappa}, \epsilon)$-hopset with $O(n^{1+1/\kappa}\log n \log\kappa)$ edges for any $\epsilon > 0$ and integral $\kappa \geq 1$. Abboud et al. [1] recently showed the optimality of the Elkin et al. [24] result. In particular, they showed that for any $\delta > 0$ and integer k, any hopset of size less than $n^{1+\frac{1}{2^{k+1}-1}-\delta}$ must have hop bound $h = \Omega(c_k/\epsilon^{k+1})$, where c_k is a constant depending only on k. The linear size case was then improved in [25]. As far as we know, exact hopsets (with $\epsilon = 0$) have not been explicitly studied. However, they are related to the following well studied notion.

Hopsets vs. TC-spanners. In directed graphs, a hopset can be seen as a special case of an *h-transitive-closure spanner (h-TC-spanner)*. Hopsets and TC-spanners are fundamental graph-theoretic objects and are widely used in various settings from distance oracles to pre-processing for range queries in sequential or parallel setting or even in property testing. The concept of adding transitive arcs to a digraph in order to reduce its diameter was introduced by Thorup [41] in the context of parallel processing. Bhattacharyya et al. [12] defined an h-TC-spanner of an unweighted digraph G as a digraph H with same transitive closure as G and diameter at most h. They note that this is a central concept in a long line of work around pre-processing a tree for range queries [7, 16, 42]. A TC-spanner can also be defined as a spanner (for the classical spanner definition [38]) of the transitive closure of a graph that has bounded diameter. We will see that an exact h-hopset defines a h-TC-spanner but that the converse is not necessarily true. Bhattacharyya et al. [12] proposed a construction of h-TC-spanner of size $O(n \log n \lambda_h(n))$ for H-minor-free graphs (where λ_h denotes the hth-row inverse Ackermann function, cf. Section 5).

Exact Distance Oracles. A long line of research studies the interplay between data structure space and query decoding time. A lot of attention has been given to distance oracles for planar graphs [23, 10, 17, 14, 26, 21, 30], and it has recently been shown that a distance oracle with $O(n^{1.5})$ space and $O(\log n)$ query-time is possible [30]. In the context of weighted directed graphs with treewidth t, Chaudhuri and Zaroliagis [15] propose a distance oracle using $O(t^2 n \lambda_h(n))$ space and $O(t^3 h + \lambda_h(n))$ query time for integral $h > 1$ where λ_h is the hth-row inverse Ackermann function (as defined in Subsection 5). In the context of unweighted graphs with treewidth t, Farzan and Kamali [27] obtain distance oracles with $O(t^3 \log^3 t)$ query time using optimal space (within low order terms). This construction heavily relies on the unweighted setting as exhaustive look-up tables are constructed for handling graphs with polylogarithmic size.

Distance Labelings and 2-Hopsets. The distance labeling problem is a special case of a distributed distance oracle, and consists of assigning labels to the nodes of a graph such that the distance between two nodes s and t can be computed from the labels of s and t (see, e.g., [28]).

The notion of 2-hopset studied in this work coincides with the special case of two-hop distance labeling (also called *hub-labeling*), where labels are constructed from hub sets: in hub-labeling, a small hub set $S(u) \subseteq V(G)$ is assigned to each node of a graph G such that for any pair u, v of nodes, the intersection of hub sets $S(u) \cap S(v)$ contains a node on a shortest $u - v$ path. Such a construction is formally proposed in [20] and is implicitly introduced by Gavoille et al. [28] and applied to graphs of treewidth t with labels of $O(t \log n)$ size and allows to answer distance queries in $O(t \log n)$ time; the hub sets have a hierarchical structure, which allows for an improvement of query time to $O(t \log \log n)$ time by a binary search over levels. Hub labelings are the currently best known distance labelings for sparse graphs, achieving sublinear node label size [8, 29], and may also be used to provide a 2-additive-approximation for distance labeling in general graphs using sublinear-space labels [29].

In graphs of bounded highway dimension, hub labels were among the first identified distance oracles to provide label size and query time polynomial in the highway dimension and polylogarithmic in other graph parameters [5]. This result was then extended to the more general class of graphs with bounded skeleton dimension [37, 36].

Hub sets with near to optimal size can be constructed in polynomial time. A greedy set cover-type $O(\log n)$-approximation algorithm (with respect to average size of a hub set) was proposed by Cohen et al. [20]. For the case of USP graphs, this approximation ratio was improved by Angelidakis et al. [9] to the logarithm of the graph hop-diameter D_H, i.e., the maximum number of hops of a shortest path in G, showing an approximation gap between USP and non-USP graphs.

2 Preliminaries

2.1 Definitions

We are given a weighted undirected graph $G = (V, E, \omega)$ where $\omega : E \to \mathbb{R}^+$ associates a weight with each edge of G. For a positive integer parameter h and a pair $u, v \in V$, the *h-limited distance* between u and v, denoted $d^h_G(u, v)$, is defined as the length of the shortest path from u to v that contains at most h edges (aka *hops*). The usual shortest path distance can be defined as $d_G(u, v) = d^{n-1}_G(u, v)$. For the sake of brevity, we often let uv denote the pair $\{u, v\}$ representing an edge from u to v.

▶ **Definition 1.** *An (exact) h-hopset for a weighted graph G is a set of edges H such that $d^h_{G\cup H}(u, v) = d_G(u, v)$ for all u, v in $V(G)$ where $G \cup H = (V, E \cup H, w')$ is the graph augmented with edges of the hopset with weights $w'(u, v) = d_G(u, v)$ for $uv \in H$ and $w'(u, v) = w(u, v)$ for $uv \in E \setminus H$. The parameter h is called the* hopbound *of the hopset. Edges from set H are called* shortcuts *in G.*

By convention, we will assume that all self-loops at nodes of V are included in H. Thus, $G \cup H$ is a graph whose h-th power in the $(\min, +)$ algebra on $n \times n$ matrices of edge weights corresponds to the transitive closure of the weight matrix of graph G.

Equivalently, a h-hopset can be defined as a set H of edges such that for any pair s, t, there exists a path P of at most h edges from s to t in $G \cup H$ and a shortest path Q from s to t in G such that all nodes of P belong to Q and appear in the same order. Note that a h-hopset is completely specified by its set H of edges as the associated weights are deduced from distances in the graph.

2.2 Using a Hopset as a Distance Oracle

Hopsets may be used to answer shortest-path queries in a graph $G = (V, E)$. In general, given a hopset H, the naïve way to approach a query for $d_G(u, v)$ for a given node pair u, v is to perform a bidirectional Dijkstra search in graph $G \cup H$ from this node pair, limited to a maximum of $\lceil h/2 \rceil$ hops distance from each of these nodes. We have, in particular for any pair $u, v \in V$:

$$d_G(u, v) = \min_{w \in V} (d^{\lceil h/2 \rceil}_{G\cup H}(u, w) + d^{\lfloor h/2 \rfloor}_{G\cup H}(v, w)).$$

Different optimizations of this technique are possible.

In this paper, we focus only on the time complexity of the case of $h = 3$, where we perform the following optimization of query execution. We represent set H as the union of two (not necessarily disjoint) sets of shortcuts, $H = H_1 \cup H_2$, where an edge belongs to H_1 if it is used as the first or third (last) hop on a shortest path in $G \cup H$, and it belongs to $G \cup H_2$ if it is used as the second hop on such a path. By convention, we assume that self-loops at nodes are added to H_1, thus e.g. a 3-hop path between a pair of adjacent nodes in G is constructed by taking a self-loop from H_1, the correct edge from $G \subseteq G \cup H_2$, and another self-loop from H_1. (Note that we never directly use edges of G as first or last hops in the hopset; if such an edge is required for correctness of construction, it should be explicitly added to set H_1.) We further apply an orientation to the shortcuts in H_1, constructing a corresponding set of arcs $\vec{H}_1$, such that, for any node pair $u, v \in V$, there exist $x, y \in V$ such that $(u, x) \in \vec{H}_1$, $\{x, y\} \in H_2$, $(v, y) \in \vec{H}_1$, and:

$$d_G(u, v) = d_G(u, x) + d_G(x, y) + d_G(y, v).$$

The orientation (w, z) of an arc in $\vec{H}_1$ indicates that edge $\{w, z\}$ can be used as the first edge of a 3-hop path from w or as the third edge of a 3-hop path to w. We note that $|H_1| \leq |\vec{H}_1| \leq 2|H_1|$, since each shortcut from H_1 corresponds to at most a pair of symmetric

arcs in $\vec{H}_1$. For a node $w \in V$, let $N_1(w) = |\{x \in V : (w,x) \in \vec{H}_1\}|$ represent the out-neighborhood of w in the graph $(V, \vec{H}_1)$. To perform shortest path queries on G, for each node w, we now store the list $\{(x, d_G(w,x)) : x \in N_1(w)\}$. We also store a hash map, mapping all node pairs $\{x,y\} \in H_2$ to the length of the respective link, $d_G(x,y)$. Now, we answer the distance query for a node pair $u, v \in G$ as follows:

$$d_G(u,v) = \min_{x \in N_1(u), y \in N_1(v) : \{x,y\} \in H_2} (d_G(u,x) + d_G(x,y) + d_G(y,v)).$$

Using the given data structures, the query is then processed using $|N_1(u)| \cdot |N_1(v)|$ hashmap look-ups, one for each pair $(x,y) \in N_1(u) \times N_1(v)$, i.e., in time $\mathcal{T}_{uv} = O(|N_1(u)| \cdot |N_1(v)|)$. Time $\mathcal{T}_{uv}$ is simply referred to as the *query time* for the considered node pair in the 3-hopset oracle H. Assuming uniform query density over all node pairs, the *uniform-average query time* $\mathcal{T}(H)$ is given as: $\mathcal{T}(H) \equiv \mathbf{E}_{uv}\mathcal{T}_{uv} = O\left(\frac{1}{n^2}\left(\sum_{u \in V} |N_1(u)|\right)^2\right) = O(|H_1|^2/n^2)$. Thus, in the uniform density setting (which we refer to only in Section 4), the average time of processing a query is proportional to the square of the average degree of a node with respect to edge set H_1.

The size of set H_2 affects only the size of the data structure required by the distance oracle, which is given as at most $\mathcal{S} = O(|E| + |H_1| + |H_2|)$ edges, with each edge represented using $O(\log n)$ bits.

In the 3-hopset distance oracles described in the following sections, we will confine ourselves to describing shortcut sets H_1 and H_2, noting that the correct orientation $\vec{H}_1$ of H_1 will follow naturally from the details of the provided constructions.

3 Bounded Skeleton Dimension

A formal definition of the notion of skeleton dimension relies on the concept of the geometric realization of a graph, cf. [37]. The *geometric realization* $\widetilde{G}$ of G can be seen as the "continuous" graph where each edge is seen as infinitely many vertices of degree two with infinitely small edges, such that for any $uv \in E(G)$ and $t \in [0,1]$, there is a node in $\widetilde{G}$ at distance $td_G(u,v)$ from u on edge uv. Given a shortest-path tree T_u of node u with length function $\ell : E(T_u) \to \mathbb{R}^+$, obtained as the union of shortest paths $\bigcup\{P_{uv} : v \in V(G)\}$, we treat it as directed from root to leaves and consider the geometric realization $\widetilde{T}_u$ of this directed graph. We define the *reach* of $v \in V(\widetilde{T}_u)$ as the distance from v to the furthest leaf in its subtree of the directed tree $\widetilde{T}_u$, i.e., $Reach_{\widetilde{T}_u}(v) := \max_{x : v \in P_{ux}} d_{\widetilde{T}_u}(v,x)$. For a given value $\alpha > 0$, we then define the *skeleton* T_u^* of T_u as the subtree of $\widetilde{T}_u$ induced by nodes with reach at least α times their distance from the root. More precisely, $\widetilde{T}_u^*$ is the subtree of $\widetilde{T}_u$ induced by $\{v \in V(\widetilde{T}_u) \mid Reach_{\widetilde{T}_u}(v) \geq \alpha d_{\widetilde{T}_u}(u,v)\}$.

The *α-skeleton dimension* k_α of a graph G is now defined as the maximum width of the skeleton of a shortest path tree, taken over cuts at all possible distances from the root of the tree: $k = \max_{u \in V(G)} \max_{r > 0} |Cut_r(\widetilde{T}_u^*)|$, where $Cut_r(\widetilde{T}_u^*)$ is the set of nodes $v \in V(\widetilde{T}_u^*)$ with $d_{\widetilde{T}_u^*}(u,v) = r$. When $\alpha = \frac{1}{2}$, $k_{1/2}$ is simply called the *skeleton dimension* of G and we let $k = k_{1/2}$ denote it.

The definition was originally proposed with $\alpha = \frac{1}{2}$ (for comparison with highway dimension) in the context of USP graphs [37]. In the long version [36], the definition is extended to other choices of α with $0 < \alpha < 1$ and applies to any choice of shortest paths trees that pairwise agree on their paths (the path from u to v in T_u must be the reverse of the path from v to u in T_v). In the non-USP case, the skeleton dimension should be measured with the best choice of agreeing trees. In particular, if a small perturbation of the edge weights of G provides unique shortest path trees whose skeletons have width at most k_α, then the skeleton dimension of G is at most k_α. The α-skeleton dimension (with parameter α) was

introduced in [36] for the sake of a general definition with fixed α value in mind. We use it here in a novel manner with α tending towards 0 as we consider larger distances, enabling analysis of our new construction.

For the definition of the related concept of *highway dimension*, we refer the readers to [4]. We note that if a graph G has highway dimension h, then G has skeleton dimension $k = k_{1/2} \leq h$; hence, in all subsequent asymptotic analyses, upper bounds expressed in terms of skeleton dimension can be replaced by analogous bounds in terms of highway dimension.

3.1 Construction of the 3-Hopset

We denote by $L_{\max}$ the maximum length of an edge in graph G. The construction of the 3-hopset H is obtained by taking a union of sets of shortcuts, each of which covers sets of node pairs within a given distance range. The first shortcut set H' covers all node pairs $u, v \in V$ with $d_G(u,v) \leq D'$, for some choice of distance bound D', whereas each of the subsequent shortcut sets $H^{(D)}$ covers nodes at a distance in an exponentially increasing distance range, $d_G(u,v) \in [D, D^{1+\epsilon}]$, where $\epsilon := \frac{1}{2\log_2 k}$ is suitably chosen. We then put:

$$H = H' \cup \bigcup_{i=1,2,\ldots} H^{(D'^{\,i(1+\epsilon)})}.$$

Construction of set H'. We note that a construction of 2-hopsets for graphs of skeleton dimension k was performed in [37]. As a direct corollary of [37][Lem. 2, Cor. 1,2], given a distance bound D', there exists a randomized polynomial-time construction of a set of shortcuts H' for graph G with the property that for any pair of nodes $u, v \in V$ with $d_G(u,v) \leq D'$, we have $d^2_{G \cup H'} = d_G(u,v)$, such that $|H'| = O(nk \log D')$, and moreover for all $u \in V$, we have $\mathbf{E}\deg_{H'}(u) = O(k \log D')$ and $\deg_{H'}(u) = O(k \log D' \log\log n + \log n)$. We directly use set H' for the value $D' := L^4_{\max} k^6 \log^{12} n$, considering H' as a 3-hopset for node pairs $u, v \in V$ with $d_G(u,v) \leq D'$. So we have:

$$|H'| = O(nk(\log\log n + \log L_{\max} + \log k)),$$

and for all $u \in V$:

$$\begin{aligned} \mathbf{E}\deg_{H'}(u) &= O(k(\log\log n + \log L_{\max} + \log k)), \\ \deg_{H'}(u) &= O(k \log\log n(\log\log n + \log L_{\max} + \log k) + \log n). \end{aligned}$$

We remark that, without loss of generality, in asymptotic analysis one may assume that $L_{\max} \leq kL$, where L is the *average* edge length in G, noting that edges longer than kL can be subdivided into edges of length at most kL by inserting additional vertices, increasing the number of nodes of the graph only by a multiplicative constant. Thus, in the above bounds, we can replace $(\log\log n + \log L_{\max} + \log k)$ by $(\log\log n + \log L + \log k)$.

Construction of set $H^{(D)}$. We now proceed to construct a 3-hopset for node pairs u, v with $d_G(u,v) \in [D, D^{1+\epsilon}]$. The construction of set $H^{(D)}$ is randomized and completely determined by assignment of real values $\rho(u) \in [0,1]$ to each node $u \in V$, uniformly and independently at random. We condition all subsequent considerations on the event that all values ρ are distinct, i.e., $|\rho(V)| = |V|$, which holds with probability 1. ($\rho(V) = \{\rho(v) | v \in V\}$)

Now, hopset $H^{(D)}$ is defined as $H^{(D)} := H^{(D)}_1 \cup H^{(D)}_2$, where following our usual notation, $H^{(D)}_1$ is the set of first and last hops, and $H^{(D)}_2$ is the set of middle hops.

Set of first and last hops. For $u \in V$, let $R^{(D)}(u)$ be the set of nodes which lie on a shortest path of length at least D which has one of its endpoints at u, and which have minimum value of ρ among all vertices on this path at distance in $[D/4, D/2]$ from u:

$$R^{(D)}(u) = \bigcup_{v \in V : d_G(u,v) \geq D} \left\{ \text{argmin}_{r \in P_{uv}, d_G(u,r) \in [D/4, D/2]} \rho(r) \right\}.$$

We now put: $H_1^{(D)} := \{ur : u \in V, r \in R(u)\}$.

Set of middle hops. We put in $H_2^{(D)}$ links between all pairs of nodes which have a small value of ρ, satisfy the natural upper bound of $D^{1+\epsilon}$ on distance between them, and have sufficiently large reach, i.e., the shortest path between them can be extended by at least $D/4$:

$$H_2^{(D)} := \left\{ qr : q, r \in \bigcup_{u \in V} R^{(D)}(u) \wedge d_G(q,r) \leq D^{1+\epsilon} - D/2 \wedge (\exists_{v \in V}\ r \in P_{qv} \wedge d_G(r,v) \geq D/4) \right\}.$$

The validity of H as a 3-hopset is immediate to verify from the construction: consider u, v and $i \geq 0$ such that $d_G(u,v) \in [D, D^{1+\epsilon}]$ with $D = D'^{i(1+\epsilon)}$.

For $q = \text{argmin}_{w \in P_{uv}, d_G(u,w) \in [D/4, D/2]} \rho(w)$ and $r = \text{argmin}_{w \in P_{uv}, d_G(u,w) \in [D/2, 3D/4]} \rho(w)$, we then have $uq \in H_1^{(D)}$, $qr \in H_2^{(D)}$ and $vr \in H_1^{(D)}$, yielding a 3-hop shortest path from u to v. For $d_G(u,v) \leq D'$, H' contains a 2-hop shortest path from u to v.

3.2 Bound on 3-Hopset Size and Oracle Time

▶ **Lemma 2.** *Fix $u \in V$ and $D > 0$. We have: $|R^{(D)}(u)| \leq k$.*

From the above Lemma, it follows that for any $u \in V$, we have $\deg_{H_1^{(D)}}(u) \leq k$. Thus summing over all the $O(\log\log(nL_{\max})/\log(1+\epsilon)) = O(\log\log(nL_{\max})\log k)$ levels of the construction, we successively obtain:

$$\deg_{H_1}(u) \leq \deg_{H'}(u) + k \cdot O(\log\log(nL_{\max})\log k) = O(k \log\log n \log k(\log\log n + \log L) + \log n), \quad (1)$$

$$\mathbf{E}\deg_{H_1}(u) \leq \mathbf{E}\deg_{H'}(u) + k \cdot O(\log\log(nL_{\max})\log k) = O(k \log k(\log\log n + \log L)), \quad (2)$$

$$|H_1| \leq |H'| + nk \cdot O(\log\log(nL_{\max})\log k) = O(nk \log k(\log\log n + \log L)). \quad (3)$$

We now proceed to bound the size of the set H_2 of middle hopsets.

▶ **Lemma 3.** *Fix $D \geq D'$. With probability $1 - O(1/n^2)$, it holds that for all $u \in V$ and for all $r \in R^{(D)}(u)$, we have $\rho(r) \leq L_{\max}/D$.*

We now proceed under the assumption that the event from the claim of the Lemma holds. We now consider an arbitrary node $q \in R^{(D)}(u)$ for some $u \in V$, and look at $\deg_{H_2^{(D)}}(q)$. We now have that if $qr \in H_2^{(D)}$, then by the definition of $H_2^{(D)}$ and the above Lemma, the following conditions jointly hold:

- $\rho(r) \leq L_{\max}/D$
- $r \in \{w \in V : \exists_{v \in V}\ D^{1+\epsilon} \geq d_G(q,v) \geq d_G(q,w) + D/4 \wedge P_{qw} \subseteq P_{gv}\} =: W(q)$.

We note that $W(q)$ is the subset of the vertex set of the shortest path tree of node q, pruned to contain only those paths which have reach at least $D/4$ at depth less than $D^{1+\epsilon}$. This tree has depth bounded by $D^{1+\epsilon}$, and width bounded by an α-skeleton dimension k_α (following [36]), with parameter $\alpha = \frac{D/4}{D^{1+\epsilon}} = D^{-\epsilon}/4$. Following [36][Section 6], k_α can be easily expressed using skeleton dimension $k = k_{1/2}$ as:

$$k_\alpha \leq k^{\lceil \log_2(1+1/\alpha) \rceil} < k^{1+\log_2(4D^\epsilon)} = k^3 D^{\epsilon \log_2 k}.$$

We then have $|W(q)| \leq D^{1+\epsilon}k_\alpha < k^3 D^{1+\epsilon(1+\log_2 k)}$. Moreover, by an easy concentration bound, we have that for all $q \in V$, $|\{r \in W(q) : \rho(r) \leq L_{\max}/D\}| = O(\log n) + \frac{2L_{\max}}{D}|W(q)|$, with probability $1 - O(1/n^2)$. It follows that with probability $1 - O(1/n^2)$, we have for all $q \in \bigcup_{u\in V} R^{(D)}(u)$:

$$\deg_{H_2^{(D)}}(q) \leq O(\log n) + \frac{2L_{\max}}{D}|W(q)| \leq O(\log n + L_{\max}k^3 D^{\epsilon \log_2 k}).$$

Noting that with probability $1 - O(1/n^2)$:

$$|\bigcup_{u\in V} R^{(D)}(u)| \leq |\{w \in V : \rho(w) \leq L_{\max}/D| \leq O(\log n + nL_{\max}/D)$$

we finally obtain that with probability $1 - O(1/n^2)$:

$$\begin{aligned}|H_2^{(D)}| &\leq O(\log n + nL_{\max}/D)O(\log n + L_{\max}k^3 D^{\epsilon \log_2 k}) = O(\log^2 n + nL_{\max}^2 k^3 D^{\epsilon \log_2 k - 1}) \\ &\leq O(nL_{\max}^2 k^3 D^{-1/2}) \leq O(nD'^{-1/4}) \leq O(n/\log^3 n),\end{aligned}$$

where in the last two transformations we use the fact that $\epsilon = \frac{1}{2\log_2 k}$ and that $D \geq D' \geq L_{\max}^4 k^6 \log^{12} n$. Using a union bound and summing over all levels of the construction, we eventually obtain that with probability $1 - O(1/n)$:

$$|H_2| \leq O(n/\log_2 n). \tag{4}$$

Thus, the set of middle links is sparse and does not contribute to the asymptotic size of the overall representation of the 3-hopset.

Overall, considering a randomized construction which rejects random choices of ρ for which any of the considered w.h.p. events fail, by combining Eq. (1)–(4) with the hopset-based distance oracle framework described in the Preliminaries, we obtain the following Theorem.

▶ **Theorem 4.** *For a unique shortest path graph with skeleton dimension k and average link length $L \geq 1$, there exists a randomized construction of a 3-hopset distance oracle of size $|H| = O(nk \log k(\log\log n + \log L))$, which for an arbitrary queried node pair performs distance queries in expected time $O(k^2 \log^2 k(\log^2 \log n + \log^2 L))$ (where the expectation is taken over the randomized construction of the oracle), and in time $O(k^2 \log^2 k \log^2 \log n(\log^2 \log n + \log^2 L) + \log^2 n)$ with certainty.*

In particular, for graphs with constant-length edges and small skeleton dimension ($k = O(\log n)$), the 3-hopset has size $|H| = O(nk \log k \log\log n)$, with expected time of any query given as $O(k^2 \log^2 k \log^2 \log n)$.

4 LP-based Approximation Algorithm

In this section, we propose an Integer Linear Programming (ILP) formulation for h-hopsets with a minimum number of edges, which we then relax to a LP formulation. Whereas both formulations are applicable to the general case, we prove relations between them only for USP graphs.

4.1 ILP and LP Formulations

A necessary and sufficient condition for H to be a h-hopset for G is that for every pair of vertices s, t there exists a path $P_{st} = (s = v_0, v_1, \ldots, v_{l_{st}} = t)$ in $G \cup H$ such that $l_{st} \leq h$ and in graph G there exists some shortest $s-t$ path passing through all of the vertices $v_0, \ldots, v_{l_{st}}$, in the given order. For a fixed pair s, t, we consider the directed graph H^{st} with vertex set

$V \times \{0, \ldots, h\} \equiv V_h$ (by convention, elements of V_h will be denoted compactly as v_i, where $v \in V$, $i \in \{0, \ldots, h\}$) and with an arc set defined as follows. For $i \in \{0, \ldots, h-1\}$, we add arc (u_i, v_{i+1}) to H^{st} if and only if $\{u, v\} \in G \cup H$ and u, v lie on some shortest $s-t$ path in the given order, i.e., if $d_G(s,u) + d_G(u,v) + d_G(v,t) = d_G(s,t)$. In particular, all arcs of the form (u_i, u_{i+1}), for $u \in V$ on a $s-t$ shortest path, belong to H^{st}. Now, we have that H is a h-hopset for G if and only if there exists a path from s_0 to t_h in H^{st}. This is equivalent to saying that for all $s, t \in V$, the flow value from s_0 to t_h is at least 1 in H^{st}. Given graph G, we thus have the following ILP formulation for the minimum h-hopset problem, using indicator variables x_{uv} for $G \cup H$ (given as 1 if $\{u, v\} \in G \cup H$ and 0 otherwise) and variables $f^{st}_{u_i v_j}$, representing the flow value along arc (u_i, v_j) in H^{st}:

$$\text{Minimize:} \quad \sum_{u \neq v, \{u,v\} \notin E} x_{uv} \tag{5}$$

Subject to:

$$x_{uv} \in \{0, 1\} \tag{6}$$

$$0 \leq f^{st}_{u_i v_j} \leq \begin{cases} x_{uv}, & \text{if } j = i+1 \text{ and } d_G(s,u) + d_G(u,v) + d_G(v,t) = d_G(s,t), \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

$$\sum_{u_i} f^{st}_{v_j u_i} - \sum_{u_i} f^{st}_{u_i v_j} = \begin{cases} 0, & \text{for } v_j \in V_h \setminus \{s_0, t_h\} \\ +1, & \text{for } v_j = s_0 \\ -1, & \text{for } v_j = t_h \end{cases}, \tag{8}$$

where indices s, t, u, v traverse V and indices i, j traverse $\{0, \ldots, h\}$.

To obtain an LP relaxation of the above problem, we replace the integral condition $x_{uv} \in \{0, 1\}$ by the fractional one $x_{uv} \in [0, 1]$. We look at the connection between the integral and fractional forms for the special case of unique shortest path graphs.

We remark that the above formulation can be seen as a generalization of the LP and ILP statement of Angelidakis et al. [9] proposed for the special case of 2-hop labeling. In the case of 2-hop labeling, Angelidakis et al. do not rely on an explicit flow formulation but use a single constraint of the simpler form $\sum_{w \in P^{st}} \min\{x_{sw}, x_{wt}\} \geq 1$, where P^{st} represents the set of nodes on some shortest $s-t$ path in G. However, the analysis of the integrality gap does not carry over from the case of $h = 2$ to $h > 2$, i.e., as soon as there exist internal shortcuts which have neither s nor t as one of their endpoints.

4.2 Bounding Integrality Gap for Unique Shortest Path Graphs

We analyze the integrality gap of the above LP formulation for the case of *unique shortest path (USP) graphs*, i.e., graphs in which each pair of nodes $s, t \in V$ is connected by a unique shortest path P^{st} in G. We will occasionally identify P^{st} with its set of nodes, and we will introduce a linear order on its vertices, writing for $u, v \in P^{st}$ that $u <^{st} v$ if $d_G(s,u) < d_G(s,v)$; we will denote the order simply as "$<$" when the path P^{st} is clear from the context. Observe that in the LP formulation, we may have $f^{st}_{u_i v_j} \neq 0$ only if $u <^{st} v$ and $j = i+1$. Thus, fixing $s, t \in V$, the flow $f^{st} = (f^{st}_{u_i v_j} : u_i, v_j \in V_h)$ is non-zero between vertices of $\{P^{st}\} \times \{0, 1, \ldots, h\}$ only, and the flow is oriented towards t on this path.

Let $(x_{uv}, f^{st}_{u_i v_j})$ be a fixed solution to the LP problem in a USP graph, with cost $COST_{\text{LP}} = \sum_{u \neq v, \{u,v\} \notin E} x_{uv}$. We will show how to use this set to construct a valid hopset H'' for G (thus, equivalently, also solving the ILP formulation). We first apply a randomized rounding procedure following the classical scheme of Raghavan and Thomson [39]. We define the family

of independent random variables $(x'_{u_i v_{i+1}} : u, v \in V, i \in \{0, \ldots, h\})$, with $x'_{u_i v_{i+1}} \in \{0, 1\}$. For $u \neq v, \{u, v\} \notin E$ we put $\Pr[x'_{u_i v_{i+1}} = 1] = \min\{Cx_{uv}, 1\}$, where $C \geq 1$ is a suitably chosen probability amplification parameter (we put $C = 8h \ln n$). We will assume, without affecting the validity or cost of the solution, that $x_{uv} = x'_{u_i v_{i+1}} = 1$, when $u = v$ or $\{u, v\} \in E$.

We denote $H' = \{\{u, v\} : u, v \in V \wedge u \neq v \wedge \{u, v\} \notin E \wedge \exists_{i \in \{0,\ldots,h-1\}}\ x'_{u_i v_{i+1}} = 1\}$. Let $\pi : V \to \{1, \ldots, n\}$ be a bijection picked uniformly at random (it is a random permutation when $V = \{1, \ldots, n\}$). We define the set of shortcuts $S(\{u, v\})$ associated with each pair $\{u, v\} \in H'$ as the set of all pairs of nodes on path P^{uv}, one of which is a prefix minimum on this path with respect to π, and the other of which is a suffix minimum with respect to π:

$$S(\{u, v\}) := \left\{ \{u^*, v^*\} : u^*, v^* \in P^{uv} \wedge \pi(u^*) = \min_{z \in P^{uv}, z \leq_{uv} u^*} \pi(z) \wedge \pi(v^*) = \min_{z \in P^{uv}, z \geq_{uv} v^*} \pi(z) \right\}.$$

The obtained solution is given as the set of all such shortcuts: $H'' := \bigcup_{\{u,v\} \in H'} S(\{u, v\})$.

▶ **Proposition 5.** *With probability $1 - O(1/n)$, set H'' is a hopset for G of size $O(h^2 \log^3 n \cdot COST_{\text{LP}})$.*

We remark that the above Proposition implies that the h-hopset problem can be efficiently approximated by finding an optimal fractional LP solution and constructing set H''.

▶ **Theorem 6.** *There exists a randomized polynomial-time $O(\text{poly} \log n)$-approximation algorithm for the h-hopset problem in unique shortest path graphs, for any $h \leq O(\text{poly} \log n)$.*

4.3 Approximating Average Query Time for 3-Hopsets

In order to design an efficient distance oracle based on 3-hopsets, we follow the framework described in the preliminaries and use an LP-rounding technique to obtain sets $H_1 \cup H_2 =: H$. The obtained claim relies on the notion of uniform-average query time introduced in the Preliminaries.

▶ **Theorem 7.** *For any feasible bound $\mathcal{S}$, let $H_{OPT,\mathcal{S}}$ be a 3-hopset for a unique shortest path graph, which satisfies the given bound on the number of edges $|H_{OPT,\mathcal{S}}| \leq \mathcal{S}$ and such that the uniform-average query time $\mathcal{T}(H_{OPT,\mathcal{S}})$ is minimized. Then, there exists a randomized polynomial-time algorithm which finds a 3-hopset H with $|H''| \leq O(\log^3 n)\mathcal{S}$ and $\mathcal{T}(H'') \leq O(\log^4 n)\mathcal{T}(H_{OPT,\mathcal{S}})$.*

We remark that the above Theorem can be directly generalized to a notion of average query time for non-uniform query densities, in which the goal is to minimize expected query time in a model in which each node $v \in V$ is assigned its relative frequency $f_v \in [0, 1]$, and a node pair uv is queried with frequency $f_u f_v$.

5 Bounded Treewidth Graphs

We now show how to obtain h-hopsets for graphs with bounded treewidth by following a classical construction for trees. We first begin with preliminaries recalling the definitions of treewidth and inverse Ackermann function.

Treewidth definition. Recall that a graph G has treewidth t if there exists a tree T whose nodes are subsets of $V(G)$ called *bags* such that: $|X| \leq t + 1$ for all $X \in V(T)$; for all edges $uv \in E(G)$, there exists a bag $X \in V(T)$ containing both u and v $(u, v \in X)$; and for all nodes $u \in V(G)$, the bags containing u form a sub-tree of T. Without loss of generality, we assume that each bag contains exactly $t + 1$ nodes, and that two neighboring bags share exactly t nodes (the decomposition is standard). This implies $|V(T)| \leq n$ as each bag brings

one new node. Note that removing a non-leaf bag separates the graph into several connected components. We consider that all edges of T have weight 1. For convenience, we assume that T is rooted at some bag R and define for each node $u \in V(G)$ the root bag of u as the bag $R_u \in V(T)$ containing u which is closest to the root.

Inverse Ackermann notation. The *kth-row inverse Ackermann* function $\lambda_k(.)$ can be defined by $\lambda_0(n) = \frac{n}{2}$, $\lambda_1(n) = \sqrt{n}$, $\lambda_2(n) = \log n$, $\lambda_3(n) = \log\log n$, $\lambda_4(n) = \log^* n$, and more generally for $k \geq 2$ by the recurrence $\lambda_k(n) = \lambda^*_{k-2}(n)$ where we define for any function f: $f^{(0)}(n) = n$, $f^{(i)}(n) = f(f^{(i-1)}(n))$ for $i > 0$, and $f^*(n) = \min\{j \mid f^{(j)}(n) \leq 1\}$. The *inverse Ackermann* function can be defined as $\alpha(n) = \min\{j \mid \lambda_{2j}(n) \leq j\}$. See [33] for a more formal definition based on Ackermann function.

We first consider the case of (weighted) trees for which the construction of h-hopsets is classical (even though the connection with hopsets was not made). It is implicit in [7, 16], explicit for unweighted trees in [13] and directed trees in [42]. We provide a short construction which fine-grains the dependence of the hopset size on h (e.g., replacing $2h$ by h with respect to the asymptotic analysis in [7]). The construction is based on the following folklore lemma for splitting a tree into smaller sub-trees (it can be seen as a generalization of the existence of a centroid).

▶ **Lemma 8.** *Given a rooted tree T with n nodes and a value $p > 1$, there exists a set P of at most $2p$ nodes such that each connected component of $T \setminus P$ contains less than n/p nodes and is connected to at most two nodes in P. Set P can be computed in linear time through a bottom-up traversal of the tree.*

***h*-hopset construction for trees.** A 1-hopset in a tree T is obtained by adding all pairs as edges with appropriate weight. For $h > 1$, we recursively define a h-hopset of T as follows. Select a set P of $2p$ nodes at most with $p = \frac{n}{\lambda_{h-2}(n)}$ according to Lemma 8. When $h = 2$, we add an edge from each node u of T to each node in P. When $h > 2$, we consider the forest T' induced by nodes in P: it has node set P and edges xy such that y is the closest ancestor of x in T that belongs to P. The weight of such an edge is defined as $w'(x, y) = d_T(x, y)$. We then add a $(h-2)$-hopset of T' to the construction. Additionally, we add one or two edges per node not in P: for each connected component C of $T \setminus P$, add an edge ux for each node $u \in C$ and each $x \in P$ connected to C. Note that Lemma 8 ensures that there are at most two such nodes x for a given component C. In both cases ($h \geq 2$), we construct recursively a h-hopset of each sub-tree induced by a connected component C of $T \setminus P$. In the special case of $h = 3$, the $(h-2)$-hopsets contribute to H_2 while all edges connecting to a node in some selected set P contribute to H_1 according to the $H = H_1 \cup H_2$ convention introduced in the Preliminaries.

Following a similar approach on the tree decomposition of a graph with treewidth t, we obtain the following result (detailed construction is given in the full paper [33]).

▶ **Theorem 9.** *For all $h > 1$, any graph with treewidth t has a h-hopset with $O(tn\lambda_h(n))$ edges and a $2(\alpha(n)+1)$-hopset with $O(t^2 n)$ edges.*

For the special case of $h = 3$, we have $\lambda_3(n) = \log\log n$, and the size required to represent the 3-hop data structure is $\mathcal{S} = O(tn \log\log n)$ edges. Following the convention $H = H_1 \cup H_2$, we note that we have $\deg_{H_1}(v) = O(t \log\log n)$ for any $v \in V$. The following bound on the query time follows.

▶ **Theorem 10.** *Any graph with treewidth t admits a 3-hopset distance oracle represented on $O(tn \log\log n)$ edges of $O(\log n)$ bits, with a query time of $O(t^2 \log^2 \log n)$.*

References

1 Amir Abboud, Greg Bodwin, and Seth Pettie. A Hierarchy of Lower Bounds for Sublinear Additive Spanners. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 568–576. SIAM, 2017. doi:10.1137/1.9781611974782.36.

2 Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. VC-dimension and shortest path algorithms. In *ICALP*, volume 6755 of *Lecture Notes in Computer Science*, pages 690–699. Springer, 2011.

3 Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. Highway Dimension and Provably Efficient Shortest Path Algorithms. *J. ACM*, 63(5):41:1–41:26, 2016. doi:10.1145/2985473.

4 Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. Highway Dimension and Provably Efficient Shortest Path Algorithms. *J. ACM*, 63(5):41:1–41:26, 2016. doi:10.1145/2985473.

5 Ittai Abraham, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. Highway Dimension, Shortest Paths, and Provably Efficient Algorithms. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 782–793. SIAM, 2010. doi:10.1137/1.9781611973075.64.

6 Takuya Akiba, Yoichi Iwata, and Yuichi Yoshida. Dynamic and historical shortest-path distance queries on large evolving networks by pruned landmark labeling. In Chin-Wan Chung, Andrei Z. Broder, Kyuseok Shim, and Torsten Suel, editors, *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014*, pages 237–248. ACM, 2014. doi:10.1145/2566486.2568007.

7 Noga Alon and Baruch Schieber. Optimal preprocessing for answering on-line product queries. Technical Report 71/87, Tel Aviv University, 1987.

8 Stephen Alstrup, Søren Dahlgaard, Mathias Bæk Tejs Knudsen, and Ely Porat. Sublinear Distance Labeling. In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, volume 57 of *LIPIcs*, pages 5:1–5:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.ESA.2016.5.

9 Haris Angelidakis, Yury Makarychev, and Vsevolod Oparin. Algorithmic and Hardness Results for the Hub Labeling Problem. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1442–1461. SIAM, 2017. doi:10.1137/1.9781611974782.94.

10 Srinivasa Rao Arikati, Danny Z. Chen, L. Paul Chew, Gautam Das, Michiel H. M. Smid, and Christos D. Zaroliagis. Planar Spanners and Approximate Shortest Path Queries among Obstacles in the Plane. In Josep Díaz and Maria J. Serna, editors, *Algorithms - ESA '96, Fourth Annual European Symposium, Barcelona, Spain, September 25-27, 1996, Proceedings*, volume 1136 of *Lecture Notes in Computer Science*, pages 514–528. Springer, 1996. doi:10.1007/3-540-61680-2_79.

11 H. Bast, Stefan Funke, and Domagoj Matijevic. Ultrafast Shortest-Path Queries via Transit Nodes. In Camil Demetrescu, Andrew V. Goldberg, and David S. Johnson, editors, *The Shortest Path Problem, Proceedings of a DIMACS Workshop, Piscataway, New Jersey, USA, November 13-14, 2006*, volume 74 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 175–192. DIMACS/AMS, 2006.

12 Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P. Woodruff. Transitive-Closure Spanners. *SIAM J. Comput.*, 41(6):1380–1425, 2012. doi:10.1137/110826655.

13 Hans L. Bodlaender, Gerard Tel, and Nicola Santoro. Trade-Offs in Non-Reversing Diameter. *Nord. J. Comput.*, 1(1):111–134, 1994.

14 Sergio Cabello. Many Distances in Planar Graphs. *Algorithmica*, 62(1-2):361–381, 2012. doi:10.1007/s00453-010-9459-0.

15 Shiva Chaudhuri and Christos D. Zaroliagis. Shortest Paths in Digraphs of Small Treewidth. Part I: Sequential Algorithms. *Algorithmica*, 27(3):212–226, 2000. doi:10.1007/s004530010016.

16 Bernard Chazelle. Computing on a Free Tree via Complexity-Preserving Mappings. *Algorithmica*, 2:337–361, 1987. doi:10.1007/BF01840366.

17 Danny Z. Chen and Jinhui Xu. Shortest path queries in planar graphs. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 469–478, 2000. doi:10.1145/335305.335359.

18 Edith Cohen. Using Selective Path-Doubling for Parallel Shortest-Path Computations. *J. Algorithms*, 22(1):30–56, 1997. doi:10.1006/jagm.1996.0813.

19 Edith Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest paths. *J. ACM*, 47(1):132–166, 2000. doi:10.1145/331605.331610.

20 Edith Cohen, Eran Halperin, Haim Kaplan, and Uri Zwick. Reachability and Distance Queries via 2-Hop Labels. *SIAM J. Comput.*, 32(5):1338–1355, May 2003. doi:10.1137/S0097539702403098.

21 Vincent Cohen-Addad, Søren Dahlgaard, and Christian Wulff-Nilsen. Fast and Compact Exact Distance Oracle for Planar Graphs. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 962–973, 2017. doi:10.1109/FOCS.2017.93.

22 Daniel Delling, Andrew V. Goldberg, Thomas Pajor, and Renato F. Werneck. Robust Distance Queries on Massive Networks. In Andreas S. Schulz and Dorothea Wagner, editors, *Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, volume 8737 of *Lecture Notes in Computer Science*, pages 321–333. Springer, 2014. doi:10.1007/978-3-662-44777-2_27.

23 Hristo Djidjev. On-Line Algorithms for Shortest Path Problems on Planar Digraphs. In Fabrizio d'Amore, Paolo Giulio Franciosa, and Alberto Marchetti-Spaccamela, editors, *Graph-Theoretic Concepts in Computer Science, 22nd International Workshop, WG '96, Cadenabbia (Como), Italy, June 12-14, 1996, Proceedings*, volume 1197 of *Lecture Notes in Computer Science*, pages 151–165. Springer, 1996. doi:10.1007/3-540-62559-3_14.

24 Michael Elkin and Ofer Neiman. Hopsets with Constant Hopbound, and Applications to Approximate Shortest Paths. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 128–137. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.22.

25 Michael Elkin and Ofer Neiman. Linear-Size Hopsets with Small Hopbound, and Distributed Routing with Low Memory. *CoRR*, abs/1704.08468, 2017. arXiv:1704.08468.

26 Jittat Fakcharoenphol and Satish Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *J. Comput. Syst. Sci.*, 72(5):868–889, 2006. doi:10.1016/j.jcss.2005.05.007.

27 Arash Farzan and Shahin Kamali. Compact Navigation and Distance Oracles for Graphs with Small Treewidth. *Algorithmica*, 69(1):92–116, 2014. doi:10.1007/s00453-012-9712-9.

28 Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance Labeling in Graphs. *J. Algorithms*, 53(1):85–112, October 2004. doi:10.1016/j.jalgor.2004.05.002.

29 Pawel Gawrychowski, Adrian Kosowski, and Przemyslaw Uznanski. Sublinear-Space Distance Labeling Using Hubs. In *Distributed Computing - 30th International Symposium, DISC 2016, Paris, France, September 27-29, 2016. Proceedings*, pages 230–242, 2016. doi:10.1007/978-3-662-53426-7_17.

30 Pawel Gawrychowski, Shay Mozes, Oren Weimann, and Christian Wulff-Nilsen. Better Tradeoffs for Exact Distance Oracles in Planar Graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 515–529, 2018. doi:10.1137/1.9781611975031.34.

31 Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. In Catherine C. McGeoch, editor, *Experimental Algorithms, 7th International Workshop, WEA 2008, Provincetown, MA, USA, May 30-June 1, 2008, Proceedings*, volume 5038 of *Lecture Notes in Computer Science*, pages 319–333. Springer, 2008. doi:10.1007/978-3-540-68552-4_24.

32 Andrew V. Goldberg, Haim Kaplan, and Renato F. Werneck. Reach for A*: Efficient Point-to-Point Shortest Path Algorithms. In *ALENEX*, pages 129–143. SIAM, 2006.

33 Siddharth Gupta, Adrian Kosowski, and Laurent Viennot. Exact Distance Oracles Using Hopsets. *CoRR*, abs/1803.06977, 2018. arXiv:1803.06977.

34 Ronald J. Gutman. Reach-Based Routing: A New Approach to Shortest Path Algorithms Optimized for Road Networks. In *ALENEX/ANALCO*, pages 100–111. SIAM, 2004.

35 Philip N. Klein and Sairam Subramanian. A Randomized Parallel Algorithm for Single-Source Shortest Paths. *J. Algorithms*, 25(2):205–220, 1997. doi:10.1006/jagm.1997.0888.

36 Adrian Kosowski and Laurent Viennot. Beyond Highway Dimension: Small Distance Labels Using Tree Skeletons. *CoRR*, abs/1609.00512, 2016. arXiv:1609.00512.

37 Adrian Kosowski and Laurent Viennot. Beyond Highway Dimension: Small Distance Labels Using Tree Skeletons. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1462–1478. SIAM, 2017. doi:10.1137/1.9781611974782.95.

38 David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989. doi:10.1002/jgt.3190130114.

39 Prabhakar Raghavan and Clark D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987. doi:10.1007/BF02579324.

40 Hanmao Shi and Thomas H. Spencer. Time-Work Tradeoffs of the Single-Source Shortest Paths Problem. *J. Algorithms*, 30(1):19–32, 1999. doi:10.1006/jagm.1998.0968.

41 Mikkel Thorup. Shortcutting Planar Digraphs. *Combinatorics, Probability & Computing*, 4:287–315, 1995. doi:10.1017/S0963548300001668.

42 Mikkel Thorup. Parallel Shortcutting of Rooted Trees. *J. Algorithms*, 23(1):139–159, 1997. doi:10.1006/jagm.1996.0829.

43 Jeffrey D. Ullman and Mihalis Yannakakis. High-Probability Parallel Transitive Closure Algorithms. In *SPAA*, pages 200–209, 1990. doi:10.1145/97444.97686.

Optimal Strategies for Patrolling Fences

Bernhard Haeupler
Carnegie Mellon University, Pittsburgh, PA, USA
haeupler@cs.cmu.edu

Fabian Kuhn
University of Freiburg, Germany
kuhn@cs.uni-freiburg.de

Anders Martinsson
ETH Zurich, Switzerland
anders.martinsson@inf.ethz.ch

Kalina Petrova
ETH Zurich, Switzerland
kpetrova@student.ethz.ch

Pascal Pfister
ETH Zurich, Switzerland
ppfister@inf.ethz.ch

Abstract

A classical multi-agent fence patrolling problem asks: What is the maximum length L of a line fence that k agents with maximum speeds $v_1, \dots, v_k$ can patrol if each point on the line needs to be visited at least once every unit of time. It is easy to see that $L = \alpha \sum_{i=1}^{k} v_i$ for some efficiency $\alpha \in [\frac{1}{2}, 1)$. After a series of works [3, 8, 9, 10] giving better and better efficiencies, it was conjectured by Kawamura and Soejima [10] that the best possible efficiency approaches $\frac{2}{3}$. No upper bounds on the efficiency below 1 were known.

We prove the first such upper bounds and tightly bound the optimal efficiency in terms of the minimum speed ratio $s = \frac{v_{\max}}{v_{\min}}$ and the number of agents k. Our bounds of $\alpha \leq \frac{1}{1+\frac{1}{s}}$ and $\alpha \leq 1 - \frac{1}{\sqrt{k}+1}$ imply that in order to achieve efficiency $1-\epsilon$, at least $k \geq \Omega(\epsilon^{-2})$ agents with a speed ratio of $s \geq \Omega(\epsilon^{-1})$ are necessary. Guided by our upper bounds, we construct a scheme whose efficiency approaches 1, disproving the conjecture stated above. Our scheme asymptotically matches our upper bounds in terms of the maximal speed difference and the number of agents used.

A variation of the fence patrolling problem considers a circular fence instead and asks for its circumference to be maximized. We consider the unidirectional case of this variation, where all agents are only allowed to move in one direction, say clockwise. At first, a strategy yielding $L = \max_{r \in [k]} r \cdot v_r$ where $v_1 \geq v_2 \geq \dots \geq v_k$ was conjectured to be optimal by Czyzowicz et al. [3] This was proven not to be the case by giving constructions for only specific numbers of agents with marginal improvements of L. We give a general construction that yields $L = \frac{1}{33 \log_e \log_2(k)} \sum_{i=1}^{k} v_i$ for any set of agents, which in particular for the case $1, 1/2, \dots, 1/k$ diverges as $k \to \infty$, thus resolving a conjecture by Kawamura and Soejima [10] affirmatively.

2012 ACM Subject Classification Networks → Network algorithms; Theory of computation → Distributed algorithms

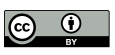
Keywords and phrases multi-agent systems, patrolling algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.144

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Related Version A full version of the paper is available at `https://arxiv.org/abs/1809.06727`.

Funding *Bernhard Haeupler*: Part of this work was done as a visiting professor at ETH Zürich supported by Mohsen Ghaffari. Supported also in part by NSF grants CCF-1527110, CCF-1618280, CCF-1814603, CCF-1910588, NSF CAREER award CCF-1750808 and a Sloan Research Fellowship. *Kalina Petrova*: Supported in part by Excellence Scholarship & Opportunity Programme.



46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 144; pp. 144:1–144:13

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

1 Introduction

Patrolling is a fundamental task in robotics, multi-agent systems, and security settings. Given some environment of interest, and a collection of mobile agents, the aim is to coordinate the movements of the agents in order to, for example, guard an area from intrusion by an enemy, prevent accidents or failure of equipment, maintain up-to-date information of the environment, etc. For each of these tasks, ensuring that certain points in the environment get visited/monitored frequently is crucial. Performance of patrolling algorithms is consequently often measured in terms of *idleness* – roughly speaking, the time between two consecutive visits to a point in the environment.

Multi-agent patrolling has been extensively studied in the robotics literature since the early 2000s, e.g., see [1, 11] and the survey [12]. However, even for extremely clean and very simple models, determining optimal patrolling schemes poses many natural mathematical questions with interesting and surprisingly sophisticated answers [2, 3, 4, 5, 6, 7, 8, 9, 10].

1.1 Fence Patrolling

This paper studies a classical fence patrolling problem introduced by Czyzowicz et al. [3], which might be one of the cleanest and most natural patrolling problems: What is the maximum length L of a fence that k agents $a_1, \dots, a_k$ with maximum speeds $v_1, \dots, v_k$ can patrol if each point needs to be visited at least once every unit of time. Czyzowicz et al. introduce two variations of this question – the fence could be either an open curve, or a closed curve. For simplicity, we assume the open curve is a line segment and the closed curve is a circle.

For the line segment, it is easy to see that for any speeds the maximum length L satisfies $L = \alpha \sum_{i=1}^{k} v_i$ for some efficiency $\alpha \in [\frac{1}{2}, 1)$. In particular, in one unit of time an agent a_i can cover a length of at most v_i and all agents can cover at most a total length of $\sum_{i=1}^{k} v_i$. An efficiency of exactly $\alpha = 1$ is furthermore never possible because agents have to turn around eventually. On the other hand, an efficiency of $\alpha = \frac{1}{2}$ can easily be achieved by the following strategy:

Partition-based strategy, $\mathcal{A}_1$: For all $i \in [k]$, agent a_i patrols a subsegment of length $\frac{1}{2}v_i$ by going back and forth on this segment once every unit of time. This patrols a segment of length $L = \frac{1}{2}\sum_{i=1}^{k} v_i$ with idle time 1.

Considering patrol schedules on a circle, the picture is quite different than for a line segment. Again, the length L of any circle that can be patrolled by a set of agents is upper-bounded by the sum of the maximum speeds of the agents, since agent a_i cannot cover a length of more than v_i. Here, however, it is easy to find collections of agents and a corresponding patrol schedule that achieves this exactly – imagine k identical agents starting equidistantly along the circle and moving in unison in the same direction, say counter-clockwise.

1.2 Prior Work on Fence Patrolling

1.2.1 Prior Work on the Line Segment

Czyzowicz et al. [3] observed that the trivial scheme $\mathcal{A}_1$ with efficiency $\frac{1}{2}$ is optimal if the paths of the agents never cross. To see this, note that the leftmost agent a_i cannot walk away further than $\frac{1}{2}v_i$ from the leftmost point of the fence as it would take more than one unit of time between two visits of this point. By the same argument the agent a_j to the

right of agent a_i cannot ever be further away than $\frac{1}{2}(v_i + v_j)$ from the leftmost point of the fence and induction shows that a total fence length of $\frac{1}{2}\sum_{i=1}^{k} v_i$ is best possible. For the special case of all agents having the same speed the assumption that the paths of the agents never cross is furthermore without loss of generality as one can equally well switch identities of agents at a crossing, making the agents bounce off each other instead of crossing. In the worst case an efficiency of $\frac{1}{2}$ is thus optimal and Czyzowicz et al. posited [3] that indeed no better efficiency can be achieved for any speeds.

Surprisingly, Kawamura and Kobayashi [9] disproved this by providing an explicit fence patrolling schedule for 6 agents with speeds $1, 1, 1, 1, \frac{7}{3}$, and $\frac{1}{2}$ for a fence of length $\frac{7}{2}$, thus achieving an efficiency of $\frac{21}{41} > \frac{1}{2}$. This was improved by Dumitrescu, Ghosh and Tóth [8], who proposed a family of patrolling schedules with efficiency approaching $\frac{25}{48}$, and finally by Kawamura and Soejima [10] who achieved an efficiency approaching $\frac{2}{3}$. Kawamura and Soejima furthermore explicitly conjectured that no efficiency better than $\frac{2}{3}$ is possible for any set of speeds [10, Conjecture 6, page 9].

On the other hand, except for the setting of equal speeds discussed above, no upper bounds on the efficiency below 1 have been provided in the literature [3, 9, 8, 10].

1.2.2 Prior work on the Circle

For a general set of agents, Czyzowicz et al. [3] proposed the following universal scheme that generalizes the schedule above for equal speeds:
Runners strategy, $\mathcal{A}_2$: Assume $v_1 \geq v_2 \geq \cdots \geq v_k$. Find the $r \in [k]$ that maximizes $r \cdot v_r$, and let the r fastest agents move equidistantly along the circle at speed v_r. This patrols a circle of length $L = \max_{r\in[k]} r \cdot v_r$ with idle time 1.

Suppose for a collection of agents with maximum speeds $v_1 \geq v_2 \geq \cdots \geq v_k$, $\mathcal{A}_2$ produces a schedule on a circle with length L. Without loss of generality, we can assume $L = 1$. Then $\max_{r\in[k]} r \cdot v_r = 1$, and by possibly increasing the maximum speed of some agents we may assume $v_i = 1/i$ for each $i \in [k]$. Note that increasing speeds in this way can only increase the maximum circumference that can be patrolled with idle time 1 using these agents, but will not increase the length produced by $\mathcal{A}_2$. Thus, if there is any collection of agents where there is a patrol schedule that performs better than $\mathcal{A}_2$, there must be such a schedule in the case of *harmonic* maximum speeds $1, 1/2, \ldots, 1/k$.

To analyse the performance of patrol schemes on the circle, Czyzowicz et al. considered two different cases: *unidirectional* patrol schedules, where agents are only allowed to move in one direction, and general (or *bidirectional*) patrol schedules, where agents are allowed to go in both directions. Clearly, any patrol schedule obtained through $\mathcal{A}_2$ is unidirectional.

In the bidirectional case, it is not too hard to see that there are situations where $\mathcal{A}_2$ is not optimal. Indeed, in the case of harmonic maxiumum speeds, the partition-based strategy $\mathcal{A}_1$, which works in the same way for a circular fence as for a line segment fence, would give $L = (1 + 1/2 + \cdots + 1/k)/2$, which is bigger than 1 as given by $\mathcal{A}_2$ for any $k \geq 4$. In fact, an example with three agents was given in [3] where neither $\mathcal{A}_1$ nor $\mathcal{A}_2$ are optimal. This was strengthened further by Dumitrescu et al. [8], who showed for any $k \geq 4$ there exists a collection of k agents where what they call the *train strategy* $\mathcal{A}_3$ performs strictly better than both $\mathcal{A}_1$ and $\mathcal{A}_2$. To the authors' knowledge, no universal scheme has been proposed to always produce an optimal patrol schedule in this setting.

For the unidirectional case, it was initially conjectured by Czyzowicz et al. that $\mathcal{A}_2$ is optimal for any set of agents. This was proved to be true for up to four agents. However, it was shown incorrect by parallel results by Dumitrescu et al. [8] and Kawamura and Soejima [10], who gave explicit examples of patrol schemes for 32 and 122 agents (with harmonic

speeds) with $L = 1 + \epsilon$ (for a small unspecified $\epsilon > 0$) and $L = 1.05$ respectively. Kawamura and Soejima further conjectured that the maximum length of a unidirectional circle that can be patrolled by agents with speeds $1, 1/2, \ldots 1/k$ diverges as $k \to \infty$.

2 Our Results

This paper advances the understanding of the fence patrolling problem by giving tight upper and lower bounds on the optimal efficiency for the line segment, and a construction for the circle with efficiency of $\Theta(\frac{1}{\log_e \log_2 k})$ for any set of k agents. To a large extent it concludes the main line of inquiry put forward in the works discussed above [3, 9, 8, 10].

2.1 Results for the Line Segment

We provide the first technique to prove general impossibility results for the fence patrolling problem. We explain our ideas in more detail in Section 4 and merely state our main upper bound here:

▶ **Theorem 2.1.** *Any fence patrol schedule with k agents with maximum speeds $v_1, \ldots, v_k$ patrols a fence of length at most*

$$L \leq \sum_{i=1}^{k} \frac{v_i}{1 + \frac{v_i}{\max_j v_j}}.$$

One way to interpret Theorem 2.1 is that the contribution of an agent a_i depends not only on his/her own speed v_i but also on how much slower he/she is than the fastest agent. In particular, instead of always contributing v_i, as in the trivial upper bound, an agent contributes at most $\frac{1}{1+\frac{1}{s_i}} \cdot v_i$ given that the fastest agent patrolling is a factor of s_i faster than a_i. That is, the "relative efficiency" of an agent a_i ranges anywhere between $1/2$ and 1 depending on s_i, which always constitutes an improvement over the trivial upper bound of $\sum_i v_i$.

We also show that Theorem 2.1 can be used to prove an upper bound on the efficiency of a schedule solely in terms of the number of agents:

▶ **Lemma 2.2.** *Any fence patrolling schedule with k agents has an efficiency of at most $1 - \frac{1}{\sqrt{k}+1}$.*

We note that our upper bounds are tight in several interesting special cases. Specifically, for the case of agents having identical speeds, Theorem 2.1 shows that the efficiency of the schedule (and indeed each agent) is at most $\frac{1}{2}$, reproving the result of [3]. In contrast to the symmetry argument about non-crossing agents explained above, our arguments and upper bounds easily extend to near-identical speeds as well. Lastly, it is easy to check that Theorem 2.1 is tight when applied to the configuration of agents used by Dumitrescu et al. [8] and Kawamura and Soejima [10] for their construction to obtain efficiency ratios of $25/48 - o(1)$ and $2/3 - o(1)$, respectively.

Our upper bounds do not exclude schedules with efficiency close to 1. They do however give important restrictions and clues about what an extremely efficient schedule, if it exists, has to look like. In particular, Lemma 2.2 implies that any schedule with efficiency $1 - \epsilon$ has to have at least $\left(\frac{1}{2\epsilon}\right)^2$, i.e., quadratically in $\frac{1}{\epsilon}$ many agents. In the same manner, Theorem 2.1 implies that, with $\epsilon \to 0$, the ratio between the fastest and slowest agent has to be at least $\Omega(\frac{1}{\epsilon})$, i.e. grow unboundedly. Even more interestingly, the way the upper bound in Theorem 2.1 depends on $\max_i v_i$ seems to indicate that even just a single very fast agent can raise the "relative efficiency" of slower agents from $1/2$ to almost 1.

Equipped with this better understanding and guidance from our impossibility results we were, to our surprise, able to design schedules which achieve an efficiency arbitrarily close to 1, thus disproving the conjecture of [10]:

▶ **Theorem 2.3.** *For any sufficiently large k, there exists a fence patrolling schedule with efficiency $1-\frac{3.5}{\sqrt{k}}$. Such a schedule uses $k-1$ agents of speed one and one agent with maximum speed $\Theta(\sqrt{k})$.*

Note that this theorem implies that for any $\epsilon > 0$ there exists a fence patrolling schedule with efficiency $1-\epsilon$ using $O(\frac{1}{\epsilon^2})$ agents – one with speed $\Theta(\frac{1}{\epsilon})$ and all others with speed 1. In other words, the efficiency can be made arbitrarily close to 1 by choosing the appropriate number and maximum speeds of agents.

We remark that Theorem 2.3 also shows that both our upper bounds are asymptotically tight. In particular, the optimal efficiency for any schedule with k agents is indeed $1-\Theta(\frac{1}{\sqrt{k}})$. Furthermore, for any $s \geq 1$, there is a configuration (with $k = \Theta(s^2)$ agents), where the maximum speeds of the agents differ by a factor s and for which the optimal efficiency is $\frac{1}{1+\frac{1}{\Theta(s)}} = 1-\Theta(\frac{1}{s})$.

2.2 Results for the Circle

We resolve the conjecture by Kawamura and Soejima affirmatively. Namely, for any large enough k, we can construct a patrol schedule with idle time 1 using agents with maximum speeds $1, 1/2, \ldots 1/k$ that patrols a unidirectional circle of length $L = \Theta\left(\frac{\log_2 k}{\log_e \log_2 k}\right)$. In fact, our construction extends to a new universal scheme for the unidirectional circle. This is captured in the following theorem.

▶ **Theorem 2.4.** *For k sufficiently large and for any k agents with maximum speeds $v_1, \ldots v_k$ there exists a patrol scheme with idle time 1 that patrols a unidirectional circle of length*

$$L = \frac{1}{33 \log_e \log_2 k} \sum_{i=1}^{k} v_i.$$

The construction of our schedule has two steps: we first divide the agents into $\Theta(\log_2 k)$ groups, reducing the speed of some and discarding others so that each group consists of a power of 2 number of agents that move with the same speed, which is also a power of 2 times the sum of speeds. This allows us to use a randomized construction, in which the agents from each group are placed equidistantly around the circle with a random offset from some fixed "beginning" of the circle, and move around it with the same speed. We show that with this patrol schedule, most points are visited as frequently as required by our theorem. Then as a second phase, we cut out the bad points – that is, the ones that are not visited as frequently as necessary. We move the patrol schedule to a smaller circle, intuitively only consisting of the good bits. Agents move as if they were on the larger circle, but whenever moving though a cut-out segment, they just stand still instead. The details of this scheme together with a proof sketch will be given in Section 6.

2.3 Organization

The rest of the paper is organized as follows: We first give a more formal model description of the fence patrolling problem as well as discuss some related models and works in Section 3. In Section 4 we explain and prove our upper bounds for the line segment. Section 5 explains

and gives a proof sketch of our optimal fence patrolling schedule for the line segment. Finally, Section 6 presents and and gives a proof sketch our schedule for a circle with length $\Theta(\frac{1}{\log_e \log_2 k} \sum_{i=1}^{k} v_i)$. Formal and complete proofs for the two schedules can be found at `https://arxiv.org/abs/1809.06727`.

3 Fence Patrolling and Related Models

In this section we give a more detailed formal definition for the fence patrolling model/problem and briefly discuss related models and results. The fence patrolling model as given by [3] is defined as follows:

- The *environment* $\mathcal{E}$ to be patrolled is 1-dimensional and consists of a line segment of length L or a circle of circumference L. This line segment or circle is also referred to as a *fence*.
- The fence patrolling problem consists of some finite number $k \in \mathbb{N}$ of mobile agents $a_1, a_2, \ldots, a_k$ to patrol the fence, each having a possibly distinct positive maximum speed $v_1, v_2, \ldots, v_k \in \mathbb{R}_+$.
- A *schedule* for the fence patrolling problem consists of a k-tuple of functions $a_1, a_2, \ldots, a_k : [0, \infty) \to \mathcal{E}$ such that, for all $i \in [k]$, $t \geq 0$ and $\epsilon > 0$,

  $$\text{dist}(a_i(t + \epsilon), a_i(t)) \leq \epsilon \cdot v_i.$$

  That is, we assume patrolling starts at $t = 0$ and goes on indefinitely. Each agent follows a predetermined trajectory, in which he/she moves along $\mathcal{E}$ with at most his/her maximum speed. In the case of a circular fence, the function $\text{dist}(x, y)$ refers to the length of the shorter circle arc between $x, y \in \mathcal{E}$. In the case of the unidirectional circle, we have the additional requirement that $\forall i \in [k], t \geq 0$ and $0 < \epsilon < \frac{L}{2v_i}$, the shorter arc between $a_i(t)$ and $a_i(t + \epsilon)$ is the one that spans clockwise from $a_i(t)$.
- We say that a patrol schedule has *idle time* T for some fixed positive parameter T if for all $t \geq T$ and for all $x \in \mathcal{E}$, there is some agent that visits x during $[t - T, t]$. Intuitively, this condition means that an intruder cannot remain undetected at a point for more than T time.
- Given a patrol schedule, we say that a point $(x, t) \in \mathcal{E} \times [T, \infty)$ is *T-covered* if some agent a_i visits the point $x \in \mathcal{E}$ on the fence during the time interval $[t - T, t]$. Note that in this model an agent patrols/monitors a point $x \in \mathcal{E}$ by visiting it. On the one hand, this means the agents are limited to zero line of sight. On the other hand, no additional operation (e.g. stop and look around) is necessary to patrol a point.

One can see that a schedule has idle time T if and only if every point $(x, t) \in \mathcal{E} \times [T, \infty)$ is T-covered. It is easy to observe that any patrol schedule of a fence of length L with idle time T can be rescaled to a schedule of a fence of length $\alpha \cdot L$ with idle time $\frac{1}{\alpha} \cdot T$ for any $\alpha > 0$. Thus, to simplify terminology, we assume henceforth that $T = 1$ and we refer to 1-covered simply as covered.

Related models have been considered in the literature: where agents have positive line of sight [7], where agents have distinct walking and patrolling speeds [5], where some agents may be faulty [4], where only some regions of the environment need to be patrolled [2], or where the environment is a geometric tree [6]. However, all of these models feature identical agents and in particular do not allow for varying maximum speeds. Overall, the model given above is likely the cleanest and most natural model in which agents with different speeds can and have been studied. Despite the extreme simplicity of this model, this paper and prior

works on the fence patrolling problem [3, 9, 8, 10] show that very surprising and intricate phenomena occur when agents have different speeds and that these nontrivial consequences can be studied in the model defined above.

4 Impossibility Results for the Line Segment: Proof of Theorem 2.1 and Lemma 2.2

In this section, we prove two upper bounds on the length of a straight line fence (i.e. $\mathcal{E} = [0, L]$) patrolled by agents of maximum speeds $v_1, \ldots, v_k$.

Proof of Theorem 2.1. The main idea of the proof is to consider the two-dimensional spacetime continuum $\mathcal{S} := [0, L] \times [0, \infty)$ and the trajectories of the agents along with the points they cover. Since, as noted in Section 3, a patrol schedule with agents $a_1, \ldots, a_k$ has idle time 1 if and only if $\forall x \in [0, L], \forall t \geq 1$, $(x, t) \in \mathcal{S}$ is covered by at least one agent a_j, the theorem can be equivalently stated as that, for any patrol schedule such that all points $(x, t) \in S$ with $t \geq 1$ are covered by some agent, we have

$$L \leq \sum_{i=1}^{k} \frac{1}{\frac{1}{v_i} + \frac{1}{v_{\max}}}. \tag{4.1}$$

In fact, we will show (4.1) under the weaker assumption that only points $(x, t) \in S$ with $t \in [1, 2k]$ are covered by some agent.

Given a patrol schedule of $[0, L]$ with agents $a_1, \ldots, a_k$ and a non-empty subset $A \subseteq \{a_1, \ldots, a_k\}$, we define the *right border* (see Figure 2) of A as the function $B^A : [1, \infty) \to [0, L]$ given by $B^A(t) := \max\{x \in [0, L] : (x, t) \text{ is covered by some agent in } A\}$. We show (4.1) by considering consecutively the collections of agents $A_1, \ldots, A_q$, where $A_1 = \{a_{i_1}\}$, $\forall j \in [q] \setminus \{1\}, A_j = A_{j-1} \cup \{a_{i_j}\}$, $i_1, \ldots, i_q$ is a sequence of distinct integers in $[k]$ to be specified later, and $1 \leq q \leq k$ as we might not need to consider all agents. The intuition behind this is that we are starting with an empty set and adding more agents in a specific order until some termination condition is met. It is clear that $\forall t \geq 0, \forall j \in [q-1], B^{A_j}(t) \leq B^{A_{j+1}}(t)$. The key idea of the proof is to consider what happens to the right border of A_j as j increases (that is, as more agents are added). An example of the right borders of A_j and A_{j+1} for some j is shown in Figure 1.

At this point we prove a claim which will be useful in specifying the sequence $i_1, \ldots, i_q$.

▷ Claim 4.1. For any patrol schedule of $[0, L]$ with set of agents $A = \{a_1, \ldots, a_k\}$ and idle time 1, for any subset A' of $\{a_1, \ldots, a_k\}$, and for any point $(p_x, p_t) \in \mathcal{S}$ on the right border of A' (that is, such that $B^{A'}(p_t) = p_x$), there exists an $\varepsilon > 0$ such that there is at least one agent $a_i \in A \setminus A'$ that covers all points $(p_x + \nu, p_t)$ for $\nu \in [0, \varepsilon]$.

Proof. Note that there could not exist three points $(x_1, t), (x_2, t), (x_3, t)$ such that $0 \leq x_1 < x_2 < x_3 \leq L$ and some agent a_j covers (x_1, t) and (x_3, t) but not (x_2, t). This is because the trajectory of every agent can be considered as a continuous function $f_{a_j} : [0, \infty) \to [0, L]$, so it cannot be that $\exists t_1, t_3 \in [t-1, t]$ such that $f(t_1) = x_1$ and $f(t_3) = x_3$ but $\nexists t_2 \in [t-1, t]$ with $f(t_2) = x_2$. It follows that the set of points C_j on the segment between (p_x, p_t) and (L, p_t) covered by some agent a_j must be either the empty set or a segment. Now consider the set of agents A_p in $A \setminus A'$ that cover (p_x, p_t) and note that $\exists \varepsilon > 0$ such that $\exists a \in A_p$ that covers $(p_x + \varepsilon, p_t)$ (otherwise choose the non-empty C_j with $a_j \in A \setminus A'$ with the leftmost left end (p_l, p_t) that is strictly to the right of p_x and notice that all points $(p_{x'}, p_t)$ with $p_{x'} \in (p_x, p_l)$ are not covered by any agent in A, which is impossible as all points in $\mathcal{S}$ should be covered). But now $a \in A_p$ covers both (p_x, p_t) and $(p_x + \varepsilon, p_t)$, therefore it also covers anything in between since the points it covers between (p_x, p_t) and (L, p_t) must form a segment. ◁

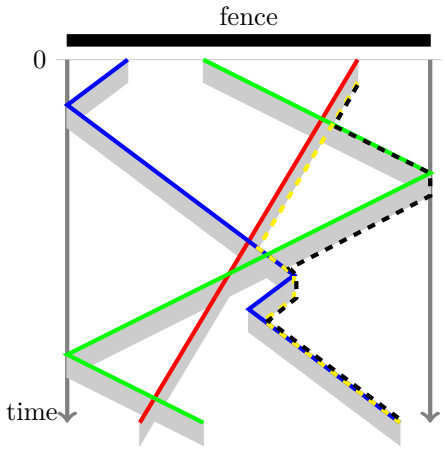


Figure 1 If we have added the blue agent and the red agent so far, then the right border is shown in yellow. If we now also add the green agent, the new right border is shown in black.

Now we can continue with the proof of our main upper bound by specifying $i_1, \ldots, i_q$. Consider a fixed patrol schedule of $[0, L]$ with agents $a_1, \ldots, a_k$ and idle time 1. To pick the sequence $i_1, \ldots, i_q$, consider the following procedure: initially, put $l_0 = (x_0, t_0) = (0, k)$ and pick $i_1 \in [k]$ such that agent a_{i_1} covers l_0. For each consecutive $j = 1, 2, \ldots$, we let $l_j = (x_j, t_j)$ be such that

$$t_j = \arg \min_{t \in [k-j, k+j]} B^{A_j}(t)$$

and $x_j = B^{A_j}(t_j)$. Intuitively, l_j is the leftmost point of B^{A_j} between times $k - j$ and $k + j$. Now if $x_j = L$, we stop adding agents and we set $q := j$. Note that if $j = k$, then $x_j = L$ as agents $a_1, \ldots, a_k$ cover all of $[1, 2k]$. If $x_j < L$, pick as agent $a_{i_{j+1}}$ an agent that covers all the points with coordinates $(x_j + \nu, t_j)$ for $\nu \in [0, \varepsilon]$ for some small enough $\varepsilon > 0$. Such an agent should exist by Claim 4.1. To make sure l_j is always defined for any $j \in \{0, 1, \ldots, q\}$, if $q = k$, set $l_k := (L, k)$.

We note that $x_0, x_1, ..., x_q$ is non-decreasing, $x_0 = 0$ and $x_q = L$ since we either stopped adding agents when $q < k$ because $x_q = L$ or we stopped when $q = k$, in which case all of $[0, L] \times [1, 2k]$ should be covered. Hence the theorem follows if we can show that $\forall j \in \{0, 1, \ldots, q-1\}, x_{j+1} - x_j \leq \frac{1}{\frac{1}{v_{i_{j+1}}} + \frac{1}{v_{\max}}}$.

In order to bound this difference, we investigate how the right border moves when agent $a_{i_{j+1}}$ is added. Note that $\forall t \in [0, \infty)$,

$$B^{A_{j+1}}(t) = \max(B^{A_j}(t), B^{\{a_{j+1}\}}(t)).$$

For any time $t > t_j$, the rightmost point at time t that could be covered by any agent in A_j is $(x_j + (t - t_j)v_{\max}, t)$ since the speed of any agent is at most $v_{\max}$. Similarly, for any time $t < t_j$, the rightmost point at time t that could be covered is $(x_j + (t_j - t)v_{\max}, t)$. Thus, $\forall t \in [0, \infty)$,

$$B^{A_j}(t) \leq x_j + |t - t_j| v_{\max}.$$

Denote by u the ray $(x_j + (t_j - t)v_{\max}, t)$ where $t \leq t_j$, and by w the ray $(x_j + (t - t_j)v_{\max}, t)$ where $t \geq t_j$.

Next, consider $B^{\{a_{j+1}\}}(t)$. Since agent a_{j+1} covers l_j, this means that he/she visits x_j at some time between $t_j - 1$ and t_j, say at point (x_j, t_{visit}). Under the restriction that the trajectory of agent $a_{i_{j+1}}$ should go through (x_j, t_{visit}), it is clear that $B^{\{a_{i_{j+1}}\}}(t)$ is maximized if agent $a_{i_{j+1}}$ comes from the right at maximum speed, hits (x_j, t_{visit}) and then turns around and moves to the right at maximum speed, in which case equality is achieved in

$$\forall t \in [0, \infty), B^{\{a_{i_{j+1}}\}}(t) \leq \min(x' + |t - t'|v_{i_{j+1}}, L),$$

where $(x', t') = (x_j + \frac{v_{i_{j+1}}}{2}, t_{visit} + \frac{1}{2})$. Denote by h the ray $(x' + (t' - t)v_{i_{j+1}}, t)$ where $t \leq t'$ and by g the ray $(x' + (t - t')v_{i_{j+1}}, t)$ where $t \geq t'$.

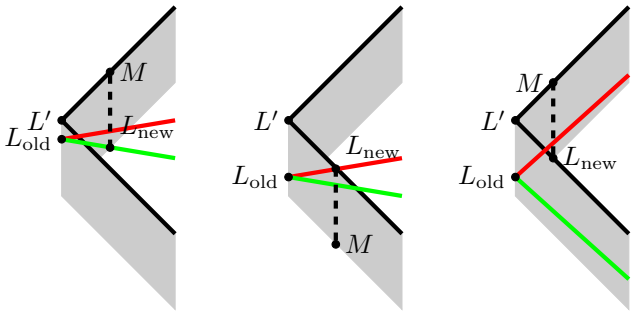


Figure 2 The rays R_1 and R_2 give a bound to the right of the current right border and are given in red and green respectively, and in black and gray we have a bound to the right of the trajectory of the newly added agent a_i and its shadow, i.e. the points that are covered by it.

We thus get $\forall t \in [0, \infty)$,

$$B^{A_{j+1}}(t) = \max(B^{A_j}(t), B^{\{a_{i_{j+1}}\}}(t)) \leq f(t),$$

where $f(t) = \max(x_j + |t - t_j|v_{\max}, x' + |t - t'|v_{i_{j+1}})$. Consider $t_{new} = \arg\min_{t \in [1,\infty)} f(t)$. Let $L_{new} = (x_{new} := f(t_{new}), t_{new})$ be the leftmost point on the aforementioned upper bound on $B^{A_{j+1}}(t)$. Notice that L_{new} is either the intersection of h and w, or the intersection of g and u, or the intersection of g and h. These three cases are illustrated in Figure 2. We have u in red and w in green. Consider the upper bound on $B^{\{a_{i_{j+1}}\}}(t)$ mentioned above. The trajectory of agent $a_{i_{j+1}}$ that would correspond to matching this upper bound is given in black and the points $a_{i_{j+1}}$ would cover if this was his/her trajectory are given in gray. We have that $L_{old} = (x_{old}, t_{old}) := l_j$. It can be seen by inspection of the three cases in Figure 2 that $|t_{new} - t_j| \leq 1$. Then $t_{new} \in [k - (j + 1), k + (j + 1)]$, which makes L_{new} a candidate for l_{j+1}, therefore $l_{j+1} = (x_{j+1}, t_{j+1})$ will have $x_{j+1} \leq x_{new}$. Thus it is enough to show that $x_{new} - x_j \leq \frac{1}{\frac{1}{v_{i_{j+1}}} + \frac{1}{v_{\max}}}$.

We need an upper bound on $d = x_{new} - x_{old}$. We consider the points $M = (x_M, t_M)$ and $N = (x_N, t_N)$ as illustrated in Figure 2, such that in all three cases $x_M = x_{new}$ and $|t_M - t_{new}| = 1$. In Case 1, we consider the segment MN of slope $\frac{1}{v_{i_{j+1}}}$ and $x_M - x_N = d$, and the segment $L_{old}L_{new}$ of w of slope $-\frac{1}{v_{\max}}$ and $x_{new} - x_{old} = d$. This gives us

$$\frac{d}{v_{i_{j+1}}} + \frac{d}{v_{\max}} \leq 1 \Rightarrow d \leq \frac{1}{\frac{1}{v_{i_{j+1}}} + \frac{1}{v_{\max}}}. \tag{4.2}$$

In Case 2, we consider the segment $L_{new}L_{old}$ of u of slope $\frac{1}{v_{\max}}$ and $x_{new} - x_{old} = d$, and the segment NM of slope $-\frac{1}{v_{i_{j+1}}}$ and $x_M - x_N = d$. This implies Equation (4.2) for Case 2

as well. In Case 3, we consider the segment MN of slope $\frac{1}{v_{i_{j+1}}}$ and $x_M - x_N = d$, and the segment NL_{new} of slope $-\frac{1}{v_{i_{j+1}}}$ and $x_{new} - x_N = d$. This means that

$$\frac{2d}{v_{i_{j+1}}} = 1 \Rightarrow d = \frac{v_{i_{j+1}}}{2} \leq \frac{1}{\frac{1}{v_{i_{j+1}}} + \frac{1}{v}}.$$

Therefore, $x_{new} - x_{old} \leq \frac{1}{\frac{1}{v_{i_{j+1}}} + \frac{1}{v_{\max}}}$ as desired. ◀

Proof of Lemma 2.2. We show how $L \leq \left(1 - \frac{1}{\sqrt{k}+1}\right) \sum_{i=1}^{k} v_i$ follows from Theorem 2.1. First note that $L \leq \sum_{i=1}^{k} v_i - \frac{v_{\max}}{2}$ as each agent a_i contributes at most $v_i \cdot \frac{1}{1+\frac{v_i}{v_{\max}}} \leq v_i$ while the agent with maximum speed contributes exactly $\frac{v_{\max}}{2}$. Therefore, if $v_{\max} \geq \frac{1}{\sqrt{k}} \sum_{i=1}^{k} v_i$ the desired upper bound for L follows immediately. It remains to deal with the case $v_{\max} < \frac{1}{\sqrt{k}} \sum_{i=1}^{k} v_i$. For this we first note that $x \cdot \frac{1}{1+\frac{x}{v_{\max}}} = \frac{1}{\frac{1}{x}+\frac{1}{v_{\max}}}$ is a concave function in x for $0 \leq x \leq v_{\max}$, since the second derivative $-\frac{2}{(1+\frac{x}{v_{\max}})^3 v_{\max}}$ is always negative. This allows us to apply Jensen's inequality and thus we have

$$L \leq \sum_{i=1}^{k} \frac{1}{\frac{1}{v_i} + \frac{1}{v_{\max}}} \leq k \frac{1}{\frac{1}{v_{avg}} + \frac{1}{v_{\max}}} = \frac{1}{1 + \frac{\sum_{i=1}^{k} v_i}{k \cdot v_{\max}}} \sum_{i=1}^{k} v_i \leq \left(1 - \frac{1}{\sqrt{k}+1}\right) \sum_{i=1}^{k} v_i,$$

where $v_{avg} = \frac{1}{k} \sum_{i=1}^{k} v_i$. This concludes the proof of Lemma 2.2. ◀

5 A Schedule with Efficiency $1 - \epsilon$ for the Line Segment: Proof of Theorem 2.3

In this section, we give proof sketch that for any k agents, there exist speeds $v_1, \ldots, v_k$ and a scheme for these agents to patrol a fence of length

$$L = \left(1 - \frac{3.5}{\sqrt{k}} + O(1/k)\right) \sum_{i=1}^{k} v_i.$$

This improves the result from [10] and therewith falsifies the corresponding conjecture stated in that paper.

Proof sketch of Theorem 2.3. Assume k is sufficiently large, and, for ease of notation, define $n := k - 2$. Let $L = n - 3/2\sqrt{n}$. We construct a schedule that patrols $\mathcal{E} = [0, L]$ with idle time 1, using $n + 1$ agents with maximum speed 1 and 1 agent with maximum speed $2\sqrt{n} - 1$. Thus we have a total speed of $V = \sum_{i=1}^{k} v_i = n + 2\sqrt{n}$. As the ratio between L and V approaches $1 - \frac{3.5}{\sqrt{k}} + O(1/k)$, Theorem 2.3 follows.

To simplify the presentation of the patrol schedule, we will allow the agents to occasionally "step out of the fence $[0, L]$", i.e. we allow an agent a_i to assume positions $a_i(t) < 0$ and $a_i(t) > L$ (to avoid this, we could also modify the schedule so that they stay at the respective end of the fence for a while). To keep the notation as clean as possible, we henceforth assume that n is a square number. Our schedule works as follows (see Figure 3 for a graphical representation):

SLOW AGENTS $a_1, \ldots, a_n$: For each $i \in \{0, \ldots, n\}$, agent a_i starts at time 0 at position $x = i - i/\sqrt{n}$ and moves $i/(2\sqrt{n})$ time units to the right. Then he or she repeats:

- move to the left for $\sqrt{n}$ time units.
- move to the right for $\sqrt{n}$ time units.

FAST AGENT a_{n+1}: The fast agent a_{n+1} starts at time 0 and repeats the following four steps:

(1) Move from position 0 to position $L + 1/2$ with speed $2\sqrt{n} - 1$.

(2) Move from position $L + 1/2$ to position $-1/2$ during the next $\sqrt{n}/2 + 1$ time units (e.g. with constant speed $(L + 1)/(\sqrt{n}/2 + 1) = 2\sqrt{n} - 7 + (16)/(\sqrt{n} + 2)$).

(3) Move from position $-1/2$ to position L with speed $2\sqrt{n} - 1$.

(4) Move from position L to position 0 in the next $\sqrt{n}/2$ time units (e.g. with constant speed $(L)/(\sqrt{n}/2) = 2\sqrt{n} - 3$).

The idea behind our patrol schedule is to initially place the agents with maximum speed 1 equidistantly along the fence with gaps of length slightly smaller than 1, similar to the schedule for the fast agents in [10]. In contrast to their schedule, this is performed slightly out of phase between the agents. This will cover most of the points on the fence. The only problem appears whenever the agents turn around, as then the points right next to these turning points are not visited for more than 1 time unit, hence creating uncovered triangles in the "spacetime" diagram (white triangles in Figure 3). By timing the turning times of the agents appropriately, we ensure that these uncovered triangles are placed such that they can all be cleaned up by the last fast agent. Figure 3 gives a complete illustration of our schedule for 18 agents and intuitively shows that each point x on the line L is visited at least once every unit of time. ◀

6 A schedule for the unidirectonal circle: Proof of Theorem 2.4

In this section, we will present a schedule with which a group of agents $a_1, \dots, a_k$ with maximum speeds $v_1, \dots, v_k$ can patrol a circle with circumference

$$\frac{1}{33 \log_e \log_2(k)} \sum_{i=1}^{k} v_i,$$

followed by a short sketch of why the proposed schedule behaves as claimed. Our schedule will be divided in two parts. First, we will give a randomized construction for a strategy which allows the agents $a_1, \dots, a_k$ to patrol a circle with circumference 1 such that with probability $1 - o(1)$ "most" of the points are visited at least every $\Theta\left(\log\log(k)/(\sum_{i=1}^{k} v_i)\right)$ time units. Then, we will argue that one can remove the "bad" points and then "blow up" the circle to obtain the desired result.

Proof sketch of Theorem 2.4. We are given a group $a_1, \dots, a_k$ of agents with maximum speeds $v_1 \dots, v_k$ and define $V := \sum_{i=1}^{k} v_i$. We propose the following schedule:

Circle Schedule:

(1) Round speeds down to the next power of 2 and omit too slow agents:
For all $i \in [k]$ let $j_i \in \mathbb{N}$ be the non-negative integer such that $V \cdot 2^{-j_i} \leq v_i < V \cdot 2^{-j_i+1}$. We define $v'_i := V \cdot 2^{-j_i}$ and $I := \left\{i \in [k] : v'_i \geq \frac{V}{4 \cdot k}\right\}$.

(2) Group remaining agents according to their speed:
For all $i \in \{0, \dots, \lceil \log_2(k) \rceil + 2\}$ define $G_i := \left\{j \in I : v'_j = V \cdot 2^{-i}\right\}$.

(3) Reduce number of agents in each group to a power of 2:
For all $i \in \{0, \dots, \lceil \log_2(k) \rceil + 2\}$ let $h_i \in \mathbb{N}$ be the positive integer such that $2^{h_i} \leq |G_i| < 2^{h_i+1}$ and let $G'_i \subseteq G_i$ be an arbitrary subset of G_i of size 2^{h_i}. We denote by $m'_i = |G'_i| \cdot v'_a = V \cdot 2^{h_i - i}$ the *mass of the group* G'_i, where $a \in G'_i$ (that is, each agent in G'_i has maximum speed v'_a after the rounding down in step 1).

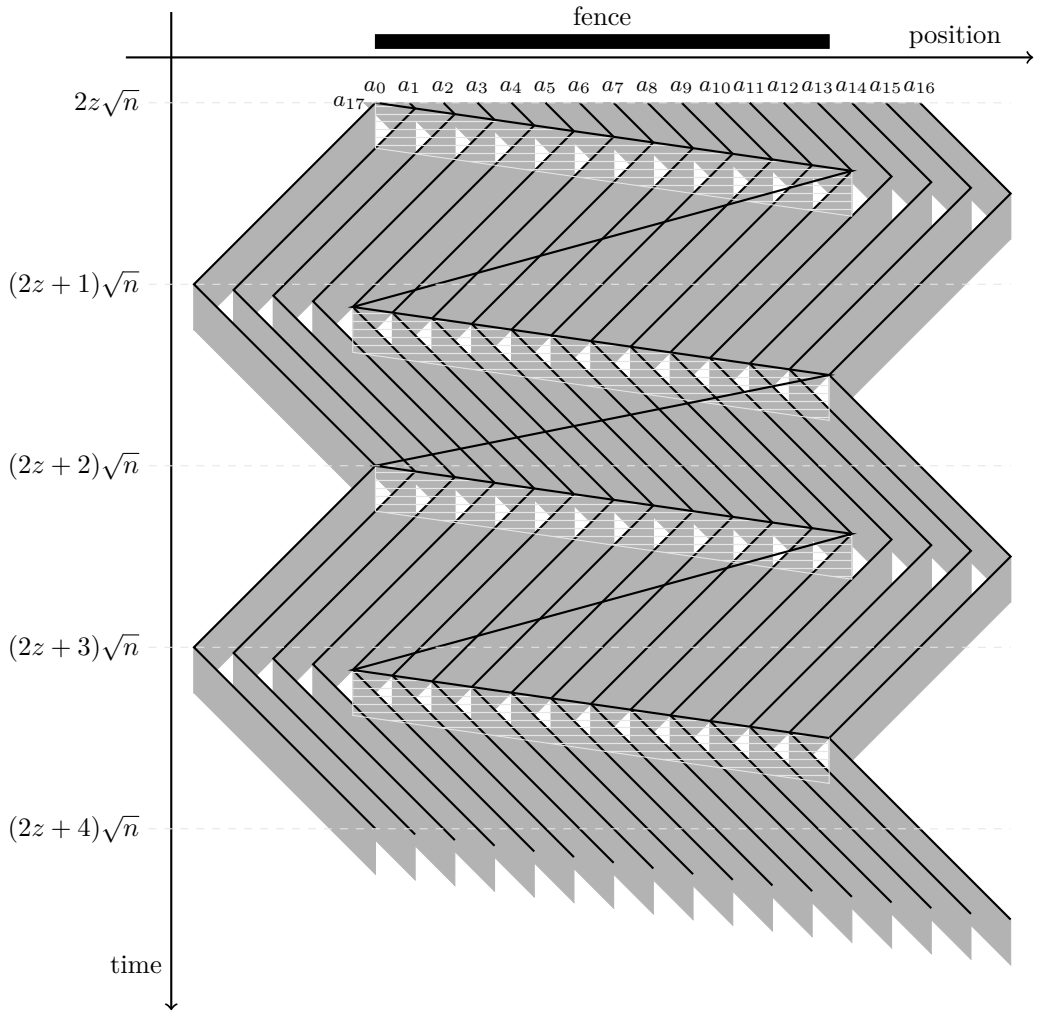


Figure 3 The described schedule for $n = 16$. The dark grey area describes the points (x, t) which are covered by the slow agents $a_0, \ldots, a_n$ while the light grey shaded area describes the points (x, t) which are covered by the fast agent in steps (1) and (3) of his protocol.

(4) Omit groups with too small mass:
Let $J := \left\{ i \in \{0, \ldots, \lceil \log_2(k) \rceil + 2\} : m'_i \geq \frac{V}{16(\lceil \log_2(k) \rceil + 3)} \right\}$.

(5) Patrol schedule: For each $j \in J$, pick an independent uniform random number r_j in the interval $[0, \frac{1}{|G'_j|})$. At time 0, we place the $|G'_j|$ agents from the group G'_j at positions

$$r_j, r_j + \frac{1}{|G'_j|}, \ldots, r_j + \frac{|G'_j| - 1}{|G'_j|},$$

i.e. we place all agents from the same group equidistantly from each other along the circle with a random offset from the origin. Then the agents G'_j walk along the circle with speed v'_j at all times.

Note that the two rounding steps and the two omitting steps each reduced the total "available" speed by a factor of at most 2, therefore we have that $\sum_{j \in J} m'_j \geq V/16$. Defining

$$T = \frac{1}{\min_{j \in J} m'_j} \leq 16(\lceil \log_2(k) \rceil + 3)/V,$$

we observe that after T time units the distribution of agents along the circle repeats itself, i.e. if an agent with speed s is located at position x at time t, then another (or the same) agent of speed s is located at position x at time $t+T$. Cutting the time interval into small pieces of size $\tau \approx 16 \log_e \log_2(k)/V$, one can show that with probability $1-o(1)$, almost all points get visited at least once in each of these small time intervals, and hence all these points get visited at least every 2τ time units. Next, we can "cut away" the few "bad" points for which this is not true and obtain a circle of circumference $1-\epsilon$. Adjusting the above schedule so that agents "stand still" on the smaller circle whenever they cross a "bad" point on the original circle gives us a schedule for which each point (on the small circle) is visited at least every 2τ time units. Rescaling this patrol schedule then gives the scheme to patrol a circle with circumference $(1-\epsilon)\frac{1}{2\tau} \geq \frac{1}{33 \log_e \log_2(k)} \sum_{i=1}^{k} v_i$ as desired. ◀

References

1. Yann Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In *Proceedings. IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2004. (IAT 2004).*, pages 302–308, 2004. doi:10.1109/IAT.2004.1342959.
2. Andrew Collins, Jurek Czyzowicz, Leszek Gasieniec, Adrian Kosowski, Evangelos Kranakis, Danny Krizanc, Russell Martin, and Oscar Morales Ponce. Optimal patrolling of fragmented boundaries. In *SPAA*, pages 241–250. ACM, 2013.
3. Jurek Czyzowicz, Leszek Gąsieniec, Adrian Kosowski, and Evangelos Kranakis. Boundary patrolling by mobile agents with distinct maximal speeds. In *European Symposium on Algorithms*, pages 701–712. Springer, 2011.
4. Jurek Czyzowicz, Leszek Gasieniec, Adrian Kosowski, Evangelos Kranakis, Danny Krizanc, and Najmeh Taleb. When Patrolmen Become Corrupted: Monitoring a Graph Using Faulty Mobile Robots. *Algorithmica*, 79(3):925–940, 2017. doi:10.1007/s00453-016-0233-9.
5. Jurek Czyzowicz, Konstantinos Georgiou, Evangelos Kranakis, Fraser MacQuarrie, and Dominik Pajak. Distributed Patrolling with Two-Speed Robots (and an Application to Transportation). In Begoña Vitoriano and Greg H. Parlier, editors, *Operations Research and Enterprise Systems*, pages 71–95, Cham, 2017. Springer International Publishing.
6. Jurek Czyzowicz, Adrian Kosowski, Evangelos Kranakis, and Najmeh Taleb. Patrolling Trees with Mobile Robots. In Frédéric Cuppens, Lingyu Wang, Nora Cuppens-Boulahia, Nadia Tawbi, and Joaquin Garcia-Alfaro, editors, *Foundations and Practice of Security*, pages 331–344, Cham, 2017. Springer International Publishing.
7. Jurek Czyzowicz, Evangelos Kranakis, Dominik Pajak, and Najmeh Taleb. Patrolling by Robots Equipped with Visibility. In Magnús M. Halldórsson, editor, *Structural Information and Communication Complexity*, pages 224–234, Cham, 2014. Springer International Publishing.
8. Adrian Dumitrescu, Anirban Ghosh, and Csaba D Tóth. On Fence Patrolling by Mobile Agents. *The Electronic Journal of Combinatorics*, 21(3):P3–4, 2014.
9. Akitoshi Kawamura and Yusuke Kobayashi. Fence patrolling by mobile agents with distinct speeds. *Distributed Computing*, 28(2):147–154, 2015.
10. Akitoshi Kawamura and Makoto Soejima. Simple strategies versus optimal schedules in multi-agent patrolling. In *International Conference on Algorithms and Complexity*, pages 261–273. Springer, 2015.
11. Aydano Machado, Geber Ramalho, Jean-Daniel Zucker, and Alexis Drogoul. Multi-agent Patrolling: An Empirical Analysis of Alternative Architectures. In Jaime Simão Sichman, Françcois Bousquet, and Paul Davidsson, editors, *Multi-Agent-Based Simulation II*, pages 155–170, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
12. David Portugal and Rui Rocha. A Survey on Multi-robot Patrolling Algorithms. In Luis M. Camarinha-Matos, editor, *Technological Innovation for Sustainability*, pages 139–146, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

Matroid Coflow Scheduling

Sungjin Im
University of California at Merced, USA
sim3@ucmerced.edu

Benjamin Moseley
Carnegie Mellon University, Pittsburgh, PA, USA
moseleyb@andrew.cmu.edu

Kirk Pruhs
University of Pittsburgh, PA, USA
kirk@cs.pitt.edu

Manish Purohit
Google, Mountain View, CA, USA
mpurohit@google.com

Abstract

We consider the matroid coflow scheduling problem, where each job is comprised of a set of flows and the family of sets that can be scheduled at any time form a matroid. Our main result is a polynomial-time algorithm that yields a 2-approximation for the objective of minimizing the weighted completion time. This result is tight assuming $P \neq NP$. As a by-product we also obtain the first $(2+\epsilon)$-approximation algorithm for the preemptive concurrent open shop scheduling problem.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases Coflow Scheduling, Concurrent Open Shop, Matroid Scheduling

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.145

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Funding *Sungjin Im*: Supported in part by NSF grants CCF-1409130 and CCF-1617653.
Benjamin Moseley: Supported in part by a Google Research Award and NSF grants CCF-1617724, CCF-1733873 and CCF-1725543.
Kirk Pruhs: Supported in part by NSF grants CCF-1421508 and CCF-1535755, and an IBM Faculty Award.

Acknowledgements We thank the anonymous reviewers for their thorough reviews and many helpful suggestions.

1 Introduction

Coflows were introduced in [5] as: "We propose coflows, a networking abstraction to express the communication requirements of prevalent data parallel programming paradigms. Coflows make it easier for the applications to convey their communication semantics to the network, which in turn enables the network to better optimize common communication patterns." Data parallel application frameworks such as MapReduce [9] and Spark [31] have a unique processing pattern that interleaves local computation with communication across machines. Due to the size of the large data sets processed, communication often tends to be a bottleneck in the performance of these platforms and the coflow model abstracts out this bottleneck. Theoretical work on coflow scheduling has primarily focused on the switch model (also called matching model) where the underlying network is assumed to have full-bisection bandwidth and the set of flows that can be scheduled at any time step is restricted to be form a matching.


While there are several reasonable formulations/models of scheduling coflows, the following will be convenient for our purposes. The input consists of a collection J of jobs, where each job $j \in J$ is comprised of a set U_j of tasks (also called flows), a non-negative integer w_j and a release time r_j. Each task $e \in U_j$ has a processing requirement p_e. For example, in the setting of a network supporting MapReduce [9] computations, each job could be a MapReduce job, and a task/flow could represent a required communication within a shuffle phase of a job. Let $U = \cup_{j \in J} U_j$ be the collection of all tasks. Further the input contains a downward-closed set system $\mathcal{M} = (U, \mathcal{I})$. Here $\mathcal{I} \subseteq 2^U$ and elements of $\mathcal{I}$ are called the *independent sets* of $\mathcal{M}$. Conceptually a collection of tasks is independent (and in $\mathcal{I}$) if they can be simultaneously scheduled by the network. A feasible output is a schedule σ that schedules all the flows. That is for each integer time t, σ specifies a collection σ_t of tasks processed/scheduled at time t. In order to be feasible, σ must satisfy the conditions that:

- every task $e \in U$ is scheduled for p_e time steps, and
- at each time t, the scheduled tasks/flows σ_t are in $\mathcal{I}$.

A job j completes at the first time C_j such that every task in U_j has been scheduled fully. The objective is to minimize the total weighted completion time of the jobs. That is, to minimize $\sum_j w_j C_j$.

In this paper, we consider coflow scheduling when the set system $\mathcal{M}$ forms a matroid. The starting point for our investigations is the question whether there is an algorithm to effectively schedule coflows that involve aggregating information, stored at various locations in a network, to a common sink location. Such gathering communication patterns were identified as common in [5]. We model aggregation communications by assuming that for each job j, U_j is a collection of locations in the network where the units of information needed for job j are stored. It is natural to define the independent sets to be locations that can simultaneously be routed to the sink without violating any capacity constraint of the network. In this case, $\mathcal{M}$ is a matroid, and more specifically, a gammoid. Note that the symmetric problem, of disseminating data from a fixed location to various locations in the network, is also common, and essentially equivalent to the aggregation problem.

The matroid coflow scheduling problem as defined here also naturally captures a number of well-studied scheduling problems.

- Parallel Identical Machines Scheduling: Each job j has a single task. The matroid $\mathcal{M} = (U, \mathcal{I})$ is the uniform matroid of rank m, i.e., any set of m jobs can be scheduled in parallel.
- (Preemptive) Concurrent Open Shop Scheduling: In the concurrent open shop scheduling problem, each job j comprises of m tasks, one on each machine, i.e. $U_j = \{t_{ij}\}_{i=1}^m$. Task t_{ij} needs to be scheduled for time p_{ij} and the job is completed when all its tasks are completed. To model this setting, consider $T_i = \{t_{ij}\}_{j=1}^n$ to be set of all tasks that need to be scheduled on machine i. $\mathcal{M}$ is a partition matroid that ensures that a set S of tasks is independent if and only if $|S \cap T_i| \leq 1$ for each machine i.

1.1 Our Contributions

We first consider coflow scheduling on unit length tasks when $\mathcal{M}$ is a matroid. Our main result is:

▶ **Theorem 1.** *There is a deterministic polynomial-time algorithm for coflow scheduling with unit length tasks, when $\mathcal{M}$ is a matroid, that is 2-approximate with respect to the objective of minimizing total weighted completion time.*

We note that Theorem 1 can be extended to the case that tasks may have arbitrary processing times, albeit at a slight loss in the approximation factor.

▶ **Theorem 2.** *There is a deterministic polynomial-time algorithm for coflow scheduling with arbitrary length tasks, when $\mathcal{M}$ is a matroid, that is $(2+\epsilon)$-approximate with respect to the objective of minimizing total weighted completion time, for any constant $\epsilon > 0$.*

As with all the approximation results for coflow scheduling in the literature, our algorithm is based on rounding a natural time-indexed linear program. Intuitively the rounding extracts a deadline C_j^* for each job j. This time is roughly $1/\lambda$ times later than the first time when every task in U_j has been scheduled at least to the extent λ in the solution to LP. Here the value of λ is randomly chosen. The expected value of C_j^* is shown to be at most twice the fractional completion time for j in the solution to LP; this "stretching" (also called slow-motion) idea has been used in other scheduling contexts [12, 22, 27]. This can be viewed as deriving from the LP a fractional schedule where each job j is fully completed by time C_j^*. Then, we observe that the problem of scheduling tasks to meet the C_j^* deadlines can be expressed as a matroid intersection problem. As the matroid intersection polytope is integral [26], one can find an integral schedule meeting these deadlines. Finally, by derandomizing the random choice of λ, we derive our main theorem.

The approximation guarantee in Theorem 1 is tight assuming $P \neq NP$. This is because it is NP-hard to approximate the total weighted completion time for concurrent open shop (even with unit sized tasks) within a factor of $2-\epsilon$ [23], and this problem is a special case of matroid coflow scheduling, where the matroid is a partition matroid. Somewhat surprisingly, even for the concurrent open shop scheduling with release times, the previous best known approximation factor was 3 [10, 17]. (See also additional discussion in [2].) Thus, Theorem 2 immediately yields an improved approximation algorithm for preemptive concurrent open shop with arbitrary release times.

▶ **Corollary 3.** *There is a deterministic, polynomial-time $(2+\epsilon)$ approximation algorithm for the preemptive concurrent open shop scheduling problem when jobs have arbitrary release times, for any constant $\epsilon > 0$. If all the release times and processing requirements are polynomially bounded, then the approximation guarantee improves to 2.*

We believe our primary technical contribution is the high-level approach to reduce a weighted completion time scheduling problem to a deadline-constrained scheduling problem. Our approach to first extract a deadline for each job from the LP solution and then finding an integer schedule that meets those deadlines can be viewed as a strict generalization of processing jobs in increasing order of their completion time derived from the LP, which has been a very common rounding tool in scheduling literature; e.g. [21, 28, 2]. Our novel approach allows us to handle the matroid constraint, which we believe is natural and quite general.

1.2 Related Results

Most of the theoretical/algorithmic work on coflows has been on matching coflows [20, 16, 15, 2, 1]. These results essentially abstract out the network by modeling the network as an n-by-n switch, or equivalently a complete bipartite graph, and by modeling supportable flows by matchings in the graph. This is well motivated in practice as the networks in many data centers are hierarchical, with higher network elements having higher capacities. Thus a matching between servers at leaves of the network is a not unreasonable approximation of a communication supportable by the network. We note that matching coflows correspond

to coflows in our framework when the set system $\mathcal{M}$ is an intersection of two partition matroids. The first constant (16.54) approximation for coflow scheduling in this model was given in [20]. Currently the best known approximation ratios are 5 for when jobs may have variable release times, and 4 when all jobs arrive at time 0 [2, 29], respectively. Note that the 2-approximation algorithms claimed in [18] and [11] are both flawed; see [2] and [11] for the discussion of the flaws.

Jahanjou et al. [14] consider several problems where there is an underlying network with capacities on the edges. If the tasks are paths in the network, and $\mathcal{I}$ consists of collections of paths that don't collectively violate any edge capacity, then their work gives a algorithm for producing a fractional schedule (which is equivalent to time being continuous) that is $O(1)$-approximate with respect to total weighted completion time. If the tasks are (source, sink) pairs in the network, and $\mathcal{I}$ consists of collections of (source, sink) pairs that can be simultaneously routed without violating any edge capacity, then their work gives a algorithm for producing a fractional unsplittable schedule that is $O(\log E / \log \log E)$-approximate with respect to total weighted completion time, together with a matching hardness result; here, E is the number of edges. Our work is not comparable to theirs since different constraints are addressed and our focus is on integer schedules in contrast to theirs on fractional schedules.

Coflow scheduling is a generalization of the classical concurrent open shop scheduling problem [3, 4, 10, 17, 19, 23, 30]. Several 2-approximation algorithms were shown [4, 10, 17] via LP rounding. Matching hardness results were shown in [3, 23]. When jobs have different release times, the same LP relaxations yielded 3-approximations [10, 17]. Later, [19] gave a simple greedy algorithm that matches the best approximation ratio when all jobs arrive at time 0. Recently, [2] gave a combinatorial 3-approximation via a primal-dual analysis when jobs have non-uniform release times.

Coflow scheduling has been actively studied within the networking community; some examples include [5, 6, 7, 18, 32].

1.3 Organization

The rest of the paper is organized as follows. In Section 2 we give some basic definitions and notation. In Section 3 we give the linear programming formulation. In Section 4 we explain how to round a solution to the linear program. In Section 5 we discuss the derandomization. In Section 6, we discuss the extension to tasks with variable processing times.

2 Definitions and Notations

We first consider the matroid coflow scheduling problem with unit length tasks. We will discuss three types of schedules, and two types of objectives. In a discrete-time schedule, we consider that time is divided into unit length intervals (also called time slots), and the schedule specifies the set of jobs processed during each time slot. We let time slot t refer to the interval of time $(t-1, t]$. In an *integer* discrete-time schedule, at each time slot t, an independent set in the matroid is scheduled. In a *fractional* discrete-time schedule, at each time slot t, a convex combination of independent sets from the matroid are scheduled. In other words, in such a fractional schedule, the set of tasks scheduled at time slot t can be expressed as $\sum_{S \in \mathcal{I}} \alpha_S 1_S$, where $\sum_{S \in \mathcal{I}} \alpha_S = 1$, and 1_S is the characteristic vector corresponding to independent set $S \in \mathcal{I}$. A valid feasible solution is restricted to be an integer discrete-time schedule. On the other hand, during our analysis, we will also consider *continuous* schedules. A continuous schedule specifies an independent set of tasks to be scheduled at each instantaneous time τ (as opposed to during a unit-length time slot).

The completion time C_j of a job j is the first time when all tasks in U_j have been completed. We let $q_e(t) : [0, T] \to \{0, 1\}$ denote an indicator function defined for each task $e \in U$, where $q_e(t) = 1$ if and only if task e (more precisely an independent set including e) is scheduled at time t in σ. We let $Q_e(t) = \int_{\tau=0}^{t} q_e(\tau)d\tau$ denote the extent to which task e is scheduled by time t. Let $\tilde{C}_j(v)$ denote the first time when every task in U_j has been scheduled by extent at least v. The fractional completion time of job j is then $\tilde{C}_j = \int_{v=0}^{\infty} \tilde{C}_j(v)dv$. We will use $\text{COST}(\mathsf{LP})$ to denote the optimum objective of the LP, which we will describe soon.

3 Linear Program

In this section we give a linear programming formulation LP of our matroid coflow problem when tasks have unit lengths. Let $x_{j,t}$ be an indicator variable that specifies whether job j completes at time t. For a task $e \in U_j$, let $y_{e,t}$ be an indicator variable that specifies whether task e is assigned to time slot t. Let $\rho(S)$ be the rank function of the matroid.[1] Let $T = |U|$ be an upper bound on the time by which all tasks can be completed. The formulation of LP is then:

$$\mathsf{LP}: \qquad \min \sum_{j \in J} w_j \sum_{t \in [T]} t \cdot x_{j,t}$$

$$\text{s.t.} \quad \forall j \in J, \qquad \sum_{t} x_{j,t} = 1 \tag{1}$$

$$\forall j \in J \text{ and } \forall e \in U_j \text{ and } \forall t \in [T], \qquad \sum_{s \le t} y_{e,s} \ge \sum_{s \le t} x_{j,s} \tag{2}$$

$$\forall S \subseteq U \text{ and } \forall t \in [T], \qquad \sum_{e \in S} y_{e,t} \le \rho(S) \tag{3}$$

$$\forall j \in J \text{ and } \forall e \in U_j \text{ and } \forall t \in [r_j - 1], \qquad y_{e,t} = 0 \tag{4}$$

$$\mathbf{x}, \mathbf{y} \ge 0 \tag{5}$$

Constraint (1) ensures that every job is scheduled. Constraint (2) ensures that all tasks of a job j are scheduled to at least the extent that j is completed by time t. Constraint (3) ensures that at any time step t, the set of tasks assigned to t form an independent set in the given matroid. Constraint (3) is the only constraint set that can potentially have a super-polynomial size. However, for each fixed time t, the constraint is just a polymatroid, and therefore, admits an efficient separation oracle [8, 24, 13]. In case that there are arrival/release times, constraint (4) ensures that no tasks in U_j are processed before j's release time r_j. The objective of LP is fractional weighted completion time.

Note that a solution to LP can be viewed as a fractional discrete schedule. We will use $X_{j,t} := \sum_{s \le t} x_{j,s}$ to denote the extent to which job j has been processed by time t, and use $Y_{e,t} := \sum_{s \le t} y_{e,s}$ to denote the extent to which task e has been processed by time t.

4 Rounding

In this section, we show how to round an optimal solution to LP to obtain a 2-approximate integral (discrete) schedule. For each job j and $v \in (0, 1]$, define $\bar{C}_j(v) = \frac{1}{x_{j,t}}(v - X_{j,t-1}) + (t - 1)$ if $v \in (X_{j,t-1}, X_{j,t}], t \in [T]$. Intuitively, $\bar{C}_j(v)$ is a linear interpolation of the discrete

[1] $\rho(S)$ is defined as $\max_{S' \subseteq S : S' \in \mathcal{I}} |S'|$.

times when job j is partially completed. We set a deadline $C_j^* = \lceil \frac{1}{\lambda} \bar{C}_j(\lambda) \rceil$ for each job j, where $\lambda \in (0, 1]$ is randomly drawn according to the probability density function $f(v) = 2v$. A key portion of the analysis is to show that the expected value of each $w_j C_j^*$ is at most twice the contribution of job j to the LP objective.

To analyze the expected value of C_j^*, we construct several schedules from the LP solution. In Subsection 4.1, we will show how to convert a solution of LP to a continuous schedule σ. In Subsection 4.2 we show how to convert σ into a stretched schedule σ^λ, which is another continuous schedule parameterized by $\lambda \in (0, 1]$. Finally, in Subsection 4.3 we will show how to convert this continuous schedule into (discrete-time) integer schedule with the same cost. We note that we construct schedules in Subsection 4.1 and 4.2 only for the sake of analysis. That is, we can obtain a 2-approximate integral discrete schedule only using the rounding algorithm in Subsection 4.3 with the deadlines $\{C_j^*\}_j$.

4.1 Constructing the Continuous Schedule σ

We construct a continuous schedule σ from the solution to LP. For each time t, we first decompose $\{y_{e,t}\}_{e \in U}$ into a convex combination $\sum_{S \in \mathcal{I}} \alpha_S 1_S$ of independent sets.[2] To create σ this convex combination is "smeared" across all instantaneous times during $(t-1, t]$. That is, in σ each independent set S is scheduled for $\alpha_i(\tau_2 - \tau_1)$ time units during each infinitesimal time interval $(\tau_1, \tau_2] \in (t-1, t]$. This is formalized in Proposition 4. In Lemma 5 we show that the first time when a job j is scheduled to extent v in σ is at most $\bar{C}_j(v)$. In Lemma 6 we show that the fractional weighted completion time of σ is a bit less than the objective value of the solution to LP. This is because any processing of job j done during $(t-1, t]$ has no effect until time t on the LP objective, whereas it can have effect on j's fractional weighted completion time of σ during $(t-1, t]$, before time t.

▶ **Proposition 4.** *Consider the schedule σ. For any integer $t \in [T]$ and $(\tau_1, \tau_2] \in (t-1, t]$, we have, $\int_{\tau=\tau_1}^{\tau_2} q_e(\tau) d\tau = y_{e,t}(\tau_2 - \tau_1)$.*

▶ **Lemma 5.** *Consider the schedule σ. For any j and $v \in (0, 1]$,*

$$\tilde{C}_j(v) \leq \bar{C}_j(v) =: \frac{1}{x_{j,t}}(v - X_{j,t-1}) + (t-1) \text{ if } v \in (X_{j,t-1}, X_{j,t}], t \in [T],$$

and $\tilde{C}_j(0) = 0$.

Proof. By definition, we have $\tilde{C}_j(0) = 0$, so let us assume that $v > 0$. We first show that $\tilde{C}_j(X_{j,t}) = t$. Due to constraint (2), $Y_{e,t} \geq X_{j,t}$ for all $e \in U_j$. Thus, by construction of σ, all tasks in U_j are processed by at least $X_{j,t}$ by time t, i.e., $Q_e(t) \geq X_{j,t}$, meaning that $\tilde{C}_j(X_{j,t}) \leq t$. We also have that $\tilde{C}_j(X_{j,t}) \geq t$ since we know by the optimality of the LP solution that $Y_{e,t} = X_{j,t}$ for some $e \in U_j$, therefore, $Q_e(t) = X_{j,t}$. Thus, we have $\tilde{C}_j(X_{j,t}) = t = \bar{C}_j(X_{j,t})$.

Now consider an arbitrary $v \in (0, 1]$. Let $t \in [T]$ be such that $v \in (X_{j,t-1}, X_{j,t}]$. Then, it follows that $x_{j,t} \neq 0$. Thus, from the above argument, we have $\tilde{C}_j(X_{j,t}) = t$. Let $t_v := \bar{C}_j(v)$ for notational convenience. We want to show $\tilde{C}_j(v) \leq t_v$. By Proposition 4 and construction of σ, we know that the extend to which e is processed by time t_v,

$$Q_e(t_v) = Y_{e,t-1} + y_{e,t}(t_v - (t-1)) = Y_{e,t-1} + \frac{y_{e,t}}{x_{j,t}}(v - X_{j,t-1})$$

[2] This is possible because $\{y_{e,t}\}_e$ lies in the polymatroid associated with the matroid rank function ρ due to constraint (3). It is well-known that this polymatroid is equivalent to the independence set polytope of the matroid, meaning that $\{y_{e,t}\}_e$ can be expressed as a convex combination of characteristic vectors of some independent sets. For more details, see Chapter 44 of [25].

First, if $y_{e,t} \geq x_{j,t}$, we immediately have $Q_e(t_v) \geq v + Y_{e,t-1} - X_{j,t-1} \geq v$ due to constraint (2). Otherwise, since $\frac{1}{x_{j,t}}(v - X_{j,t-1}) \leq 1$, fixing the value of $Y_{e,t} = Y_{e,t-1} + y_{e,t}$, the right-hand-side decreases when we increase $y_{e,t}$. Therefore, we have, $Q_e(t) \geq Y_{e,t-1} - (x_{j,t} - y_{e,t}) + \frac{x_{j,t}}{x_{j,t}}(v - X_{j,t-1}) = v + Y_{e,t} - X_{e,t} \geq v$, again due to constraint (2). Hence, we have $Q_e(t_v) \geq v$ for all $e \in U_j$, which immediately yields $\tilde{C}_j(v) \leq t_v$. ◀

▶ **Lemma 6.** $\sum_{j \in J} w_j \int_{v=0}^{1} \bar{C}_j(v)dv = \text{COST}(\mathsf{LP}) - \sum_{j \in J} w_j/2$

Proof. It suffices to show that $\int_{v=0}^{1} \bar{C}_j(v)dv = \sum_{t \in [T]} t \cdot x_{j,t} - 1/2$, since summing this equation over all $j \in J$ multiplied by their weight w_j yields the lemma.

$$\int_{v=0}^{1} \bar{C}_j(v)dv = \sum_{t \in [T]} \int_{v=X_{j,t-1}}^{X_{j,t}} \bar{C}_j(v)dv = \sum_{t \in [T]: x_{j,t} \neq 0} \int_{v=X_{j,t-1}}^{X_{j,t}} \bar{C}_j(v)dv$$

$$= \sum_{t \in [T]: x_{j,t} \neq 0} \int_{v=X_{j,t-1}}^{X_{j,t}} \left(\frac{1}{x_{j,t}}(v - X_{j,t-1}) + (t-1) \right) dv$$

$$= \sum_{t \in [T]: x_{j,t} \neq 0} \left[\frac{1}{2} x_{j,t} + (t-1)x_{j,t} \right] = -\frac{1}{2} + \sum_{t \in [T]: x_{j,t} \neq 0} t \cdot x_{j,t},$$

where the last equality follows from constraint (1). ◀

4.2 Constructing the Stretched Schedule σ^λ

To construct σ^λ from σ we "stretch" the schedule σ by a factor of $1/\lambda$. More precisely, if an independent set S is scheduled in σ during an infinitesimal interval $(\tau_1, \tau_2]$, the same independent set is scheduled in σ^λ during $(\tau_1/\lambda, \tau_2/\lambda]$. In Lemma 7 we show that σ^λ completes job j by time $C_j^* = \lceil \frac{\bar{C}_j(\lambda)}{\lambda} \rceil$. In Lemma 8 we upper bound the expected cost of $\sum_j w_j C_j^*$ by twice $\text{COST}(\mathsf{LP})$.

▶ **Lemma 7.** *The schedule σ^λ completes every job j by time C_j^*.*

Proof. Lemma 5 shows that $\tilde{C}_j(v) \leq \bar{C}_j(v)$ for all $v \in (0, 1]$, meaning that every task in U_j is completed by v units by time $\bar{C}_j(v)$ in σ. Thus, in the stretched schedule σ^λ, every job j completes by time $\bar{C}_j(\lambda)/\lambda$, for any value of $\lambda \in (0, 1]$. ◀

▶ **Lemma 8.** $\mathbb{E}[\sum_{j \in J} w_j C_j^*] \leq 2\ \text{COST}(\mathsf{LP})$.

Proof. First note that

$$\sum_{j \in J} w_j \mathbb{E}[\bar{C}_j(\lambda)/\lambda] = \sum_{j \in J} w_j \int_{v=0}^{1} \bar{C}_j(v)/v \cdot (2v)dv = 2 \sum_{j \in J} w_j \int_{v=0}^{1} \bar{C}_j(v)dv \tag{6}$$

Thus, we have,

$$\mathbb{E}\Big[\sum_{j \in J} w_j C_j^*\Big] = \mathbb{E}\Big[\sum_{j \in J} w_j \lceil \frac{1}{\lambda} \bar{C}_j(\lambda) \rceil\Big] \leq \Big(\mathbb{E}\Big[\sum_j w_j \frac{1}{\lambda} \bar{C}_j(\lambda)\Big]\Big) + \sum_j w_j$$

$$= 2 \sum_j w_j \int_{v=0}^{1} \bar{C}_j(v)dv + \sum_j w_j \qquad \text{[Eqn. (6)]}$$

$$= 2\Big(\text{COST}(\mathsf{LP}) - \sum_j w_j/2\Big) + \sum_j w_j \qquad \text{[Lemma 6]}$$

$$= 2\ \text{COST}(\mathsf{LP})$$ ◀

4.3 Constructing a Discrete Integer Schedule

Let $y^*_{e,t}$ denote how much task e is processed during time interval $(t-1,t]$. In other words, task e appears in $y^*_{e,t}$ units of independents sets scheduled in σ^λ during the time interval. Then, $\{y^*_{e,t}\}_{e\in U, t\in[T]}$ satisfies the following:

1. For all $j \in J$ and $e \in U_j$, $\sum_{t\in[C^*_j]\setminus[r_j-1]} y^*_{e,t} = 1$; and .
2. For all $S \subseteq U$ and for all $t \in [T]$, $\sum_{e\in S} y^*_{e,t} \le \rho(S)$,

where the second holds true since $\{y^*_{e,t}\}_{e\in U}$ can be expressed as a convex combination of independent sets scheduled during time interval $(t-1,t]$, and therefore, lies in the matroid polytope. We now interpret $\{y^*_{e,t}\}$ as a fractional point in the intersection of two matroid polytopes. We create the following two matroids. The new universe U' is defined as $U' := \{(e,t) \mid t \in [T], j \in J, e \in U_j \text{ s.t. } r_j \le t \le C^*_j\}$. The first matroid M_1 is a partition matroid that forces to choose at most one element out of $\{(e,t)\}_t$, for each $e \in U$. Intuitively, this ensures that no task is scheduled more than once across times. The second matroid ensures that elements scheduled at each time t forms an independent set in $\mathcal{I}$. The following lemma formally defines the second matroid and shows that it is indeed a matroid.

▶ **Lemma 9.** *Define $\mathcal{I}_2 \subseteq 2^{U'}$ such that $S' \subseteq U'$ is in $\mathcal{I}_2$ if and only if for any $t \in [T]$, $\{e \mid (e,t) \in S'\} \in \mathcal{I}$. Then, $M_2 = (U', \mathcal{I}_2)$ is a matroid.*

Proof. Let $\mathcal{I}_2$ denote the family of independent sets of M_2. It is straightforward to see that $\mathcal{I}_2$ is downward closed. Thus, it suffices to show that for any $A', B' \in \mathcal{I}_2$ such that $|A'| < |B'|$, there exists $(e,t) \in B' \setminus A'$ such that $A' \cup \{(e,t)\} \in \mathcal{I}_2$. Let $U'_t := \{(e,t) \mid j \in J, e \in U_j \text{ s.t. } r_j \le t \le C^*_j\}$ denote the subset of U' restricted to time t. Consider any fixed $A', B' \in \mathcal{I}_2$ such that $|A'| < |B'|$. Then, consider any fixed time t^* such that $|A' \cap U'_{t^*}| < |B' \cap U'_{t^*}|$; such a time t^* must exist since $\{U'_t\}_t$ partitions U'. Then, for some $(e^*, t^*) \in (B' \cap U'_{t^*}) \setminus (A' \cap U'_{t^*})$, it must be the case that $\{e^*\} \cup \{e \mid (e,t^*) \in A' \cap U'_{t^*}\} \in \mathcal{I}$. This is because B' has more elements than A' that are paired up with the fixed time t^*, and therefore, the set of elements appearing in $A' \cap U'_{t^*}$ remains independent with some e^* added. Further, for any other time t, the elements appearing in the pairs of A' associated with t remain unchanged, and therefore, is in $\mathcal{I}$. ◀

Then, it is easy to see that $\{y^*_{e,t}\}$ is a point that lies in the intersection of the polymatroids that are defined by M_1 and M_2. Further, $\{y^*_{e,t}\}$ belongs to the base polymatroid of M_1; so we have $\sum_{(e,t)\in U'} y^*_{e,t} = |U|$. Since the matroid intersection polytope is well-known to be integral [26], meaning that every vertex is an integer point, a maximum independent set in the intersection of M_1 and M_2 must have $|U|$ elements. Further, we can find such a maximum independent set in polynomial time. To recap, we have found $S' \in U'$ that is a base of M_1 and is independent in M_2. This set S' immediately gives the desired integer schedule where $\{e \mid (e,t) \in S'\}$ is scheduled at each time t. Indeed, due to S' being a base of M_1, every task in U_j is scheduled exactly once during time interval $[r_j, C^*_j]$. Further, S' being independent in M_2 ensures that the set of tasks scheduled at each time forms an independent set in $\mathcal{I}$.

5 Derandomization

In this section, we discuss how to derandomize the choice of $\lambda \in (0,1]$, which was used to compute the deadlines for the jobs. This will complete the proof of Theorem 1. Let us first define *step* values. We say that $v \in (0,1]$ is a step value if $\sum_{s\le t} x_{j,s} = v$ for some $j \in J$ and integer $t \in [T]$ – in other words, exactly v fraction of some job j is completed by some integer time in the LP solution. Let V denote the set of all step values; $1 \in V$ by definition. Note that that $|V|$ is polynomially bounded in the input size, as the number of variables $x_{j,t}$ we consider in LP is at most $|J| \cdot |U|$.

Recall that in Lemma 8 we showed $\mathbb{E}[\sum_j w_j C_j^*] \leq 2\,\text{COST}(\mathsf{LP})$ when $C_j^* := \lceil \frac{1}{\lambda}\bar{C}_j(\lambda)\rceil$. This implies there exists a certain value of $\lambda \in (0,1]$ such that $\sum_j w_j C_j^* \leq 2\text{COST}(\mathsf{LP})$. For the purpose of derandomization, it suffices to find λ such that $\sum_j w_j \bar{C}_j(\lambda)/\lambda \leq 2\sum_j w_j \int_{v=0}^{1} \bar{C}_j(v)dv$; the equality is shown in equation (6) in expectation.

Towards this end, we aim to find $\lambda \in (0,1]$ that minimizes $\sum_j w_j \bar{C}_j(\lambda)/\lambda$. Suppose λ was set to a value $v \in (v_1, v_2]$, where v_1 and v_2 are two adjacent step values in V. Consider any fixed job j. Let $t \in [T]$ be such that $v \in (X_{j,t-1}, X_{j,t}]$. By definition of step values, we have $(v_1, v_2] \subseteq (X_{j,t-1}, X_{j,t}]$. Thus, we have $\bar{C}_j(v)/v = \frac{1}{x_{j,t}}(1 - \frac{X_{j,t-1}}{v}) + \frac{t-1}{v}$. This becomes a linear function in z over $[1/v_2, 1/v_1)$ if we set $z = 1/v$. Therefore, we get a piece-wise linear function $g(z)$ by summing over all jobs multiplied by their weight and considering all pairs of two adjacent step values in V. We set λ to the the inverse of z's value that achieves the global minimum, which can be found in polynomial time.

6 Arbitrary Processing Times

In this section we show how to extend Theorem 1 to allow tasks with arbitrary processing times with a loss of $(1+\epsilon)$ factor in the approximation ratio for any arbitrary constant $\epsilon > 0$. In this setting, each task e has an arbitrary integer size p_e and the task e completes when p_e independent sets including e are scheduled. As before, at each time we can schedule a set of tasks that is independent in the given matroid and a job completes when all its tasks complete.

6.1 Compact Linear Program

We first describe our new compact LP relaxation. Let $T := \sum_e p_e + \max_j r_j$, which is clearly an upper bound on the maximum time we need to consider. We define a set of times $\mathcal{T}$ that consists of polynomially many time steps. First, let $\mathcal{T}$ include every job's arrival time. Next, let $\mathcal{T}$ include all times appearing in $\{\lfloor (1+\epsilon)^i \rfloor\}_{0 \leq i \leq \lceil \log_{1+\epsilon} T \rceil + 1}$. In words, $\mathcal{T}$ includes exponentially increasing time steps by a factor of $(1+\epsilon)$ starting from 1 but includes no times greater than $(1+\epsilon)^2 T$. Let $t_1 = 1, t_2, \ldots, t_k, \ldots, t_{K+1}$ denote the (integer) times in $\mathcal{T}$ in increasing order. Let $I_i := [t_i, t_{i+1})$ where $i \in [K]$. The idea is to rewrite LP compactly as follows by replacing time-indexed variables with interval-indexed variables.

$$\min \sum_{j \in J} w_j \sum_{i \in [K]} (t_{i+1} - 1) \cdot x_{j,i}$$

$$\text{s.t.} \quad \forall e \in U, \qquad \sum_{i \in [K]} (t_{i+1} - t_i) y_{e,i} = p_e \tag{7}$$

$$\forall j \in J\ \forall e \in U_j\ \forall i \in [K], \qquad \sum_{i' \leq i} y_{e,i'}/p_e \geq \sum_{i' \leq i} x_{j,i'} \tag{8}$$

$$\forall S \subseteq U\ \forall i \in [K], \qquad \sum_{e \in S} y_{e,i} \leq \rho(S) \tag{9}$$

$$\forall j \in J\ \forall e \in U_j\ \forall i \in [K] \text{ s.t. } t_{i+1} \leq r_j, \qquad y_{e,i} = 0 \tag{10}$$

$$\mathbf{x}, \mathbf{y} \geq 0 \tag{11}$$

Here, variable $x_{j,i}$ can be viewed as the average fraction of job j that completes per unit time during I_i; so, when the job j completes during I_i for the first time, we have $\sum_{i' \leq i} x_{j,i'} = 1$. Likewise, $y_{e,i}$ has an analogous meaning for each task e but it denotes the

average *unit* of task e that is processed per unit time during I_i. Constraint (7) ensures that all tasks complete eventually. Constraint (9) ensures that the average vector representing how much each task is processed per unit time during I_t lies in the polymatroid. Constraint (10) enforces that no tasks in U_j are processed before j's arrival time; this is possible since $\mathcal{T}$ includes all jobs arrival times. Before explaining constraint (8), we explain the objective. If all intervals, $\{I_i\}$ were of unit length, the objective would be exactly the fractional total weighted completion time. However, to make the LP compact, when job j completes by $x_{j,i}$ fraction during interval I_i, we pretend that the fraction completes at the end of I_i, i.e., $t_{i+1} - 1$. Thus, we overestimate the fractional objective; but since times in I_i differ by at most $(1 + \epsilon)$ factor, our overestimate is by a factor of at most $(1 + \epsilon)$. Finally, we discuss constraint (8), which caps each job's (cumulative) processed fraction at the analogous quantity of each task of the job, which is measured as how much the task has been processed divided by its processing time. We also note that this compact LP admits the same separation oracle as the one for LP.

6.2 Rounding

As before, we seek to round the optimal LP solution. Recall that we first obtained $C_j^* := \lceil \frac{1}{\lambda} \bar{C}_j \rceil$ and found an integer schedule that completes every job j before C_j^*. We observe that the first procedure is no issue. This is because we can interpret the solution to our compact LP as a solution to LP. To see this, when a task e is processed by δ amount, pretend that there exist p_e different tasks of unit size and they are processed equally by δ/p_e amount. Thus, we can compute $\bar{C}_j(v)$ efficiently for any value of $v \in (0, 1]$. The derandomization can be done similarly.

6.3 Finding An Integer Schedule

It now remains to find an integer schedule meeting the discovered deadlines, $\{C_j^*\}_{j \in J}$. We use essentially the same idea of reducing the problem to finding an integer solution to the intersection of two matroids. However, this reduction requires some careful modifications to be implemented in polynomial time. Also, we will aim to complete every job j by $(1+O(\epsilon))C_j^*$ meeting the deadline slightly loosely.

The main idea is to use the fact that the continuous schedule σ^λ meeting the deadlines $\{C_j^*\}$ only changes polynomially many times. This is because the continuous schedule σ before the stretching is identical at all times during each of the intervals $(0, t_1 - 1], (t_1 - 1, t_2], \ldots, (t_{K-1} - 1, t_K]$ – these intervals are stretched into $(0, (t_1 - 1)/\lambda], ((t_1 - 1)/\lambda, t_2/\lambda], \ldots, ((t_{K-1} - 1)/\lambda, t_K/\lambda]$, respectively. We split the interval including the time $T' = |U|^2/\epsilon^2$ into two, the left one ending at $|U|^2/\epsilon^2$ and the right one starting at $|U|^2/\epsilon^2$. Here, assume that $1/\epsilon$ is an integer. We also add time $\bar{C}_j(\lambda)/\lambda$ for every $j \in J$ and split the intervals accordingly. To simplify the notation, we recycle the notations I_i. By reindexing the resulting intervals and merging some initial intervals, we have $I_0 := (0, T']$, I_1, I_2, ..., $I_{K'}$. We say that an interval is small if its starting time or ending time is not a power of $(1 + \epsilon)$ divided by λ; more precisely, $((t_{i-1} - 1)/\lambda, t_i/\lambda]$ is small if t_{i-1} or t_i is not a power of $(1 + \epsilon)$ divided by λ. Note that there are at most $4|J| + 4 \leq 8|J| \leq 8|U|$ small intervals since each job's arrival time and deadline together can create at most 4 small intervals; the extra four come from time 0, the final time, and T'.

For each interval I_i, let $Q_e(I_i)$ denote the amount of task e processed during I_i, which can be easily computed in polynomial time. For each interval, we will construct an integer schedule that schedules each task as much as the continuous schedule σ^λ does without using

too many time steps compared to the interval's length; more precisely, the integer schedule will process at least $\lceil Q_e(I_i) \rceil$ units of task e. We categorize the intervals into three groups. Depending on the category where each interval belongs, we construct an integer schedule differently or give a different upper bound on the length of the integer schedule. At the end, we will concatenate the constructed integer intervals in increasing order of times. In the following, $|I|$ denotes I's length.

The first interval, $I_0 = (0, T']$. Using the same idea we used for handling unit-sized tasks, we find an integer schedule that processes at least $\lfloor Q_e(I_i) \rfloor$, meeting all job deadlines no greater than T'. Note that I_0 has a polynomial length; thus, the desired integer schedule can be computed in polynomial time. Then, we can greedily schedule each task e per unit time such that $Q_e(I_i)$ is not an integer. Note that such a task e hasn't completed by time T', so the task (more precisely, the job to which the task belongs) has deadline at least T'. Therefore, we will be able to charge the extra delay of at most $|U|$ to the corresponding job's deadline directly.

I_i that is not small, for $i \geq 1$. We seek to construct an integer schedule of length $(1 + O(\epsilon))|I_i|$. Towards this end, we do the following. Suppose we divide the interval into $\lceil \frac{|I_i|}{|U|/\epsilon} \rceil$ subintervals of length $|U|/\epsilon$; there can be at most one subinterval of a smaller length and we will handle it later. Next, for each subinterval of length $|U|/\epsilon$, we try to schedule $\lceil \frac{|U|/\epsilon}{|I_i|} Q_e(I_i) \rceil$ units of each task e. Since the length is polynomial in $|U|$, we can find an integer schedule of length $|U|/\epsilon + 1$ that schedules $\lfloor \frac{|U|/\epsilon}{|I_i|} Q_e(I_i) \rfloor$ units of each task e. By scheduling one task per unit time, we can schedule $\lceil \frac{|U|/\epsilon}{|I_i|} Q_e(I_i) \rceil$ units of each task e for $|U|/\epsilon + 1 + |U| \leq (|U|/\epsilon) \cdot (1 + 2\epsilon)$ time steps. Here, our integer schedule's length is at most $(1 + 2\epsilon)$ times the subinterval's length, $|U|/\epsilon$. This integer schedule is repeated $\lfloor \frac{|I_i|}{|U|/\epsilon} \rfloor$ times. We now handle the smaller subinterval of length less than $|U|/\epsilon$. Using a similar argument, we can process more units of each task than the continuous schedule, using at most $|U|/\epsilon + 1 + |U| \leq 2|U|/\epsilon$ time steps. Here we use the fact that I_i has length significantly greater than $|U|$. To see this, suppose we had not added jobs arrival times, deadlines or T' in the process of creating the intervals. Then the intervals preceding I_i have exponentially decreasing lengths by a factor of $(1 + \epsilon)$. Using this observation, we can argue that I_i's length is at least $\epsilon/2$ times I_i's starting time. Since I_i's starting time is greater than T', we have that I_i's length is at least $(\epsilon/2) \cdot T' = (\epsilon/2) \cdot (|U|^2/\epsilon^2) = |U|^2/(2\epsilon)$. So, we can charge the number of time steps spent to handle the smaller subinterval, which is at most $2|U|/\epsilon$, to the length of I_i. From all these arguments, we can construct an integer schedule of length at most $(1 + 6\epsilon)|I_i|$.

I_i that is small, for $i \geq 1$. We seek to construct an integer schedule of length $(1+O(\epsilon))|I_i|+ 2|U|/\epsilon$. The whole idea is the same for the intervals that are not small. The only difference is that we cannot charge the extra time steps we spend to handle the smaller subinterval, which is at most $2|U|/\epsilon$, to the length of I_i. Thus, we just use the upper bound on the length of our integer schedule.

As mentioned before, we concatenate the integer schedules originating from $I_0, I_1, \ldots, I_K$ in this order to obtain the final schedule. It now remains to show that each job completes by time $(1 + O(\epsilon))C_j^*$. We already showed that our integer schedule completes every job j before its deadline C_j^* if it is smaller than T'. For any other job j, it must be the case that $\bar{C}_j(\lambda)/\lambda$ is greater than T'. Let I_i be the interval including $\bar{C}_j(\lambda)/\lambda$. Due to the way the intervals are

constructed, $\bar{C}_j(\lambda)/\lambda$ must be equal to I_i's finish time. Our goal is to show that we complete j not too late compared to I_i's finish time. That is, we want to show that the total length of the integer schedules originating from $I_0, I_1, \ldots, I_i$ is at most $(1+O(\epsilon))\sum_{i' \le i} |I_{i'}|$. Indeed, the total length is at most,

$$|I_0| + |U| + \sum_{i'=[i]:I_{i'} \text{ is small}} ((1+O(\epsilon))|I_i| + 2|U|/\epsilon) + \sum_{i'=[i]:I_{i'} \text{ is not small}} (1+O(\epsilon))|I_i|$$

$$\le \sum_{i'=0}^{i} (1+O(\epsilon))|I_{i'}| + |U| + (2|U|/\epsilon)\cdot(8|U|) \le \sum_{i'=0}^{i} (1+O(\epsilon))|I_{i'}| + O(\epsilon)|I_0|$$

Here, the first inequality follows from the fact that there are at most $8|U|$ small intervals, as argued above. The second inequality is immediate from $|I_0| = T' = |U|^2/\epsilon^2$. Therefore, we have shown that each job completes by time $(1+O(\epsilon))C^*_j$, which establishes that our final schedule's objective is at most $(1+O(\epsilon))$ times the compact LP's optimum. Since we showed the compact LP lower bounds the optimum times $(1+\epsilon)$, we obtain a $2(1+\epsilon)$-approximate schedule for arbitrary $\epsilon > 0$ by scaling ϵ appropriately.

References

1. Saksham Agarwal, Shijin Rajakrishnan, Akshay Narayan, Rachit Agarwal, David Shmoys, and Amin Vahdat. Sincronia: Near-optimal Network Design for Coflows. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, pages 16–29. ACM, 2018.
2. Saba Ahmadi, Samir Khuller, Manish Purohit, and Sheng Yang. On Scheduling Coflows. In *IPCO*, pages 13–24. Springer, 2017.
3. Nikhil Bansal and Subhash Khot. Inapproximability of hypergraph vertex cover and applications to scheduling problems. In *ICALP*, pages 250–261. Springer, 2010.
4. Zhi-Long Chen and Nicholas G Hall. Supply chain scheduling: Conflict and cooperation in assembly systems. *Operations Research*, 55(6):1072–1089, 2007.
5. Mosharaf Chowdhury and Ion Stoica. Coflow: A networking abstraction for cluster applications. In *ACM Workshop on Hot Topics in Networks*, pages 31–36. ACM, 2012.
6. Mosharaf Chowdhury and Ion Stoica. Efficient coflow scheduling without prior knowledge. In *SIGCOMM*, pages 393–406. ACM, 2015.
7. Mosharaf Chowdhury, Yuan Zhong, and Ion Stoica. Efficient Coflow Scheduling with Varys. In *SIGCOMM*, SIGCOMM '14, pages 443–454, New York, NY, USA, 2014. ACM.
8. William H Cunningham. Testing membership in matroid polyhedra. *Journal of Combinatorial Theory, Series B*, 36(2):161–188, 1984.
9. Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.
10. Naveen Garg, Amit Kumar, and Vinayaka Pandit. Order scheduling models: Hardness and algorithms. In *FSTTCS*, pages 96–107. Springer, 2007.
11. Sungjin Im and Manish Purohit. A Tight Approximation for Co-flow Scheduling for Minimizing Total Weighted Completion Time. *CoRR*, abs/1707.04331, 2017. `arXiv:1707.04331`.
12. Sungjin Im, Maxim Sviridenko, and Ruben Van Der Zwaan. Preemptive and non-preemptive generalized min sum set cover. *Mathematical Programming*, 145(1-2):377–401, 2014.
13. Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *JACM*, 48(4):761–777, 2001.
14. Hamidreza Jahanjou, Erez Kantor, and Rajmohan Rajaraman. Asymptotically Optimal Approximation Algorithms for Coflow Scheduling. In *SPAA*, pages 45–54. ACM, 2017.
15. Samir Khuller, Jingling Li, Pascal Sturmfels, Kevin Sun, and Prayaag Venkat. Select and permute: An improved online framework for scheduling to minimize weighted completion time. In *LATIN*, pages 669–682. Springer, 2018.

16 Samir Khuller and Manish Purohit. Brief announcement: Improved approximation algorithms for scheduling co-flows. In *SPAA*, pages 239–240. ACM, 2016.

17 Joseph Y-T Leung, Haibing Li, and Michael Pinedo. Scheduling orders for multiple product types to minimize total weighted completion time. *Discrete Applied Mathematics*, 155(8):945–970, 2007.

18 S. Luo, H. Yu, Y. Zhao, S. Wang, S. Yu, and L. Li. Towards Practical and Near-optimal Coflow Scheduling for Data Center Networks. *IEEE Transactions on Parallel and Distributed Systems*, 27(11):3366–3380, 2016.

19 Monaldo Mastrolilli, Maurice Queyranne, Andreas S Schulz, Ola Svensson, and Nelson A Uhan. Minimizing the sum of weighted completion times in a concurrent open shop. *Operations Research Letters*, 38(5):390–395, 2010.

20 Zhen Qiu, Cliff Stein, and Yuan Zhong. Minimizing the Total Weighted Completion Time of Coflows in Datacenter Networks. In *Symposium on Parallel Algorithms and Architectures*, pages 294–303. ACM, 2015.

21 Maurice Queyranne and Andreas S Schulz. Approximation bounds for a general class of precedence constrained parallel machine scheduling problems. *SIAM Journal on Computing*, 35(5):1241–1253, 2006.

22 Maurice Queyranne and Maxim Sviridenko. A (2+ ε)-approximation algorithm for the generalized preemptive open shop problem with minsum objective. *Journal of Algorithms*, 45(2):202–212, 2002.

23 Sushant Sachdeva and Rishi Saket. Optimal inapproximability for scheduling problems via structural hardness for hypergraph vertex cover. In *IEEE Conference on Computational Complexity*, pages 219–229. IEEE, 2013.

24 Alexander Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000.

25 Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.

26 Alexander Schrijver. Matroid Intersection. In *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24, chapter 41. Springer Science & Business Media, 2003.

27 Andreas S Schulz and Martin Skutella. Random-based scheduling new approximations and LP lower bounds. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 119–133. Springer, 1997.

28 Andreas S Schulz and Martin Skutella. Scheduling unrelated machines by randomized rounding. *SIAM Journal on Discrete Mathematics*, 15(4):450–469, 2002.

29 Mehrnoosh Shafiee and Javad Ghaderi. An Improved Bound for Minimizing the Total Weighted Completion Time of Coflows in Datacenters. *arXiv preprint*, 2017. `arXiv:1704.08357`.

30 Guoqing Wang and TC Edwin Cheng. Customer order scheduling to minimize total weighted completion time. *Omega*, 35(5):623–626, 2007.

31 Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.

32 Yangming Zhao, Kai Chen, Wei Bai, Minlan Yu, Chen Tian, Yanhui Geng, Yiming Zhang, Dan Li, and Sheng Wang. RAPIER: Integrating routing and scheduling for coflow-aware data center networks. In *INFOCOM*, pages 424–432. IEEE, 2015.

Multi-Round Cooperative Search Games with Multiple Players

Amos Korman
Université de Paris, IRIF, CNRS, F-75013 Paris, France
amos.korman@irif.fr

Yoav Rodeh
Ort Braude College, Karmiel, Israel
yoav.rodeh@braude.ac.il

Abstract

Assume that a treasure is placed in one of M boxes according to a known distribution and that k searchers are searching for it in parallel during T rounds. We study the question of how to incentivize selfish players so that group performance would be maximized. Here, this is measured by the *success probability*, namely, the probability that at least one player finds the treasure. We focus on *congestion policies* $C(\ell)$ that specify the reward that a player receives if it is one of ℓ players that (simultaneously) find the treasure for the first time. Our main technical contribution is proving that the *exclusive policy*, in which $C(1) = 1$ and $C(\ell) = 0$ for $\ell > 1$, yields a *price of anarchy* of $(1 - (1 - 1/k)^k)^{-1}$, and that this is the best possible price among all symmetric reward mechanisms. For this policy we also have an explicit description of a symmetric equilibrium, which is in some sense unique, and moreover enjoys the best success probability among all symmetric profiles. For general congestion policies, we show how to polynomially find, for any $\theta > 0$, a symmetric multiplicative $(1 + \theta)(1 + C(k))$-equilibrium.

Together with an appropriate reward policy, a central entity can suggest players to play a particular profile at equilibrium. As our main conceptual contribution, we advocate the use of symmetric equilibria for such purposes. Besides being fair, we argue that symmetric equilibria can also become highly robust to crashes of players. Indeed, in many cases, despite the fact that some small fraction of players crash (or refuse to participate), symmetric equilibria remain efficient in terms of their group performances and, at the same time, serve as approximate equilibria. We show that this principle holds for a class of games, which we call *monotonously scalable* games. This applies in particular to our search game, assuming the natural *sharing policy*, in which $C(\ell) = 1/\ell$. For the exclusive policy, this general result does not hold, but we show that the symmetric equilibrium is nevertheless robust under mild assumptions.

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

Keywords and phrases Algorithmic Mechanism Design, Parallel Algorithms, Collaborative Search, Fault-Tolerance, Price of Anarchy, Price of Stability, Symmetric Equilibria

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.146

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Related Version A full version of the paper is available at `https://arxiv.org/abs/1811.01270`.

Funding This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No 648032).

1 Introduction

Searching in groups is ubiquitous in multiple contexts, including in the biological world, in human populations as well as on the internet [8, 11, 16]. In many cases there is some prior on the distribution of the searched target. Moreover, when the space is large, each searcher typically needs to inspect multiple possibilities, which in some circumstances can only be done sequentially. This paper introduces a game theoretic perspective to such multi-round treasure hunt searches, generalizing a basic collaborative Bayesian framework previously introduced in [8].

Consider the case that a treasure is placed in one of M boxes according to a known distribution f and that k searchers are searching for it in parallel during T rounds, each specifying a box to visit in each round. Assume w.l.o.g. that the boxes are ordered such that lower index boxes have higher probability to host the treasure, i.e., $f(x) \geq f(x+1)$. We evaluate the group performance by the *success probability*, that is, the probability that the treasure is found by at least one searcher.

If coordination is allowed, letting searcher i visit box $(t-1)k+i$ at time t will maximize success probability. However, as simple as this algorithm is, it is very sensitive to faults of all sorts. For example, if an adversary that knows where the treasure is can crash a searcher before the search starts (i.e., prevent it from searching), then it can reduce the search probability to zero.

The authors of [8] suggested the use of *identical non-coordinating* algorithms. In such scenarios all processors act independently, using no communication or coordination, executing the same probabilistic algorithm, differing only by the results of their coin flips. As argued in [8], in addition to their economic use of communication, identical non-coordinating algorithms enjoy inherent robustness to different kind of faults. For example, assume that there are $k+k'$ searchers, and that an adversary can fail up to k' searchers. Letting all searchers run the best non-coordinating algorithm for k searchers guarantees that regardless of which $\ell \leq k'$ searchers fail, the overall search efficiency is at least as good as the non-coordinating one for k players. Of course, since k' players might fail, any solution can only hope to achieve the best performance of k players. As it applies to the group performance we term this property as *group robustness*. Among the main results in [8] is identifying a non-coordinating algorithm, denoted $A^\star$, whose expected running time is minimal among non-coordinating algorithms. Moreover, for every given T, if this algorithm runs for T rounds, it also maximizes the success probability.

The current paper studies the game theoretic version of this multi-round search problem[1]. The setting of [8] assumes that the searchers adhere fully to the instructions of a central entity. In contrast, in a game theoretical context, searchers are self-interested and one needs to incentivize them to behave as desired, e.g., by awarding those players that find the treasure first. For many real world contexts, the competitive setting is in fact the more realistic one to assume. Applications range from crowd sourcing [26], multi-agent searching on the internet [25], grant proposals [18], to even contexts of animals [17]. See the full version for more details.

[1] We concentrate on the normal form version in which players do not receive any feedback during the search (except when the treasure is found in which case the game ends). In particular, we assume that players cannot communicate with each other.

In the competitive setting, choosing a good *rewarding policy* becomes a problem in algorithmic mechanism design [24]. Typically, a reward policy is evaluated by its *price of anarchy (PoA)*, namely, the ratio between the performances of the best collaborative algorithm and the worst equilibrium [19]. Aiming to both accelerate the convergence process to an equilibrium and obtain a preferable one, the announcement of the reward policy can be accompanied by a proposition for players to play particular strategies that form a profile at equilibrium.

This paper highlights the benefits of suggesting (non-coordinating) *symmetric equilibria* in such scenarios, that is, to suggest the same non-coordinating strategy to be used by all players, such that the resulting profile is at equilibrium. This is of course relevant assuming that the *price of symmetric stability (PoSS)*, namely, the ratio between the performances of the best collaborative algorithm and the best symmetric equilibrium, is low. Besides the obvious reasons of fairness and simplicity, from the perspective of a central entity who is interested in the overall success probability, we obtain the group robustness property mentioned above, by suggesting that the $k + k'$ players play according to the strategy that is a symmetric equilibrium for k players. Obviously, this group robustness is valid only provided that the players indeed play according to the suggested strategy. However, the suggested strategy is guaranteed to be an equilibrium only for k players, while in fact, the adversary may keep some of the extra k' players alive. Interestingly, however, in many cases, a symmetric equilibrium for k players also serves as an approximate equilibrium for $k + k'$ players, as long as $k' \ll k$. As we show, this *equilibrium robustness* property is rather general, holding for a class of games, that we call *monotonously scalable* games.

1.1 The Collaborative Search Game

A treasure is placed in one of M boxes according to a known distribution f and k players are searching for it in parallel during T rounds. Assume w.l.o.g. that $f(x) > 0$ for every x and that $f(x) \geq f(x + 1)$.

Strategies. An *execution* of T rounds is a sequence of box visitations $\sigma = x(1), x(2), \ldots x(T)$, one for each round $i \leq T$. We assume that a player visiting a box has no information on whether other players have already visited that box or are currently visiting it. Hence, a *strategy* of a player is a probability distribution over the space of executions of T rounds. Note that the probability of visiting a box x in a certain round may depend on the boxes visited by the player until this round, but not on the actions of other players. A strategy is *non-redundant* if at any given round it always checks a box it didn't check before (as long as there are such boxes).

A *profile* is a collection of k strategies, one for each player. Special attention will be devoted to *symmetric profiles*. In such profiles all players play the same strategy (note that their actual executions may be different, due to different probabilistic choices).

Probability Matrix. While slightly abusing notation, we shall associate each strategy A with its *probability matrix*, $A : \{1, \ldots, M\} \times \{1, \ldots, T\} \to [0, 1]$, where $A(x, t)$ is the probability that strategy A visits x for the first time at round t. We also denote $\widetilde{A}(x, t)$ as the probability that A does not visit x by, and including, time t. That is, $\widetilde{A}(x, t) = 1 - \sum_{s \leq t} A(x, t)$ and $\widetilde{A}(x, 0) = 1$. For convenience we denote by $\delta_{x,t}$ the matrix of all zeros except 1 at x, t. Its dimensions will be clear from context.

Group Performance. A profile is evaluated by its *success probability*, i.e., the probability that at least one player finds the treasure by time T. Formally, let $\mathbb{P}$ be a profile. Then,

$$\texttt{success}(\mathbb{P}) = \sum_x f(x) \left(1 - \prod_{A \in \mathbb{P}} \widetilde{A}(x,T)\right).$$

The expected running time in the symmetric case, which is $\sum_x f(x) \sum_t \widetilde{A}(x,t)^k$, was studied in [8]. That paper identified a strategy, denoted $A^\star$, that minimizes this quantity. In fact, it does so by minimizing the term $\sum_x f(x)\widetilde{A}(x,t)^k$ for each t separately. Note that minimizing the case $t = T$ is exactly the same as maximizing the success probability. Thus, restricted to the case where all searchers use the same strategy, $A^\star$ simultaneously optimizes the success probability as well as optimizes the expected running time. For completeness, a description of $A^\star$ is provided below.

Algorithm $A^\star$. We note that in [8] the matrix of $A^\star$ is given, and then an algorithm is explicitly described that has its matrix (Section 4.3 in [8]). We describe the matrix only, as its details are necessary for this paper.

Denote $q(x) = f(x)^{-1/(k-1)}$. For each t, $\widetilde{A^\star}(x,t) = \min(1, \alpha(t)q(x))$, where $\alpha(t) \geq 0$ is such that $\sum_x \widetilde{A^\star}(x,t) = M - t$. Of course, once $\widetilde{A^\star}$ is known, then so is $A^\star$: $A^\star(x,t) = \widetilde{A^\star}(x,t-1) - \widetilde{A^\star}(x,t)$.

Congestion Policies. A natural way to incentivize players is by rewarding those players that find the treasure before others. A *congestion policy* $C(\ell)$ is a function specifying the reward that a player receives if it is one of ℓ players that (simultaneously) find the treasure for the first time. We assume that $C(1) = 1$, and that C is non-negative and non-increasing. Due to the fact that the policy $C \equiv 1$ is rather degenerate, we henceforth assume that $C \not\equiv 1$. We shall give special attention to the following policies.

- The *sharing policy* is defined by $C_{\texttt{share}}(\ell) = 1/\ell$, namely, the treasure is shared equally among all those who find it first.
- The *exclusive policy* is defined by $C_{\texttt{ex}}(1) = 1$, and $C_{\texttt{ex}}(\ell) = 0$ for $\ell > 1$, namely, the treasure is given to the first one that finds it exclusively; if more than one discover it, they get nothing[2].

A *configuration* is a triplet (C, f, T), where C is a congestion policy, T is a positive integer, and f is a positive non-increasing probability distribution on M boxes.

Values, Utilities and Equilibria. Let (C, f, T) be a configuration. The *value* of box x at round t when playing against a profile $\mathbb{P}$ is the expected gain from visiting x at round t. Formally,

$$\begin{aligned} v_{\mathbb{P}}(x,t) &= f(x) \sum_{\ell=0}^{k-1} C(\ell+1) \Pr\left(\begin{array}{l} x \text{ was not visited before time } t, \text{ and} \\ \text{at time } t \text{ is visited by } \ell \text{ players of } \mathbb{P} \end{array}\right) \\ &= f(x) \sum_{\ell=0}^{k-1} C(\ell+1) \sum_{\substack{I \subseteq \mathbb{P} \\ |I| = \ell}} \prod_{A \in I} A(x,t) \prod_{A \notin I} \widetilde{A}(x,t). \end{aligned}$$

[2] In the one round game, the exclusive policy yields a utility for a player that equals its marginal contribution to the social welfare, i.e., the success probability [27]. However, this is not the case in the multi-round game.

The *utility of A in round t* and the *utility of A* are defined as:

$$U_{\mathbb{P}}(A,t) := \sum_x A(x,t) \cdot v_{\mathbb{P}^{-A}}(x,t), \qquad U_{\mathbb{P}}(A) := \sum_t U_{\mathbb{P}}(A,t), \tag{1}$$

where $\mathbb{P}^{-A}$ is the set of players of $\mathbb{P}$ excluding A. Here are some specific cases we are interested in:

- For symmetric profiles, $v_A(x,t)$ denotes the value when playing against $k-1$ players playing A. Then $v_A(x,t) = f(x)\sum_{\ell=0}^{k-1} C(\ell+1)\binom{k-1}{\ell} A(x,t)^{\ell}\widetilde{A}(x,t)^{k-\ell-1}$.
- For the exclusive policy, $v_{\mathbb{P}}(x,t) = f(x)\prod_{A\in\mathbb{P}}\widetilde{A}(x,t)$.
- For the exclusive policy in symmetric profiles, $v_A(x,t) = f(x)\widetilde{A}(x,t)^{k-1}$.

A profile $\mathbb{P}$ is a *Nash equilibrium* under a configuration, if for any $A \in \mathbb{P}$ and any other strategy B, $U_{\mathbb{P}^{-A}}(A) \geq U_{\mathbb{P}^{-A}}(B)$. Similarly, a strategy A is called a *symmetric equilibrium* if the profile A^k consisting of all k players playing according to A is an equilibrium. We also use the notion of approximate equilibrium. For $\epsilon > 0$, we say a profile $\mathbb{P}$ is a $(1+\epsilon)$-equilibrium if for every $A \in \mathbb{P}$ and for every strategy B, $U_{\mathbb{P}^{-A}}(B) \leq (1+\epsilon)U_{\mathbb{P}^{-A}}(A)$.

A Game of Doubly-Substochastic Matrices. Both the expressions for the success probability and utility solely depend on the values of the probability matrices associated with the strategies in question. Hence we view all strategies sharing the same matrix as *equivalent*. Note that a matrix does not necessarily correspond to a unique strategy, as illustrated by the following equivalent strategies, for which $A(x,t) = B(x,t) = 1/M$ for every $t \leq M$ and 0 thereafter:

- Strategy A chooses uniformly at every round one of the boxes it didn't choose yet.
- Strategy B chooses once $x \in \{0,\ldots,M-1\}$. Then, at round t it visits box $(x+t \mod M)+1$.

Matrices are much simpler to handle than strategies, and so we would rather think of our game as a game of probability matrices than a game of strategies. For this we need to characterize which matrices are indeed probability matrices of strategies. Clearly, a probability matrix is non-negative. Also, by their definition, each row and each column sums to at most 1. Such a matrix is called *doubly-substochastic*. In the extended version we prove the converse, i.e., that every doubly-substochastic matrix is a probability matrix of some strategy. Furthermore, this strategy is implementable as a polynomial algorithm. We will therefore view our game as a game of doubly-substochastic matrices.

Greediness. Informally, a strategy is greedy at a round if its utility in this round is the maximum possible in this round. Formally, given a profile $\mathbb{P}$ and some strategy A, we say that A is *greedy* w.r.t. $\mathbb{P}$ at time t if for any strategy B such that for every x and $s < t$, $B(x,s) = A(x,s)$, we have $U_{\mathbb{P}}(A,t) \geq U_{\mathbb{P}}(B,t)$. We say A is greedy w.r.t. $\mathbb{P}$ if it is greedy w.r.t. $\mathbb{P}$ for each $t \leq T$. A strategy A is called *self-greedy* (or *sgreedy* for short) if it is greedy w.r.t. the profile with $k-1$ players playing A.

Evaluating Policies. Let (C,f,T) be a configuration. Denote by $\text{Nash}(C,f,T)$ the set of equilibria for this configuration, and by $\text{S-Nash}(C,f,T)$ the subset of symmetric ones. Let $\mathcal{P}(T)$ be the set of all profiles of T-round strategies. We are interested in the following measures.

- The *Price of Anarchy (PoA)* is $\texttt{PoA}(C,f,T) := \frac{\max_{\mathbb{P}\in\mathcal{P}(T)} \texttt{success}(\mathbb{P})}{\min_{\mathbb{P}\in\text{Nash}(C,f,T)} \texttt{success}(\mathbb{P})}$.
- The *Price of Symmetric Stability (PoSS)* is $\texttt{PoSS}(C,f,T) := \frac{\max_{\mathbb{P}\in\mathcal{P}(T)} \texttt{success}(\mathbb{P})}{\max_{A\in\text{S-Nash}(C,f,T)} \texttt{success}(A^k)}$.
- The *Price of Symmetric Anarchy (PoSA)* is $\texttt{PoSA}(C,f,T) = \frac{\max_{\mathbb{P}\in\mathcal{P}(T)} \texttt{success}(\mathbb{P})}{\min_{A\in\text{S-Nash}(C,f,T)} \texttt{success}(A^k)}$.

On the Difficulty of the Multi-Round Game. The setting of multi-rounds poses several challenges that do not exist in the single round game. An important one is the fact that, in contrast to the single round game, the multi-round game is not a potential game. Indeed, being a potential game has several implications, and a significant one is that such a game has always a pure equilibrium. However, we show that multi-round games do not always have pure equilibria and hence they are not potential games. Another important difference is that for policies that incur high levels of competition (such as the exclusive policy), profiles that maximize the success probability are at equilibrium in the single round case, whereas they are not in the multi-round game. See the full version of the paper for more details.

1.2 Our Results

Equilibrium Robustness. We first provide a simple, yet general, robustness result, that holds for symmetric (approximate) equilibria in a family of games, termed *monotonously scalable*. Informally, these are games in which the sum of utilities of players can only increase when more players are added, yet for each player, its individual utility can only decrease. Our search game with the sharing policy is one such example.

▶ **Theorem 1.** *Consider a symmetric monotonously scalable game. If A is a symmetric $(1+\epsilon)$-equilibrium for k players, then it is an $(1+\epsilon)(1+\ell/k)$-equilibrium when played by $k+\ell$ players.*

Theorem 1 is applicable in fault tolerant contexts. Consider a monotonously scalable game with $k+k'$ players out of which at most k' may fail. Let A_k be a symmetric (approximate) equilibrium designed for k players and assume that its social utility is high compared to the optimal profile with k players. The theorem implies that if players play A_k, then regardless of which $\ell \leq k'$ players fail (or decline to participate), the incentive to switch strategy would be very small, as long as $k' \ll k$. Moreover, due to symmetry, if the social utility of the game is monotone, then the social utility of A_k when played with k players is guaranteed when playing with more. Thus, in such cases we obtain both group robustness and equilibrium robustness.

General Congestion Policies. Coming back to our search game, we consider general policies, focus on symmetric profiles, and specifically, on the properties of sgreedy strategies.

▶ **Theorem 2.** *For every policy C there exists a non-redundant sgreedy strategy. Moreover, all such strategies are equivalent and are symmetric $(1+C(k))$-equilibria.*

When $C(k)=0$ this shows that a non-redundant sgreedy strategy is actually a symmetric equilibrium. We next claim that this is the only symmetric equilibrium (up to equivalence).

▷ Claim 3. For any policy such that $C(k)=0$, all symmetric equilibria are equivalent.

Theorem 2 is non-constructive because it requires calculating the inverse of non-trivial functions. Therefore, we resort to an approximate solution.

▶ **Theorem 4.** *Given $\theta > 0$, there exists an algorithm that takes as input a configuration, and produces a symmetric $(1+C(k))(1+\theta)$-equilibrium. The algorithm runs in polynomial time in T, k, M, $\log(1/\theta)$, $\log(1/(1-C(k)))$, and $\log(1/f(M))$.*

The Exclusive Policy. Recall that the exclusive policy is defined by $C_{\mathtt{ex}}(1) = 1$ and $C_{\mathtt{ex}}(\ell) = 0$ for every $\ell > 1$. We show that $A^\star$ is a non-redundant and sgreedy strategy in the exclusive policy. Hence, Theorem 2 implies the following.

▶ **Theorem 5.** *Under the exclusive policy, Strategy $A^\star$ of [8] is a symmetric equilibrium.*

Claim 3 together with the fact (established in [8]) that $A^\star$ has the highest success probability among symmetric profiles, implies that both the PoSS and the PoSA of $C_{\mathtt{ex}}$ are optimal (and equal) on any f and T when compared to any other policy. The next theorem considers general equilibria.

▶ **Theorem 6.** *Consider the exclusive policy. For any profile $\mathbb{P}_{\mathtt{nash}}$ at equilibrium and any symmetric profile A, $\mathtt{success}(\mathbb{P}_{\mathtt{nash}}) \geq \mathtt{success}(A)$.*

Observe that, as $A^\star$ is a symmetric equilibrium, Theorem 6 provides an alternative proof for the optimality of $A^\star$ (established in [8]). Interestingly, this alternative proof is based on game theoretic considerations, which is a very rare approach in optimality proofs.

Combining Theorems 5 and 6, we obtain:

▶ **Corollary 7.** *For any f and T, $\mathtt{PoA}(C_{\mathtt{ex}}, f, T) = \mathtt{PoSA}(C_{\mathtt{ex}}, f, T)$. Moreover, for any policy C, $\mathtt{PoA}(C_{\mathtt{ex}}, f, T) \leq \mathtt{PoA}(C, f, T)$.*

At first glance the effectiveness of C_{ex} might not seem so surprising. Indeed, it seems natural that high levels of competition would incentivize players to disperse. However, it is important to note that $C_{\mathtt{ex}}$ is not extreme in this sense, as one may allow congestion policies to also have negative values upon collisions. Moreover, one could potentially define more complex kinds of policies, e.g., policies that depend on time, and reward early finds more. However, the fact that $A^\star$ is optimal among all symmetric profiles combined with the fact that any symmetric policy has a symmetric equilibrium [23] implies that no symmetric reward mechanism can improve either the PoSS, the PoSA, or the PoA of the exclusive policy.

We proceed to show a tight upper bound on the PoA of $C_{\mathtt{ex}}$. Note that as k goes to infinity the bound converges to $e/(e-1) \approx 1.582$.

▶ **Theorem 8.** *For every T, $\sup_f \mathtt{PoA}(C_{\mathtt{ex}}, f, T) = (1-(1-1/k)^k)^{-1}$.*

Concluding the results on the exclusive policy, we study the robustness of $A^\star$ in the full version of the paper. Let $A^\star_k$ denote algorithm $A^\star$ when set to work for k players. Unfortunately, for any ϵ, there are cases where $A^\star_k$ is not a $(1+\epsilon)$-equilibrium even when played by $k+1$ players. However, as indicated below, $A^\star$ is robust to failures under reasonable assumptions regarding the distribution f.

▶ **Theorem 9.** *If $\frac{f(1)}{f(M)} \leq (1+\epsilon)^{\frac{k-1}{k'}}$, then $A^\star_k$ is a $(1+\epsilon)$-equilibrium when played by $k+k'$ players.*

The Sharing Policy. Another important policy to consider is the sharing policy. This policy naturally arises in some circumstances, and may be considered as a less harsh alternative to the exclusive one. Although not optimal, it follows from Vetta [31] that its PoA is at most 2 (see the full version). Furthermore, as this policy yields a monotonously scalable game, a symmetric equilibrium under it is also robust. Therefore, the existence of a symmetric profile which is both robust and has a reasonable success probability is guaranteed.

Unfortunately, we did not manage to find a polynomial algorithm that generates a symmetric equilibrium for this policy. However, Theorem 4 gives a symmetric $(1+\theta)(1+1/k)$-equilibrium in polynomial time for any $\theta > 0$. This strategy is also robust thanks to Theorem 1.

Moreover, the proof in [31] regarding the PoA can be extended to hold for approximate equilibria. In particular, if $\mathbb{P}$ is some $(1+\epsilon)$-equilibrium in the sharing policy, then for every f and T, $\texttt{success}(\mathbb{P}) \geq \frac{1}{2+\epsilon} \max_{\mathbb{P}' \in \mathcal{P}(T)} \texttt{success}(\mathbb{P}')$. See the full version of the paper for the proof.

1.3 Related Works

Fault tolerance has been a major topic in distributed computing for several decades, and in recent years more attention has been given to these concepts in game theory [13, 14]. For example, Gradwohl and Reingold studied conditions under which games are robust to faults, showing that equilibria in anonymous games are fault tolerant if they are "mixed enough" [12].

Restricted to a single round the search problem becomes a coverage problem, which has been investigated in several papers. For example, Collet and Korman studied in [30] (one-round) coverage while restricting attention to symmetric profiles only. The main result therein is that the exclusive policy yields the best coverage among symmetric profiles. Gairing [10] also considered the single round setting, but studied the optimal PoA of a more general family of games called *covering games* (see also [27, 28]). Motivated by policies for research grants, Kleinberg and Oren [18] considered a one-round model similar to that in [30]. Their focus however was on pure strategies only. The aforementioned papers give a good understanding of coverage games in the single round setting. As mentioned, however, the multi-round setting studied here is substantially more complex than the single-round setting.

The area of "incentivizing exploration" also studies the tradeoff between exploration, exploitation and incentives [21, 9, 26, 20]. This area often focuses on different variants of the Multi-Armed Bandit problem. The settings of selfish routing, job scheduling, and congestion games [22, 29] all bear similarities to the search game studied here, however, the social welfare measurements of success probability or running time are very different from the measures studied in these frameworks, such as makespan or latency [1, 3, 24, 2].

2 Robustness in Symmetric Monotonously Scalable Games

Consider a symmetric game where the number of players is not fixed. Let $U_{\mathbb{P}}(A)$ denote the utility that a player playing A gets if the other players play according to $\mathbb{P}$ and let $\sigma(\mathbb{P}) = \sum_{A \in \mathbb{P}} U_{\mathbb{P}-A}(A)$. We say that such a game is *monotonously scalable* if:

1. Adding more players can only increase the sum of utilities, i.e., if $\mathbb{P} \subseteq \mathbb{P}'$ then $\sigma(\mathbb{P}) \leq \sigma(\mathbb{P}')$.
2. Adding more players can only decrease the individual utilities, i.e., if $\mathbb{P} \subseteq \mathbb{P}'$ then for all $A \in \mathbb{P}$, $U_{\mathbb{P}-A}(A) \geq U_{\mathbb{P}'-A}(A)$.

▶ **Theorem 1.** *Consider a symmetric monotonously scalable game. If A is a symmetric $(1+\epsilon)$-equilibrium for k players, then it is an $(1+\epsilon)(1+\ell/k)$-equilibrium when played by $k+\ell$ players.*

Proof. On the one hand by symmetry,

$$U_{A^{k+\ell-1}}(A) = \frac{\sigma(A^{k+\ell})}{k+\ell} \geq \frac{\sigma(A^k)}{k+\ell},$$

where the last step is because σ is non-decreasing. On the other hand, if B is some other strategy,

$$U_{A^{k+\ell-1}}(B) \leq U_{A^{k-1}}(B) \leq (1+\epsilon) U_{A^{k-1}}(A) = (1+\epsilon)\frac{\sigma(A^k)}{k}.$$

The first inequality is because U is non-increasing, and the second is because A is a $(1+\epsilon)$-equilibrium for k players. Therefore, what a player can gain by switching from A to B is at most a multiplicative factor of $(1+\epsilon)(k+\ell)/k = (1+\epsilon)(1+\ell/k)$. ◀

An example of such a game is our setting with the sharing policy. Note however, that our game with the exclusive policy does not satisfy the first property, as adding more players can actually deteriorate the sum of utilities. Another example is a generalization known as *covering games* [10]. This sort of game is the same as our game in a single-round version, except that each player chooses not necessarily one element, but a set of elements, from a prescribed set of sets. Again, to be a monotonously scalable game, the congestion policy should be the sharing policy. Note that one may consider a multi-round version of these games, which will be monotonously scalable as well.

3 General Policies

The proofs of this section appear in the full version of the paper.

3.1 Non-Redundancy and Monotonicity

A doubly-substochastic matrix A is called *non-redundant at time* t if $\sum_x A(x,t) = 1$. It is *non-redundant* if it is non-redundant for every $t \leq M$. In the algorithmic view, as $\sum_x A(x,t)$ is the probability that a new box is opened at time t, then a strategy's matrix is non-redundant iff it never checks a box twice, unless it already checked all boxes.

▶ **Lemma 10.** *If a profile $\mathbb{P}$ is at equilibrium and* $\texttt{success}(\mathbb{P}) < 1$ *then every player is non-redundant.*

We will later see that in the symmetric case the condition in the lemma is not needed. However, the following example shows it is necessary in general. Let $M = k$, $T > 1$, and assume that for every $1 \leq i \leq k$, player i goes to box i in every round. Under the exclusive policy, this strategy is an equilibrium, whereas each player is clearly redundant. The following monotonicity lemmas hold under any congestion policy C.

▶ **Lemma 11.** *Consider two doubly-substochastic matrices A and B. If $A(x,t) > B(x,t)$, and for all $s < t$, $A(x,s) = B(x,s)$ then $v_A(x,t) < v_B(x,t)$.*

▶ **Lemma 12.** *Let A be doubly-substochastic. For every x and t, $v_A(x,t+1) \leq v_A(x,t)$. Moreover, if $A(x,t+1) > 0$ then the inequality is strict.*

Using the above, we prove a stronger result than Lemma 10 for the symmetric case:

▶ **Lemma 13.** *If A is a symmetric equilibrium then it is non-redundant.*

Proof. Because of Lemma 10 it is sufficient to consider only the case where $\texttt{success}(A) = 1$. Let $T' = \min\{M, T\}$, and assume by contradiction that A is redundant. Thus there is some $t \leq T'$ where $\sum_x A(x,t) < 1$. Hence, $\sum_{s \leq T'} \sum_x A(x,s) < T'$. Therefore, there is some x such that $\sum_{s \leq T'} A(x,s) < 1$ and so $\sum_{s \leq t} A(x,s) < 1$. As $\texttt{success}(A) = 1$, there is some $t' > t$ such that $A(x,t') > 0$. Define $A' = A + \epsilon(\delta_{x,t} - \delta_{x,t'})$. Taking $\epsilon > 0$ small enough, A' is doubly-substochastic. Also, $U_A(A') - U_A(A) = \epsilon(v_A(x,t) - v_A(x,t'))$, which is strictly positive by Lemma 12. Contradicting the fact that A is an equilibrium. ◀

3.2 Greedy Strategies

▶ **Lemma 14.** *A non-redundant strategy A is greedy w.r.t. $\mathbb{P}$ at time t iff for every x and y, if $A(x,t) > 0$ and $v_{\mathbb{P}}(x,t) < v_{\mathbb{P}}(y,t)$ then $\widetilde{A}(y,t) = 0$.*

The lemma above gives a useful equivalent definition for greediness. We can then prove:

▶ **Theorem 2.** *For every policy C there exists a non-redundant sgreedy strategy. Moreover, all such strategies are equivalent and are symmetric $(1 + C(k))$-equilibria.*

Proof. See the full version for a proof for the existence of a strategy A that is non-redundant and sgreedy. We prove here that such a strategy is a $(1 + C(k))$-equilibrium. Consider a strategy B. We compare the utility of B to that of A when both play against $k-1$ players playing A. By non-redundancy, all of $v_A(x,t)$ are 0 when $t > M$, and so we can assume $T \leq M$.

Denote $\texttt{maxv}(t) = \max_x v_A(x,t)$. Since the utility of B in any round t is a convex combination of $v_A(x,t)$, we have $U_A(B,t) \leq \texttt{maxv}(t)$. We say that A *fills* box x at round t if $A(x,t) > 0$ and $\widetilde{A}(x,t) = 0$. The following four claims hold for any round t:

1. If A does not fill any box at round t then $U_A(A,t) = \texttt{maxv}(t)$. This is because $U_A(A,t)$ is a convex combination of $v_A(x,t)$ for the boxes where $A(x,t) > 0$, which by the characterization of greediness in Lemma 14, all have the same value at time t.
2. $U_A(A,1) = \texttt{maxv}(1)$. Why? if no box is filled in round 1, then Item 1 applies. Otherwise, for some box x, $A(x,1) = 1$, and all other boxes have $A(\cdot,1) = 0$. The result follows again by Lemma 14.
3. For any $s < t$, $U_A(A,s) \geq \texttt{maxv}(t)$. We prove this by showing that for every x, $U_A(A,s) \geq v_A(x,t)$. If $v_A(x,t) = 0$, then the claim is clear. Otherwise, $A(x,t) > 0$ or $\widetilde{A}(x,t) > 0$ or both. Either way, $\widetilde{A}(x,s) > 0$. Therefore, as A is sgreedy, for every y such that $A(y,s) > 0$, $v_A(y,s) \geq v_A(x,s) \geq v_A(x,t)$. The last inequality follows from monotonicity, i.e., Lemma 12. As $v_A(A,s)$ is a convex combination of such y's we conclude.
4. If A fills box x at time $t > 1$ then for any $s < t$, $v_A(x,t) \leq C(k)v_A(x,s)$. To see why, first note that $v_A(x,s) \geq f(x)C(1)\widetilde{A}(x,s)^{k-1} = f(x)\widetilde{A}(x,s)^{k-1}$. On the other hand, since $\widetilde{A}(x,t) = 0$, $v_A(x,t) = f(x)C(k)A(x,t)^{k-1} \leq f(x)C(k)\widetilde{A}(x,s)^{k-1}$, because $A(x,t) \leq \widetilde{A}(x,t-1) \leq \widetilde{A}(x,s)$. Combining the above two inequalities gives the result.

Denote by X_1 the set of rounds for which there is no box x that is filled by A. Let X_2 be the rest of the rounds, except for $t = 1$ which is in neither. Also denote $t_0 = \min X_2$, and $t_1 = \max X_2$. Since $U_A(B) \leq \sum_t \texttt{maxv}(t)$, by Items 1,2 and 3 above,

$$U_A(B) \leq \sum_{t \in X_1 \cup \{1\}} U_A(A,t) + \texttt{maxv}(t_0) + \sum_{t \in X_2 \setminus \{t_1\}} U_A(A,t) \leq U_A(A) + \texttt{maxv}(t_0).$$

We conclude by using Items 4 and 2 and showing:

$$\texttt{maxv}(t_0) = \max_x v_A(x,t_0) \leq \max_x C(k)v_A(x,1) = C(k)U_A(A,1) \leq C(k)U_A(A). \qquad \blacktriangleleft$$

In the full version we provide an example showing that in the sharing policy, a non-redundant sgreedy strategy is not necessarily at equilibrium. On the other hand, it is worth noting that for any policy, the existence of a symmetric equilibrium follows from [23], and for $C(k) = 0$ we can get a full characterization of such equilibria:

▷ Claim 15. For any policy such that $C(k) = 0$, all symmetric equilibria are equivalent.

Interestingly, this result does not extend to non-symmetric profiles even for the exclusive policy, as is demonstrated by the following example of a non-greedy non-redundant equilibrium. Consider three players and two rounds. $f(1) = f(2) = f(3) = (1-\epsilon)/3, f(4) = \epsilon$, for some small positive ϵ. Player 1 plays first 4 and then 1. Player 2 plays 2 and then 3, and player 3 plays 3 and then 2. This can be seen to be an equilibrium, yet player 1 is not greedy.

Finally, the proof of Theorem 4, which shows how to construct an approximate equilibrium in polynomial time, is presented in the full version of the paper. This proof involves defining notions of approximate greediness and non-redundancy, proving an equivalent of Theorem 2 for them, and then using bounds on the rate of change that $v_A(x,t)$ goes through as a function of $A(x,t)$. This allows us to polynomially find an approximate sgreedy and non-redundant matrix, thus giving a polynomial strategy with our use of the Birkhoff von-Neumann theorem (see the full version).

4 The Exclusive Policy

Missing proofs of this section appear in the full version of this paper. There, we first prove that under the exclusive policy, $A^\star$ is sgreedy and non-redundant. Hence, according to Theorem 2,

▶ **Theorem 5.** *Under the exclusive policy, Strategy $A^\star$ of [8] is a symmetric equilibrium.*

According to Claim 3 all symmetric equilibria under the exclusive policy are equivalent, and thus equivalent to $A^\star$. Hence, the optimality of $A^\star$ (w.r.t. symmetric profiles) implies that both the PoSA and PoSS of the exclusive policy are optimal. That is, for every f, T, and policy C,

$$\texttt{PoSA}(C_{\texttt{ex}}, f, T) = \texttt{PoSS}(C_{\texttt{ex}}, f, T) \leq \texttt{PoSS}(C, f, T).$$

Our next goal is to establish the PoA of the exclusive policy. For this purpose, we first prove that the success probability of any equilibrium is at least as large as that of any symmetric profile. Since $A^\star$ is a symmetric equilibrium, its optimality among symmetric profiles follows. Hence, the proof provides as alternative proof to the one in [8].

▶ **Theorem 6.** *Consider the exclusive policy. For any profile $\mathbb{P}_{\texttt{nash}}$ at equilibrium and any symmetric profile A, $\texttt{success}(\mathbb{P}_{\texttt{nash}}) \geq \texttt{success}(A)$.*

Proof. Let A be a strategy and $\mathbb{P}_{\texttt{nash}}$ be a profile at equilibrium with respect to $C_{\texttt{ex}}$. If $\texttt{success}(\mathbb{P}_{\texttt{nash}}) = 1$, then the inequality is trivial. According to Lemma 10, we can therefore assume that all players of $\mathbb{P}_{\texttt{nash}}$ are non-redundant and that $T \leq M$. Denote the probability of visiting x in profile $\mathbb{P}$ by

$$\texttt{success}(\mathbb{P}, x) = 1 - \prod_{B \in \mathbb{P}} \widetilde{B}(x, T).$$

We say that box x is *high* with respect to a profile $\mathbb{P}$ if $\texttt{success}(\mathbb{P}, x) > \texttt{success}(A, x)$, *low* if $\texttt{success}(\mathbb{P}, x) < \texttt{success}(A, x)$, and *saturated* if they are equal. The next lemma uses the fact that A is symmetric.

▶ **Lemma 16.** *If a profile $\mathbb{P}$ is non-redundant and contains no high boxes, then all boxes are saturated.*

We proceed to prove a weak greediness property for equilibria. Denote a box x *full* for player B if $\sum_t B(x,t) = 1$. Also, for readability of what follows, when $\mathbb{P}$ is clear from the context, we shall denote $v_B(x,t) = v_{\mathbb{P}-B}(x,t) = f(x) \cdot \prod_{A \in \mathbb{P} \setminus \{B\}} \widetilde{A}(x,t)$.

▶ **Lemma 17.** *Consider a profile $\mathbb{P}_{\texttt{nash}}$ at equilibrium. For every $B \in \mathbb{P}_{\texttt{nash}}$ and t, x, y such that y is not full in B, if $B(x,t) > 0$ then $v_B(x,t) \geq v_B(y,t)$.*

Proof. Assume otherwise. Define an alternative matrix B' for player B, as $B' = B + \epsilon(\delta_{y,t} - \delta_{x,t})$. For a sufficiently small $\epsilon > 0$, B' is a doubly-substochastic matrix because y is not full in B. Then, $U_{\mathbb{P}^{-B}_{\texttt{nash}}}(B') - U_{\mathbb{P}^{-B}_{\texttt{nash}}}(B) = \epsilon(v_B(y,t) - v_B(x,t)) > 0$, in contradiction. ◀

Let us define a process that starts with the profile $\mathbb{P}_{\texttt{nash}}$ and changes it by a sequence of *alterations*, each shifting some amount of probability between two boxes. Importantly, we make sure that each alteration can only decrease the success probability. Hence, the proof is concluded once we show that the final profile has a success probability that is higher than that of A.

We first describe the alternations. Each alteration considers the current profile $\mathbb{P}$, and changes it to $\mathbb{P}'$. It takes some high box x, some low box y (both w.r.t. $\mathbb{P}$), and the maximal t such that there is a player $B \in \mathbb{P}$ with $B(x,t) > 0$. It defines $B' = B + \epsilon(\delta_{y,t} - \delta_{x,t})$, and lets the player that played B play B' instead. This ϵ is taken to be the largest so that x does not become low, y does not become high, and such that $\epsilon \leq B(x,t)$, so that the entries remain non-negative. Note that B' is doubly sub-stochastic, because taking care that y remains low, also means that y's row in B' still sums to less than 1.

After this alteration, either x is saturated, y is saturated, or $B'(x,t) = 0$. Clearly, in a finite number of alterations a profile $\mathbb{P}_{\texttt{final}}$ is obtained, for which either no box is high or no box is low.

▶ **Lemma 18.** $\texttt{success}(\mathbb{P}_{\texttt{final}}) \geq \texttt{success}(A)$.

Proof. By Lemma 16, $\mathbb{P}_{\texttt{final}}$ can only contain high and saturated boxes, that is, for every x, $\texttt{success}(\mathbb{P}_{\texttt{final}}, x) \geq \texttt{success}(A, x)$. However, $\texttt{success}(\mathbb{P}) = \sum_x f(x)\,\texttt{success}(\mathbb{P}, x)$, and so $\texttt{success}(\mathbb{P}_{\texttt{final}}) \geq \texttt{success}(A)$. ◀

Lastly, the following lemma concludes the proof of Theorem 6.

▶ **Lemma 19.** *An alteration can only decrease the probability of success.* ◀

Since $A^\star$ is a symmetric equilibrium, we immediately get that for every f and T, the PoA is attained by $A^\star$, that is, $\texttt{PoA}(C_{ex}, f, T) = \max_{\mathbb{P} \in \mathcal{P}(T)} \texttt{success}(\mathbb{P})/\texttt{success}(A^\star)$. Since $A^\star$ has the best success probability among symmetric profiles, and that every policy has some symmetric equilibrium, we get Corollary 7. To make this more concrete, we show that in the worst case,

▶ **Theorem 8.** *For every T, $\sup_f \texttt{PoA}(C_{\texttt{ex}}, f, T) = (1 - (1 - 1/k)^k)^{-1}$.*

Note that as k goes to infinity the PoA converges to $e/(e-1) \approx 1.582$.

5 Future Work and Open Questions

In [8], the main complexity measure was actually the running time and not the success probability. Our results about equilibria are also relevant to this measure, but the social gain is different. For example, it is still true that $A^\star$ is an equilibrium under the exclusive policy, and that all other symmetric equilibria in the exclusive policy are equivalent to it. As $A^\star$ is optimal among symmetric profiles w.r.t. the running time, the PoSA of $C_{\texttt{ex}}$ is equal to the PoSS, and it is also the best among all policies. Furthermore, importing from [8], we know that the PoSA (w.r.t. the running time) is about 4. However, showing the analogue of

Corollary 7, namely, that the PoA of $C_{\mathtt{ex}}$ is that achieved by $A^\star$, seems difficult, especially because general equilibria are not necessarily greedy. Moreover, the results of Vetta [31] do not apply when analyzing the running time, and finding the PoA, PoSA, and PoSS of the sharing policy, for example, remains open.

Another interesting variant would be to consider feedback during the search. For example, assuming that a player visiting a box x knows whether or not other players were there before. Such a feedback can help in the case that the players collaborate [5], but seems to significantly complicate the analysis in the game theoretic variant.

Finally, we would like to encourage game theoretical studies of other frameworks of collaborative search, e.g., [4, 6, 7, 15].

References

1 Susanne Albers and Matthias Hellwig. Online Makespan Minimization with Parallel Schedules. *CoRR*, abs/1304.5625, 2013. arXiv:1304.5625.

2 Sayan Bhattacharya, Sungjin Im, Janardhan Kulkarni, and Kamesh Munagala. Coordination mechanisms from (almost) all scheduling policies. In *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 121–134, 2014. doi:10.1145/2554797.2554811.

3 Artur Czumaj and Berthold Vöcking. Tight Bounds for Worst-case Equilibria. *ACM Trans. Algorithms*, 3(1):4:1–4:17, February 2007. doi:10.1145/1186810.1186814.

4 Dariusz Dereniowski, Yann Disser, Adrian Kosowski, Dominik Pajak, and Przemyslaw Uznanski. Fast collaborative graph exploration. *Inf. Comput.*, 243:37–49, 2015. doi:10.1016/j.ic.2014.12.005.

5 Stefan Dobrev, Rastislav Královic, and Dana Pardubská. Treasure Hunt with Barely Communicating Agents. In *21st International Conference on Principles of Distributed Systems, OPODIS 2017, Lisbon, Portugal, December 18-20, 2017*, pages 14:1–14:16, 2017. doi:10.4230/LIPIcs.OPODIS.2017.14.

6 Yuval Emek, Tobias Langner, Jara Uitto, and Roger Wattenhofer. Solving the ANTS Problem with Asynchronous Finite State Machines. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, pages 471–482, 2014. doi:10.1007/978-3-662-43951-7_40.

7 Ofer Feinerman and Amos Korman. The ANTS problem. *Distributed Computing*, 30(3):149–168, 2017. doi:10.1007/s00446-016-0285-8.

8 Pierre Fraigniaud, Amos Korman, and Yoav Rodeh. Parallel Bayesian Search with no Coordination. *J. ACM*, 2019.

9 Peter Frazier, David Kempe, Jon Kleinberg, and Robert Kleinberg. Incentivizing exploration. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 5–22. ACM, 2014.

10 Martin Gairing. Covering Games: Approximation through Non-cooperation. In *Internet and Network Economics, 5th International Workshop, WINE 2009, Rome, Italy, December 14-18, 2009. Proceedings*, pages 184–195, 2009. doi:10.1007/978-3-642-10841-9_18.

11 Luc-Alain Giraldeau and Thomas Caraco. Social Foraging Theory. *Princeton University Press*, 2000.

12 Ronen Gradwohl and Omer Reingold. Fault tolerance in large games. *Games and Economic Behavior*, 86:438–457, 2014. doi:10.1016/j.geb.2013.06.007.

13 Joseph Y. Halpern. A computer scientist looks at game theory. *Games and Economic Behavior*, 45(1):114–131, 2003. doi:10.1016/S0899-8256(02)00529-8.

14 Joseph Y. Halpern. Computer Science and Game Theory: A Brief Survey. *CoRR*, abs/cs/0703148, 2007. arXiv:cs/0703148.

15 Yuya Higashikawa, Naoki Katoh, Stefan Langerman, and Shin-Ichi Tanigawa. Online Graph Exploration Algorithms for Cycles and Trees by Multiple Searchers. *J. Comb. Optim.*, 28(2):480–495, August 2014. doi:10.1007/s10878-012-9571-y.

16 Thomas T Hills, Peter M Todd, David Lazer, A David Redish, Iain D Couzin, Cognitive Search Research Group, et al. Exploration versus exploitation in space, mind, and society. *Trends in cognitive sciences*, 19(1):46–54, 2015.

17 Martyn Kennedy and Russell D. Gray. Can Ecological Theory Predict the Distribution of Foraging Animals? A Critical Analysis of Experiments on the Ideal Free Distribution. *Oikos*, 68(1):158–166, 1993. URL: http://www.jstor.org/stable/3545322.

18 Jon Kleinberg and Sigal Oren. Mechanisms for (mis) allocating scientific credit. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 529–538. ACM, 2011.

19 Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. *Computer Science Review*, 3(2):65–69, 2009. doi:10.1016/j.cosrev.2009.04.003.

20 Ilan Kremer, Yishay Mansour, and Motty Perry. Implementing the "Wisdom of the Crowd". *Journal of Political Economy*, 122(5):988–1012, 2014.

21 Yishay Mansour, Aleksandrs Slivkins, and Vasilis Syrgkanis. Bayesian incentive-compatible bandit exploration. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 565–582. ACM, 2015.

22 Dov Monderer and Lloyd S Shapley. Potential games. *Games and economic behavior*, 14(1):124–143, 1996.

23 John Nash. Non-Cooperative Games. *Annals of Mathematics*, 54(2):286–295, 1951. URL: http://www.jstor.org/stable/1969529.

24 Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

25 University of California Berkeley. BOINC. https://boinc.berkeley.edu/, 2017.

26 Yiangos Papanastasiou, Kostas Bimpikis, and Nicos Savva. Crowdsourcing exploration. *Management Science*, 2017.

27 Arthur C Pigou. The Economics of Welfare. *Library of Economics and Liberty*, 1932.

28 Vinod Ramaswamy, Dario Paccagnan, and Jason R. Marden. The Impact of Local Information on the Performance of Multiagent Systems. *CoRR*, abs/1710.01409, 2017. arXiv:1710.01409.

29 Robert W Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2(1):65–67, 1973.

30 Christian Scheideler and Jeremy T. Fineman, editors. *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures, SPAA 2018, Vienna, Austria, July 16-18, 2018*. ACM, 2018. doi:10.1145/3210377.

31 Adrian Vetta. Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 416–425. IEEE, 2002.

Polynomial Anonymous Dynamic Distributed Computing Without a Unique Leader

Dariusz R. Kowalski
Department of Computer Science, University of Liverpool, UK
SWPS University of Social Sciences and Humanities, Warsaw, Poland
D.Kowalski@liverpool.ac.uk

Miguel A. Mosteiro
Computer Science Department, Pace University, New York, NY, USA
mmosteiro@pace.edu

Abstract

Counting the number of nodes in Anonymous Dynamic Networks is enticing from an algorithmic perspective: an important computation in a restricted platform with promising applications. Starting with Michail, Chatzigiannakis, and Spirakis [18], a flurry of papers sped up the running time guarantees from doubly-exponential to polynomial [16]. There is a common theme across all those works: a distinguished node is assumed to be present, because Counting cannot be solved deterministically without at least one.

In the present work we study challenging questions that naturally follow: how to efficiently count with more than one distinguished node, or how to count without any distinguished node. More importantly, what is the minimal information needed about these distinguished nodes and what is the best we can aim for (count precision, stochastic guarantees, etc.) without any. We present negative and positive results to answer these questions. To the best of our knowledge, this is the first work that addresses them.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Anonymous Dynamic Networks, Counting, distributed algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.147

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Funding This work was supported by the National Science Center Poland (NCN) grant 2017/25/B/ST6/02553; the UK Royal Society International Exchanges 2017 Round 3 Grant #170293; and Pace University SR Grant and Kenan Fund.

1 Introduction

The recent excitement around the problem of *Counting*, i.e., computing the number of nodes of an Anonymous Dynamic Network (ADN) comes as no surprise. On one hand, knowing the number of processors is a fundamental requirement to decide termination in a myriad of distributed algorithms. On the other hand, ADN is a very restrictive scenario from an algorithmic perspective: network nodes do not have identifiers and communication links may change arbitrarily and continuously over time. The combination of an important problem with harsh computational conditions is any algorithmist's delight.

Node anonymity in ADNs is motivated by expected applications of such communication infrastructure. For instance, in ad-hoc networks embedded in the Internet of Things, nodes may have to be deployed in a massive scale, and having unique identifiers may simply be impractical or inconvenient. Moreover, low node-cost expectations may introduce uncertainty about the number of nodes that will effectively startup. Hence, the need of Counting.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 147; pp. 147:1–147:15


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

Strikingly, the progress on deterministic Counting speed-ups over various works ranged over a broad spectrum: from unbounded [11, 12] to polynomial time [16], going through doubly-exponential [11] and exponential time [10, 7]. In all that fruitful work, the ADN model has been equipped with one distinguishable node[1]. The assumption is well motivated: in a seminal paper [18], it was shown that, without at least one such node, Counting cannot be solved deterministically. But what if we have more than one special node? Do these nodes really need to be *special*? In other words, is it enough to put one of two different programs on each of the nodes, that otherwise are all identical? Moreover, can we let the nodes choose at random which program to run and have no special nodes? To the best of our knowledge, this is the first work that considers these questions.

As the more general case where all nodes are identical, only differentiated by the program they run, let the set of n network nodes be formed by ℓ *black* nodes (the "special" ones) and $n - \ell$ *white* nodes (the "regular" ones). **Our first contribution** is negative results. On one hand, if ℓ is unknown, Counting cannot be solved deterministically. On the other hand, for randomized Counting algorithms, we show that if ℓ is unknown or zero, there exist executions when the algorithm does not stop.

Even knowing how many black nodes are in the network, straightforward application of previous ideas for Counting is not clear. Indeed, each black node may carry its own count, but how do we combine or compare final counts? Even passing messages among black nodes is challenging, because black nodes are also indistinguishable among them and communication links change arbitrarily. For instance, a black node is not able to tell whether a received message is even its own, coming back after being previously sent for dissemination.

Our second contribution is a deterministic Counting protocol that computes exactly the number of network nodes. Our protocol uses no information about the network, except the number of black nodes $\ell \geq 1$. After completing its execution, all nodes obtain the exact size of the network and stop. Moreover, they stop all at the same time, allowing the algorithm to be concatenated with other computations.

This protocol resembles our Methodical Counting protocol presented in [16]. So, we call it Methodical multi-Counting (MMC). However, it is not a simple combination of multiple instances of Methodical Counting to handle multiple black nodes. We overcome the challenge of how to combine the actions of multiple indistinguishable black nodes by careful design of a set of alarms, so that *all* black nodes can simultaneously detect when a running estimate of the size is correct. Moreover, the asymptotic performance of MMC is $O(n^{4+\epsilon}(\log^3 n)/\ell)$, for an arbitrarily small $\epsilon > 0$. This is a speed-up by a factor arbitrarily close to $n\ell/\log n$ with respect to Methodical Counting. That is, even for $\ell = 1$, MMC is faster than the best previous work.

In face of the impossibility of deterministic Counting without a distinguished node [18], an enticing question is what is possible introducing randomness. **Our third contribution** is a Counting protocol that computes exactly the number of nodes in an ADN where $\ell = 0$ (no special nodes). That is, we show that randomness breaks the impossibility result of [18], and for the first time it is possible to consider an ADN model where *all* nodes are identical, indistinguishable, and run the same program. Our protocol, called Leader-less Methodical Counting (LLMC), is Monte Carlo; i.e., there is a small probability $\epsilon > 0$ of obtaining the wrong size, and the running time of $O(n^{4+\epsilon} \log^3 n)$ holds with probability $1 - \epsilon$.

[1] Exactly one, usually called *leader*. We refrain from using the name leader to avoid confusion: in our model we may have more than one, and in our algorithm they are not going to select a single one among them. Moreover, they cannot even be local leaders, because due to ADN's dynamicity they may all be connected being their own (local) leaders.

To the best of our knowledge, this is the first comprehensive study of Counting in ADNs where ℓ may be different than one.

2 Previous Work

A comprehensive overview of work related to ADNs can be found in a survey by Casteigts et al. [5] and references in the papers cited here.

With respect to lower bounds, it was proved in [9] that at least $\Omega(\log n)$ rounds are needed, even if D is constant. Also, $\Omega(\mathcal{D})$ is a lower bound since at least one node needs to hear about all other nodes to obtain the right count.

Counting and *Naming* was already studied in [18] for dynamic and static networks, showing that it is impossible to solve Counting without the presence of a distinguished node, even if nodes do not move. The Counting protocol requires knowledge of an upper bound on Δ, and obtains only an upper bound, which may be as bad as exponential.

Conscious Counting [11] computes the exact count, but it needs to start from an upper bound, and it takes exponential time only if the size upper bound is tight. In the same work and follow-up papers [12, 13], more challenging scenarios where Δ is unknown are studied, but protocols either do not terminate [11], or the protocol is terminated heuristically [13]. In experiments [13], such heuristic was found to perform well on dense topologies, but for other topologies the error rate was high. Another protocol in [12] is shown to terminate eventually, without running-time guarantees and under the assumption of having for each node an estimate of the number of neighbors in each round. In [18] it was conjectured that some knowledge of the network such as the latter would be necessary, but the conjecture was disproved later in [10]. On the other hand the protocol in [10] requires exponential space.

Incremental Counting, presented recently in [19], reduced exponentially the best-known running time guarantees. The protocol obtains the exact count, all nodes terminate simultaneously, the topology dynamics is only limited to 1-interval connectivity, it only requires polynomial space, and it only requires knowledge of the dynamic maximum degree Δ. The running time is still exponential, but reducing from doubly-exponential was an important step towards understanding the complexity of Counting.

In a follow-up paper [6], Incremental Counting was tested experimentally showing a promising polynomial behavior. The study was conducted on pessimistic inputs designed to slow the convergence, such as bounded-degree trees rooted at the leader uniformly chosen at random for each round, and a single path starting at the leader with non-leader nodes permuted uniformly at random for each round. The protocol was also tested on static versions of the inputs mentioned, classic random graphs, and networks where some disconnection is allowed. The results exposed important observations. Indeed, even for topologies that stretch the dynamic diameter, the running times obtained are below Δn^3. It was also observed that random graphs, as used in previous experimental studies [13], reduce the convergence time, and therefore are not a good choice to indicate worst-case behavior. These experiments showed good behavior even for networks that sometimes are disconnected, indicating that more relaxed models of dynamics, such as (α, β)-connectivity [14, 15], are worth to study.

All in all, the experiments in [6] showed that Incremental Counting behaves well in a variety of pessimistic inputs, but not having a proof of what a worst-case input looks like, and being the experiments restricted to a range of values of n far from the expected massive size of an ADN, a theoretical proof of polynomial time remained an open problem even from a practical perspective.

A polynomial Counting algorithm was presented in a manuscript [2], relying on the availability of an algorithm to compute average with polynomial convergence time. Such average computation is modeled as a Markov chain with underlying doubly-stochastic matrix, which requires topology information within two hops (cf. [21]). In the pure model of ADN, such information is not available, and gathering it may not be possible due to possible topology changes from round to round.

Recently, we presented the first polynomial-time deterministic Counting algorithm for ADNs in [16], called Methodical Counting. Unlike previous works, Methodical Counting does not require any knowledge of network characteristics, such as dynamic maximum degree or an upper bound on the size. That is, it works in the pure model of ADN. Like previous works, Methodical Counting requires the presence of a distinguished node. In the present work, we generalize that assumption assuming the presence of $\ell \geq 1$ distinguished nodes. As in [16], we leverage previous work on lazy random walks to analyze MMC, but the alarms to detect wrong computations had been completely re-designed to deal with multiple distinguished nodes. Moreover, with respect to Methodical Counting, MMC achieves a $\Omega((\log^2 n)/(n\ell))$ speed-up. That is, even for $\ell = 1$, MMC also provides a speed-up with respect to previous work.

In the same paper [16], we also presented extensions of Methodical Counting to compute more complex functions, such as the sum of input values held by nodes and other algebraic and Boolean functions.

With respect to randomized Counting, a linear Counting algorithm for dynamic networks was presented in [17]. The algorithm requires unique identifiers (i.e., it is not applicable to ADNs), knowledge of an upper bound on the size of the network, and only guarantees an approximation to the network size. To the best of our knowledge, no randomized Counting algorithms for ADNs have been studied before.

Other studies also dealing with the time complexity of information gathering exist [8, 3, 22, 4, 20, 23], but include in their model additional assumptions, such as the network having the same topology frequently enough or node identifiers.

3 Model, Problem, and Notation

We define the Counting problem as follows. An algorithm $\mathcal{A}$ solves the *Counting Problem* if, after completing its execution, all network nodes running $\mathcal{A}$ have obtained the size of the network and stop (not necessarily concurrently). Notice that we focus on *exact* Counting. That is, all nodes obtain the exact size of the network n, rather than an approximation as other works. We define now a class of Counting algorithms.

For a given algorithm $\mathcal{A}$, let an *execution* of $\mathcal{A}$ be a sequence of steps of $\mathcal{A}$ followed in one of the possible sequence of choices made by $\mathcal{A}$. Let $\mathcal{X}(\mathcal{A})$ be the set of all possible executions of $\mathcal{A}$. An algorithm $\mathcal{A}$ is called *eventually stopping* if, for all $X \in \mathcal{X}(\mathcal{A})$, X has finite length.

We will model worst case scenarios assuming the presence of an adversary that controls the topology of the network. In particular, we consider the following adversaries.

Let the sequence $\mathcal{E} = \langle E_1, E_2, \dots \rangle$ be the sets of communication links of an ADN for time slots $t_1, t_2, \dots$. Consider the execution of an algorithm $\mathcal{A}$. We say that an adversary is *oblivious* if it determines the sequence $\mathcal{E}$ completely before the execution of $\mathcal{A}$ begins. On the other hand, we say that an adversary is *adaptive* if it determines the sequence $\mathcal{E}$ during the execution of $\mathcal{A}$, according to the actions of $\mathcal{A}$.

Notice that the distinction between oblivious and adaptive makes sense only for randomized algorithms, given that for deterministic algorithms the actions of the algorithm are defined before the execution.

The following model is customary in the ADNs literature. We consider a network composed by a set V of $n > 1$ network *nodes* with processing and communication capabilities. It was shown in [18] that Counting cannot be solved in Anonymous Networks without the availability of at least one distinguished node in the network. Hence, all previous studies of Counting in ADNs included in the model the presence of such node called the *leader*. However, to the best of our knowledge, nothing is known about deterministic Counting in presence of *multiple* distinguished nodes. In this work, we generalize the ADN model assuming that the number of distinguished nodes is $\ell \geq 1$. Aside from the distinction between distinguished and not-distinguished, all nodes are indistinguishable within their group; we call them black nodes and white nodes resp. All black nodes execute exactly the same program, and all white nodes execute exactly the same program. That is, there are no identifiers that allow to distinguish one black (resp. white) node from another black (resp. white) node. (Although we label the nodes throughout the paper for the sake of presentation and analysis.)

Each pair of nodes that are able to communicate define a communication *link*, and the set of links is called the *topology* of the network. The nodes in a communication link are called *neighbors*. The event of sending a message to neighbors is called a *broadcast* or *transmission*. Nodes and links are reliable, in the sense that no communication or node failures occur. Hence, a broadcasted message is received by all neighbors. Moreover, links are *symmetric*, that is, if node a is able to send a message to node b, then b is able to send a message to a.

Without loss of generality, we discretize time in *rounds*. In any given round, a node may broadcast a message, receive all messages from broadcasting neighbors, and carry out some computations, in that order. Time needed for computations is assumed negligible and the size of messages is unbounded.

The set of links among nodes may change from round to round, and nodes have no way of knowing which were the neighbors they had before. These topology changes are arbitrary, limited only to maintain the network connected in each round. That is, at any given round the topology is such that there is a *path*, i.e., a sequence of links, between each pair of nodes, but the set of links may change arbitrarily from round to round. This adversarial model of dynamics is known as 1-*interval connectivity* in [17].

The following notation will be used. The maximum number of neighbors that any node may have at any given time, called the *dynamic maximum degree*, is denoted as Δ or $d_{\max}$ indistinctively. The maximum length of a path between any pair of nodes at any given time is called the *dynamic diameter* and it is denoted as D. The maximum length of an opportunistic path between any pair of nodes over many time slots is called the *chronopath* [14] and it is denoted as $\mathcal{D}$.

4 Impossibility of Counting

Note that the results of this section hold even for static anonymous networks. The proofs of the following lemmas are left to the full version of this paper for brevity.

▶ **Lemma 1.** *For every positive integer ℓ there are two networks of $\ell + 4$ and $2(\ell + 4)$ nodes, respectively, such that:*

- **(i)** *no deterministic algorithm could successfully accomplish Counting on both of them in finite time, even if some ℓ nodes are black in the former and 2ℓ nodes are black in the latter network.*
- **(ii)** *for any randomized algorithm there is an execution in which no node outputs a correct count in a finite time, if no node is initially black in any of them or even if some ℓ nodes are black in the former and 2ℓ nodes are black in the latter network.*

The following results follow immediately from the lemma.

▶ **Theorem 2.** *If the number of black nodes is not given as a parameter, then there is no deterministic algorithm accomplishing Counting in finite time in ADNs.*

▶ **Theorem 3.** *If the number of black nodes is not given as a parameter or if there is no black node, then for every randomized algorithm there is an execution in which no node outputs correct node count in finite time in some fixed anonymous networks.*

▶ **Corollary 4.** *If there is no black node or their number is not explicitly known to the nodes, there is no eventually stopping randomized algorithm accomplishing Counting in ADNs even with approximation smaller than $\sqrt{2}$ and even against an oblivious adversary.*

Since there is no eventually stopping PTAS algorithm for Counting if the exact number of black nodes is unknown or (in case of randomized algorithms) there is no black node, we pursue two directions. One is to design a polynomial-time deterministic algorithm for exact Counting if the exact number of black nodes is apriori known. The other direction is to design a randomized algorithm computing the exact number of nodes with an arbitrary probability $1 - \epsilon$, for any $\epsilon \in (0, 1)$, even in scenario when no node is black prior the computation. The latter algorithm is polynomial in the sense of expected number of rounds and also holds with high probability (i.e., probability polynomially close to 1 in terms of n).

Remark: the inapproximability result could be extended from $\sqrt{2}$ to *any* constant by considering networks with $c \cdot (\ell + 4)$ nodes, for an arbitrary constant c instead of $c = 2$ used in the above proofs for the ease of arguments.

5 Methodical multi-Counting

In this section we present MMC. For brevity, we give the intuition of the algorithm, leaving the pseudocode to the full version of this paper.

Initially, each of the ℓ black nodes is assigned a potential of 0 and each of the $n - \ell$ white nodes is assigned a potential of ℓ. Then, the algorithm is composed by epochs, each divided into phases composed by rounds of communication. Epoch k corresponds to a size estimate k that is iteratively updated from epoch to epoch until the correct value n is found. Each epoch is divided into p phases. The purpose of each phase is for the black nodes to collect as much potential as possible from white nodes in a mass-distribution fashion as follows.

Each phase is composed by r rounds of communication. In each round, each node broadcasts its potential and receives the potential of all its neighbors. Each node keeps only a fraction $1/d$ of the potentials received. The parameters p, r, and d are functions of k. The specific functions needed to guarantee correctness and saught efficiency are defined in Theorem 14. This varying way of distributing potential is different from previous approaches using mass distribution. After communication, each node updates its own potential accordingly. That is, it adds a fraction $1/d$ of the potentials received, and subtracts a fraction $1/d$ of the potential broadcasted times the number of potentials received. Then, a new round starts. At the end of each phase, each black node "consumes" its potential. That is, it increases an internal accumulator ρ with its current potential, which is zeroed for starting the next phase.

The correctness (or incorrectness) of the estimate is detected by various alarms as follows. A node stops the update of potential described, raises its potential to ℓ, and broadcasts an alarm status "low" in each round until the end of the epoch if any of the following happens: 1) at the end of the first phase its potential is above some threshold τ as defined

in Theorem 14, 2) at any round it receives more than $d-1$ messages, or 3) at any round receives an alarm status "low" from one of its neighbors. Case 1) allows the black nodes to detect that the estimate is wrong when $k^{1+\epsilon} < n$ for some $\epsilon > 0$ (Lemmas 10 and 12), case 2) allows the black nodes to detect that d is too small and hence the estimate is low, and case 3) allows dissemination of these alarms. (In "low" status the potential is set to ℓ to facilitate the analysis, but it is not strictly needed by the algorithm.)

At the end of each epoch, each black node checks the value of ρ and updates its status accordingly. If it is within some range, call it Γ, the current estimate is correct and each black node changes its status to "done". Otherwise, the estimate is incorrect. If ρ is below (resp. above) Γ each black node changes its status to "high" (resp. "low") indicating that the estimate is too big (resp. "low"). The case when ρ is below (resp. above) Γ allows to detect when $k > n$ (resp. $k < n \leq k^{1+\epsilon}$) (c.f. Lemmas 7 and 13 respectively.).

After black nodes update their status, the network is flooded with it for k rounds. If $k \geq n$, those rounds are enough for all white nodes to receive the "done" or "high" status. If they receive "done", after completing the k rounds all nodes stop. Otherwise, after completing the k rounds all nodes update k according to status to start a new epoch. If k has not been detected to be greater than n since the computation started, it is doubled for the next epoch, otherwise it is updated as in binary search.

5.1 Analysis of Methodical multi-Counting

In this section we analyze MMC. We use standard notations $\mathbf{I}$ for the unit vector, and L_p for the norm of vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ as $||\mathbf{x}||_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$, for any $p \geq 1$. Only for the analysis, nodes are labeled as $0, 1, 2, \ldots, n-1$. The potential of a node i at the beginning of round s of phase t is denoted as $\Phi_{s,t}[i]$, the potential of all nodes is denoted as a vector $\mathbf{\Phi}_{s,t}$, and the aggregated potential is then $||\mathbf{\Phi}_{s,t}||_1$. The subindices s, t, or both are omitted sometimes for clarity. We will refer to the potential right after the last round of a phase as $\mathbf{\Phi}_{r+1}$. Such round does not exist in the algorithm, but we use this notation to distinguish between the potential right before black nodes consume their own potential and the potential at the beginning of the first round of the next phase.

First, we provide a broad description of our analysis of MMC. Consider the vector of potentials $\mathbf{\Phi}_i$ held by nodes at the beginning of any given phase i. The way that potentials are updated in each round is equivalent to the progression of a d-lazy random walk on the evolving graph underlying the network topology [1], where the initial vector of potentials is equivalent to an initial distribution $\mathbf{\Pi}_i$ on the overall potential $||\mathbf{\Phi}_i||_1$ and the probability of choosing a specific neighbor is $1/d$.

Note that MMC is not a simple "derandomization" of the lazy random walk on evolving graphs. First, in the Anonymous Dynamic Network model neighbors cannot be distinguished, and even their number is unknown at transmission time (only at receiving time the node learns the number of its neighbors). Second, due to unknown network parameters, it may happen in an execution of MMC that the total potential received could be bigger than 1. Third, our algorithm does not know a priori when to terminate and provide a result even with some reasonable accuracy, as the formulas on mixing and cover time of lazy random walks depend on the (a priori unknown) number of nodes n. Nevertheless, we can still use some results obtained in the context of analogous lazy random walks in order to prove useful properties of parts of MMC, namely, some parts in which parameters are temporarily fixed and the number of received messages does not exceed parameter d.

It was shown in [1] that random walks on d-regular explorable evolving graphs have a uniform stationary distribution, and bounds on the mixing and cover time were proved as well. Moreover, it was observed that those properties hold even if the graph is not regular and d is only an upper bound on the degree.[2]

Thus, for the cases where d is an upper bound on the number of neighboring nodes, we analyze the evolution of potentials within each phase leveraging previous work on random walks on evolving graphs. Specifically, we use the following result which is an extension of Corollary 14 in [1].

▶ **Theorem 5** (Corollary 14 in [1]). *After t rounds of a $d_{\max}$-lazy random walk on an evolving graph with n nodes, dynamic diameter D, upper bound on maximum degree $d_{\max}$, and initial distribution $\mathbf{\Pi}_0$, the following holds.*

$$\left|\left|\mathbf{\Pi}_t - \frac{\mathbf{I}}{n}\right|\right|_2^2 \leq \left(1 - \frac{1}{d_{\max} D n}\right)^t \left|\left|\mathbf{\Pi}_0 - \frac{\mathbf{I}}{n}\right|\right|_2^2$$

In between phases, black nodes "consume" their potential, effectively changing the distribution at that point. Then, a new phase starts.

In MMC, given that d is a function of the estimate k, if the estimate is low, there may be inputs for which d is not an upper bound on the number of neighbors. We show in our analysis that in those cases the black nodes detect the error and after some time all nodes increase the estimate.

Structure of the proof

The proof of correctness is structured in the following cases, depending on the relation between the size estimate k and n. For some γ and ϵ, after completing an epoch with size estimate k, we prove that ρ, the potential accumulated by a black node, must be within the following ranges.

$$\begin{aligned}
k = n \quad &\Rightarrow \quad (k-\ell)\left(1 - \frac{1}{k^\gamma}\right) \leq \rho \leq (k-\ell)\left(1 + \frac{1}{k^\gamma}\right) && \text{(Lemma 6)} \\
k > n \quad &\Rightarrow \quad \rho < (k-\ell)\left(1 - \frac{1}{k^\gamma}\right) && \text{(Lemma 13)} \\
k < n \leq k^{1+\epsilon} \quad &\Rightarrow \quad \rho > (k-\ell)\left(1 + \frac{1}{k^\gamma}\right) && \text{(Lemma 7)}
\end{aligned}$$

For the remaining case when $k^{1+\epsilon} < n$, we prove first the following relation between $\Phi_{r+1,1}$, the potential of any node at the end of the first phase, and a threshold τ.

$$\begin{aligned}
k^{1+\epsilon} < n \quad &\Rightarrow \quad \Phi_{r+1,1} > \tau \text{ for at least one node} && \text{(Lemma 10)} \\
k \geq n \quad &\Rightarrow \quad \Phi_{r+1,1} \leq \tau \text{ for all nodes} && \text{(Lemma 11)}
\end{aligned}$$

Thus, if $k^{1+\epsilon} < n$, and only if $k < n$, there is at least one node with potential above τ, which moves to a status "low" and spreads this alarm. We complete the proof with the following.

$$k^{1+\epsilon} < n \quad \Rightarrow \quad \text{all nodes receive an alarm "low" during phase 2} \quad \text{(Lemma 12)}$$

[2] Their analysis relies on Lemma 12, which bounds the eigenvalues of the transition matrix as long as it is stochastic, connected, symmetric, and non-zero entries lower bounded by $1/d$. Those conditions hold for all the transition matrices, even if the evolving graph is not regular.

Analysis

We start the analysis considering the case $k = n$ in the following lemma. The proofs of the lemmata in this section are left to the full version of this paper for brevity.

▶ **Lemma 6.** *If $d \geq k = n$, for an ADN with $\ell < k$ black nodes, for any $\gamma > 0$ there is a $\alpha \geq \max\{2, 1 + \gamma + \log_k 3\}$ such that, after running the MMC protocol for $p \geq (2\gamma \ln k)/\left(\ell\left(\frac{1}{k} + \frac{1}{k^\alpha}\right)\right)$ phases, each of $r \geq 2\alpha d k^2 \ln k$ rounds, the potential ρ consumed by each of the ℓ black nodes is such that*

$$(k-\ell)\left(1 - \frac{1}{k^\gamma}\right) \leq \rho \leq (k-\ell)\left(1 + \frac{1}{k^\gamma}\right).$$

The previous lemma shows that, after running MMC enough time, if for some black node it is $\rho > (k-\ell)\left(1 + \frac{1}{k^\gamma}\right)$ or $\rho < (k-\ell)\left(1 - \frac{1}{k^\gamma}\right)$, for some $\gamma > 0$, we know that the estimate k is wrong. However, the complementary case, that is, $(k-\ell)\left(1 - \frac{1}{k^\gamma}\right) \leq \rho \leq (k-\ell)\left(1 + \frac{1}{k^\gamma}\right)$, may occur even if the estimate is $k \neq n$ and hence the error has to be detected by other means. To prove correctness in that case we further separate the range of k in three cases. The first one, when $k < n \leq k^{1+\epsilon}$, for some $\epsilon > 0$, in the following lemma, which is based on upper bounding the potential left in the system after running MMC long enough. To ensure that $d \geq \Delta + 1$, we restrict $d \geq k^{1+\epsilon}$.

▶ **Lemma 7.** *Under the following conditions $1 < k < n \leq k^{1+\epsilon} \leq d$, $\epsilon > 0$, after running the MMC protocol for $p \geq 2\delta(\ln k)/(\ell\left(1/n + 1/k^\beta\right))$ phases, each of $r \geq 2\beta d k^{2+2\epsilon} \ln k$ rounds, under the following conditions $\beta \geq \log_k(n(2k^\delta + 1))$, $\beta > 2$, $\delta > \log_k(nk^\gamma/(nk^\gamma - (n-1)(k^\gamma + 1)))$, and $\gamma > \log_k(n-1)$. Then, the potential ρ consumed by any black node is $\rho > (k-\ell)(1 + 1/k^\gamma)$.*

We now consider the case $k^{1+\epsilon} < n$. First, we prove the following two claims that establish properties of the potential during the execution of MMC. (Recall that we use round $r+1$ to refer to potentials at the end of the phase right before black nodes consume their potential.)

▷ Claim 8. Given an ADN of n nodes running MMC with parameter d, for any round t of the first phase, such that $1 \leq t \leq r+1$, if d was larger than the number of neighbors of each node x for every round $t' < t$, then $||\mathbf{\Phi}_t||_1 = (n-\ell)\ell$.

▷ Claim 9. Given an ADN of n nodes running MMC, for any round t of any phase and any node x, it is $0 \leq \Phi_t[x] \leq \ell$.

To show that if $k^{1+\epsilon} < n$ MMC detects that the estimate is low, we focus on the first phase. We define a threshold τ and a number of rounds such that, after the first phase is completed, some nodes will have potential above τ and this can happen only if the estimate is low. Then we show that black nodes receive an alarm indicating that.

First, we show an upper bound of at most $k^{1+\epsilon}$ nodes with potential at most τ at the end of the first phase (Lemma 10). Thus, given that $k^{1+\epsilon} < n$, we know that there is at least one node with potential above τ at the end of the first phase. Second, we show that if the estimate is not low, that is $k \geq n$, then all nodes have potential at most τ at the end of the first phase (Lemma 11). That is, a potential above τ can only happen when indeed the estimate is low. Finally, we show that if $k^{1+\epsilon} < n$ an alarm "low" initiated by nodes with potential above τ must be received after $k^{1+\epsilon}$ further rounds of communication (Lemma 12).

▶ **Lemma 10.** *For $\epsilon > 0$, after running the first phase of the MMC protocol, there are at most $k^{1+\epsilon}$ nodes that have potential at most $\tau = \ell(1 - \ell/k^{1+\epsilon})$.*

▶ **Lemma 11.** *If $k \geq n$, $r \geq (4 + 2\epsilon - 2\ln(k^\epsilon - 1)/\ln k)dk^2 \ln k$, and $\epsilon > 0$, given that $k > \ell \geq 1$, at the end of the first phase no individual node should have potential larger than $\tau = \ell(1 - \ell/k^{1+\epsilon})$.*

The previous lemma shows that, if the estimate is "not-low" ($k \geq n$), at the end of the first phase all nodes must have "low" potential ($\Phi_{1,r+1} \leq \tau$). (Notice the inverse relation between estimate and potential.) In the following lemma we show that if $k^{1+\epsilon} < n$ (i.e. low estimate) there are some nodes with $\Phi_{1,r+1} > \tau$ (i.e. high potential), and that all the other nodes will know this within the following phase.

▶ **Lemma 12.** *If $k^{1+\epsilon} < n$, $0 < \epsilon \leq 1 + \log_k 4d$, and $r \geq (4 + 2\epsilon - 2\ln(k^\epsilon - 1)/\ln k)dk^2 \ln k$, within the following $k^{1+\epsilon}$ rounds after the first phase of the MMC protocol, all black nodes have received an alarm status "low".*

Finally, in the following lemma we show that if $k > n$, black nodes detect that the potential consumed is too low for the estimate k to be correct.

▶ **Lemma 13.** *Under the following conditions $d > k > n > \ell > 0$, for $\beta \geq \log_k(n(2k^\delta - 1))$, $\delta > \log_k(k^\gamma(n-\ell)/(k^\gamma - (n-\ell) - 1))$, and $\gamma > \log_k(n - \ell + 1)$, after running the MMC protocol for p phases and r rounds such that $p \leq 2\delta \ln k(1 - \ell\left(\frac{1}{n} - \frac{1}{k^\beta}\right))/(\ell\left(\frac{1}{n} - \frac{1}{k^\beta}\right))$, and $r \geq 2\beta dk^2 \ln k$, the potential ρ consumed by any black node is $\rho < (k - \ell)\left(1 - \frac{1}{k^\gamma}\right)$.*

Based on the above lemmata, we establish the correctness and running time of MMC:

▶ **Theorem 14.** *Given an ADN with n nodes, which include ℓ black nodes such that $n > \ell \geq 1$ black nodes, after running MMC for each estimate $k = \ell+1, \ell+2, \ell+3, \ldots, n$ with parameters: $d = k^{1+\epsilon}$, $p = \left\lceil \frac{2\ln k}{\ell} \max\left\{ \frac{\gamma}{1/k+1/k^\alpha}, \frac{\delta}{1/d+1/k^\beta} \right\} \right\rceil$,*
$r = \left\lceil 2dk^2(\ln k) \max\left\{ \alpha, \beta k^{2\epsilon}, 2 + \epsilon - \frac{\ln(k^\epsilon - 1)}{\ln k} \right\} \right\rceil$, and $\tau = \ell\left(1 - \frac{\ell}{k^{1+\epsilon}}\right)$, under the following conditions: $\epsilon > 0$, $\alpha \geq 1 + \gamma + \log_k 3$, $\beta \geq \log_k(d(2k^\delta + 1))$, $\gamma > \log_k(d-1)$, $\delta > \log_k \frac{dk^\gamma}{k^\gamma + 1 - d}$. Then, all nodes stop after at most $\sum_{k \in E \cup B}(pr + d)$ rounds of communication and output n, for $E = \{2^i(\ell+1) : i = 0, 1, \ldots, \log\lceil n/(\ell+1)\rceil\}$, and $B = \{(2^{\log\lceil n/(\ell+1)\rceil} - 2^i)(\ell+1) : i = 0, 1, \ldots, \log\lceil n/(\ell+1)\rceil - 2\}$.

▶ **Corollary 15.** *The time complexity of MMC on an ADN with ℓ black nodes and $n - \ell$ white nodes is $O\left(\frac{n^{4+\epsilon}}{\ell} \log^3 n\right)$, for any $\epsilon > 0$.*

6 Leader-less Methodical Counting

The main idea of LLMC (cf. Algorithm 1) is to consider consecutive powers of 2 as values of K, and for each such K to select black nodes locally with probability corresponding to the inverse of K and run MMC with $\ell = 1$. If $K \geq n$ and there is indeed one black node, MMC guarantees that all nodes find the proper count n of nodes and stop. There are however two problems: how to recognize if $K \geq n$ and what to do when there is no black node or at lest two black nodes. Algorithm LLMC overcomes these two issues implicitly, by combining the executions of MMC for consecutive powers of 2 with two other techniques:

- introducing parallel threads and carefully counting the number of threads with no black nodes recorded (locally) and requiring that their ratio is bigger than half (this is to recognize whether K is close to n with sufficiently high probability),
- making use of the fact that having more than one black node in the execution of MMC for $\ell = 1$ (as a subroutine of LLMC) cannot return an estimate bigger than as if it was run with one black node; therefore, taking the maximum of returned estimates over threads could mitigate the potential problem of more than one black node.

The main control parameter used in LLMC is K, which is an upper bound for estimates considered in one execution of the While loop (which we call epoch K). We start from sufficiently large K, to assure that starting from this value of K the chance of getting incorrect output or behavior of nodes (e.g., stopping at different times) is smaller than $\epsilon/2$ Within one While loop, we initiate $f(K)$ parallel threads and for each of them we select black nodes for the whole thread – trying to make sure that the chance of getting one black node is sufficiently large (we need it to argue about correct stopping), especially for $K \geq n$ (recall that we could not recognize for sure whether $K \geq n$ or not). Then in each thread independently we run MMC for $\ell = 1$ as a subroutine (hoping that we selected exactly one black node in the beginning of the loop), which checks all possible values of k from 1 to K to find a good estimate of n (i.e., we have to trim the execution of MMC to estimates $k \leq K$).

This approach does not work for $K < n$, as then the probability of getting more than one black node could be bigger than the one for one black node; however, we could eliminate such cases by monitoring the number of threads with no black node, which in case of $K < n$ should be compared with a large threshold (intuitively, we want LLMC to reach this threshold with high probability when K will be close to n). Note that a no-black-node thread will not output any estimate. If a thread identifies a good estimate, it puts it to the set *Count* and we do not enter the next iteration of While loop but stop, returning the maximum value in *Count* - this is to make preference to threads with one black node over those with more than one black node (we will argue that they could return values but not bigger than ones by threads with one black node).

Algorithm 1 LLMC algorithm. $\epsilon \in (0, 1)$.

procedure
  $K \leftarrow \lceil \lceil 12/(\epsilon) \rceil \rceil$ // $\lceil \lceil x \rceil \rceil$: the smallest power of 2 bigger than x
  $Count \leftarrow \emptyset$ // set of potentially "good" estimates computed in threads
  $EmptyThreads \leftarrow 0$ // number of threads with no black node detected
  while $Count = \emptyset$ or $EmptyThreads \leq f(K)/2$ **do**
    $Count \leftarrow \emptyset$, $EmptyThreads \leftarrow 0$
    $K \leftarrow 2K$
    Initiate $f(K) = 64\frac{\log(K/\epsilon)}{\log(e/(e-2))}$ parallel threads
     // parallel computation and messages sharing same resources/medium
    for each thread **do**
      for each node **do**
        Select to be a black node with probability $1/g(K)$, where $g(K) = K/2$
      end for
      $k \leftarrow MMC(K, 1)$ // refer to Algorithm 2
      if $k > 0$ **then**
        $Count \leftarrow Count \cup \{k\}$
      end if
      if no black node detected **then**
        Increase $EmptyThreads$ by 1
      end if
    end for
  end while
  return max($Count$) // Output the maximum number in $Count$ as the size n.
end procedure

Algorithm 2 Subroutine of LLMC.

Input: number of black nodes ℓ, max size estimate K.

procedure MMC (K,ℓ)
  Run MMC modified as follows:
    – Stop iterations when size estimate $k > K$
    – If estimate $k < K$, remain idle until end of phase K // `for synchronization`
    – Include a Boolean p_j in each node j as follows:
      – Initially:
        if node j is black **then** $p_j \leftarrow true$ **else** $p_j \leftarrow false$
      – In each iteration:
        Broadcast and Receive messages including p_j
        if $p_i = true$ received from some neighbor i **then** $p_j \leftarrow true$
  Upon completion:
    if $status = done$ **return** k **else return** 0
end procedure

6.1 Analysis of Leader-less Methodical Counting

We may assume that $n > 1$, otherwise the algorithm would easily recognize $n = 1$ by no received communication. Throughout the whole analysis we consider an adaptive adversary, as we allow network changes to be done online when viewing the whole history of the computation up to the current round.

We call an execution of the While loop for a fixed parameter K *epoch* K. Recall that K is a power of 2 bigger than $12/(\epsilon)$; by $\lceil\lceil x\rceil\rceil$ we denote the smallest power of 2 bigger than x.

Observe from the structure of the algorithm that nodes could only stop at the end of an epoch. In the analysis below, we will often prove some properties under condition that the stopping times are synchronized, i.e., either all nodes stop or none, and remove this assumption at the end of the analysis. That is, we will show that in fact all nodes synchronize their stopping time and output the correct value of n with desired probability at least $1 - \epsilon$. We also conjecture that all arbitrary constants in the algorithm, i.e., in the definition of starting value of K, functions $f(K)$ and $g(K)$, could be substantially lowered, as we set them high to avoid too many cases in the analysis (so making it focused on main arguments).

The proofs of the following lemmata are left to the full version of this paper for brevity.

▶ **Lemma 16.** *The probability that for some epoch K, where $K < n$, the value of EmptyThreads is bigger than $f(K)/2$ at any node is smaller than $1 - \epsilon/2$. Consequently, with probability at least $1 - \epsilon/2$: no node stops during epochs $K < n$.*

Recall from the description of the algorithm that nodes could only stop at the end of an epoch. We prove the following two structural properties of LLMC.

▶ **Proposition 17.** *Consider an epoch $K \geq n$ and assume that all nodes are active in the beginning of this epoch. If some nodes stop while some other do not at the end of epoch K, then there are more than $f(K)/2$ threads with no black node selected and no thread with exactly one black node.*

▶ **Proposition 18.** *Consider an epoch $K \geq n$ and assume that all nodes are active in the beginning of this epoch. If all nodes stop but some of them output incorrect value, then there are more than $f(K)/2$ threads with no black node and no thread with exactly one black node.*

We use these structural properties to estimate the probabilities of the two following events.

▶ **Lemma 19.** *Consider an epoch $K \geq n$ and assume that all nodes are active in the beginning of this epoch. The probability of event: in epoch K there will be some nodes that stop and some other that do not, or all nodes stop with incorrect output, is at most ϵ/K.*

▶ **Lemma 20.** *The probability of some nodes stopping at different times or outputting incorrect value is at most $\epsilon/2$.*

We now analyze good events of correct stopping by all nodes, moreover, simultaneously. We start from stating a structural property and follow with estimating of the probability of correct simultaneous stopping.

▶ **Proposition 21.** *For any epoch $K \geq n$, if all nodes are active in the beginning, the number of threads with no black node is bigger than $f(K)/2$ and there is at least one thread with one black node, then all threads with one black node store the same value in set Count and it is the biggest value stored in Count in this epoch.*

▶ **Lemma 22.** *Consider an epoch $K \geq n$ and assume that all nodes are active in the beginning of this epoch. The probability of event: in epoch K all nodes stop with correct value, is at least $1 - \exp(-f(K)/64) - \exp\left(-\frac{nf(K)}{g(K)e}\right)$.*

▶ **Corollary 23.** *Consider epoch $K = \lceil \lceil 8n \rceil \rceil$ and assume that all nodes are active in the beginning of this epoch. The probability of event: in epoch K all nodes stop with correct value, is at least $1 - \epsilon/n$.*

▶ **Theorem 24.** *For any given $\epsilon > 0$, LLMC simultaneously stops at all nodes returning correct count n of nodes in $O(n^4 \log^4 n)$ rounds, with probability at least $1 - \epsilon$, even when running against an adaptive adversary.*

7 Conclusions

This paper expanded the knowledge about feasibility of polynomial computations in anonymous dynamic environments/networks. In particular, counting-type computations can be done deterministically if symmetry is broken by existing a partition of nodes where the size of one subset is known. It is also feasible without any distinction between the nodes with an arbitrary probability. Natural open directions include a study of the message complexity of MMC and LLMC, randomized approximation solutions and extensions to other related models and problems, as well as deriving tighter upper and lower bounds in the considered setting.

References

1. Chen Avin, Michal Koucký, and Zvi Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *Automata, languages and programming*, pages 121–132. Springer, 2008.
2. Roberto Baldoni and Giuseppe A Di Luna. Counting on Anonymous Dynamic Networks: Bounds and Algorithms. manuscript, 2016.
3. Siddhartha Banerjee, Aditya Gopalan, Abhik Kumar Das, and Sanjay Shakkottai. Epidemic spreading with external agents. *IEEE Transactions on Information Theory*, 60(7):4125–4138, 2014.
4. Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, 2006.

5 Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.

6 Maitri Chakraborty, Alessia Milani, and Miguel A. Mosteiro. Counting in Practical Anonymous Dynamic Networks is Polynomial. In *Proceedings of the 4th International Conference on Networked Systems*, volume 9944 of *Lecture Notes in Computer Science*, pages 131–136, 2016.

7 Maitri Chakraborty, Alessia Milani, and Miguel A. Mosteiro. A Faster Exact-Counting Protocol for Anonymous Dynamic Networks. *Algorithmica*, 80(11):3023–3049, 2018. doi:10.1007/s00453-017-0367-4.

8 Yuxin Chen, Sanjay Shakkottai, and Jeffrey G Andrews. On the role of mobility for multimessage gossip. *IEEE Transactions on Information Theory*, 59(6):3953–3970, 2013.

9 Giuseppe Antonio Di Luna and Roberto Baldoni. Investigating the Cost of Anonymity on Dynamic Networks. *CoRR*, abs/1505.03509, 2015. arXiv:1505.03509.

10 Giuseppe Antonio Di Luna and Roberto Baldoni. Non Trivial Computations in Anonymous Dynamic Networks. In *Proceedings of the 19th International Conference on Principles of Distributed Systems*, Leibniz International Proceedings in Informatics, 2015. To appear.

11 Giuseppe Antonio Di Luna, Roberto Baldoni, Silvia Bonomi, and Ioannis Chatzigiannakis. Conscious and Unconscious Counting on Anonymous Dynamic Networks. In *Proceedings of the 15th International Conference on Distributed Computing and Networking*, volume 8314 of *Lecture Notes in Computer Science*, pages 257–271. Springer Berlin Heidelberg, 2014.

12 Giuseppe Antonio Di Luna, Roberto Baldoni, Silvia Bonomi, and Ioannis Chatzigiannakis. Counting in Anonymous Dynamic Networks under Worst-Case Adversary. In *Proceedings of the 34th International Conference on Distributed Computing Systems*, pages 338–347. IEEE, 2014.

13 Giuseppe Antonio Di Luna, Silvia Bonomi, Ioannis Chatzigiannakis, and Roberto Baldoni. Counting in Anonymous Dynamic Networks: An Experimental Perspective. In *Proceedings of the 9th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, volume 8243 of *Lecture Notes in Computer Science*, pages 139–154. Springer Berlin Heidelberg, 2014.

14 M. Farach-Colton, A. Fernández Anta, A. Milani, M. A. Mosteiro, and S. Zaks. Opportunistic Information Dissemination in Mobile Ad-hoc Networks: adaptiveness vs. obliviousness and randomization vs. determinism. In *Proc. of the 10th Latin American Theoretical Informatics Symposium*, volume 7256 of *Lecture Notes in Computer Science*, pages 303–314. Springer-Verlag, Berlin, 2012.

15 Antonio Fernández Anta, Alessia Milani, Miguel A. Mosteiro, and Shmuel Zaks. Opportunistic information dissemination in mobile ad-hoc networks: the profit of global synchrony. *Distributed Computing*, 25(4):279–296, 2012. doi:10.1007/s00446-012-0165-9.

16 Dariusz R. Kowalski and Miguel A. Mosteiro. Polynomial Counting in Anonymous Dynamic Networks with Applications to Anonymous Dynamic Algebraic Computations. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 156:1–156:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.156.

17 Fabian Kuhn, Nancy Lynch, and Rotem Oshman. Distributed Computation in Dynamic Networks. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, pages 513–522. ACM, 2010.

18 Othon Michail, Ioannis Chatzigiannakis, and Paul G Spirakis. Naming and counting in anonymous unknown dynamic networks. In *Stabilization, Safety, and Security of Distributed Systems*, pages 281–295. Springer, 2013.

19 Alessia Milani and Miguel A. Mosteiro. A Faster Counting Protocol for Anonymous Dynamic Networks. In *Proceedings of the 19th International Conference on Principles of Distributed Systems*, volume 46 of *Leibniz International Proceedings in Informatics*, pages 1–13, 2015.

20 Damon Mosk-Aoyama and Devavrat Shah. Fast distributed algorithms for computing separable functions. *IEEE Transactions on Information Theory*, 54(7):2997–3007, 2008.

21 Angelia Nedic, Alex Olshevsky, Asuman Ozdaglar, and John N Tsitsiklis. On distributed averaging algorithms and quantization effects. *IEEE Transactions on Automatic Control*, 54(11):2506–2517, 2009.

22 Sujay Sanghavi, Bruce Hajek, and Laurent Massoulié. Gossiping with multiple messages. *IEEE Transactions on Information Theory*, 53(12):4640–4654, 2007.

23 Atish Das Sarma, Anisur Rahaman Molla, and Gopal Pandurangan. Distributed computation in dynamic networks via random walks. *Theoretical Computer Science*, 581:45–66, 2015.

Noidy Conmunixatipn: On the Convergence of the Averaging Population Protocol

Frederik Mallmann-Trenn
MIT, CSAIL, Cambridge, MA, US
mallmann@mit.edu

Yannic Maus
Department of Computer Science, Technion, Haifa, Israel,
yannic.maus@cs.technion.ac.il

Dominik Pajak
Faculty of Fundamental Problems of Technology,
Wroclaw University of Science and Technology, Poland
Tooploox, Wroclaw, Poland
dominik.pajak@pwr.edu.pl

Abstract

We study a process of *averaging* in a distributed system with *noisy communication*. Each of the agents in the system starts with some value and the goal of each agent is to compute the average of all the initial values. In each round, one pair of agents is drawn uniformly at random from the whole population, communicates with each other and each of these two agents updates their local value based on their own value and the received message. The communication is noisy and whenever an agent sends any value v, the receiving agent receives $v + N$, where N is a zero-mean Gaussian random variable. The two quality measures of interest are (i) the total sum of squares $TSS(t)$, which measures the sum of square distances from the average load to the *initial average* and (ii) $\bar{\phi}(t)$, which measures the sum of square distances from the average load to the *running average* (average at time t).

It is known that the simple averaging protocol – in which an agent sends its current value and sets its new value to the average of the received value and its current value – converges eventually to a state where $\bar{\phi}(t)$ is small. It has been observed that $TSS(t)$, due to the noise, eventually diverges and previous research – mostly in control theory – has focused on showing eventual convergence w.r.t. the running average. We obtain the first probabilistic bounds on the convergence time of $\bar{\phi}(t)$ and precise bounds on the drift of $TSS(t)$ that show that although $TSS(t)$ eventually diverges, for a wide and interesting range of parameters, $TSS(t)$ stays small for a number of rounds that is polynomial in the number of agents. Our results extend to the synchronous setting and settings where the agents are restricted to discrete values and perform rounding.

2012 ACM Subject Classification Theory of computation → Distributed computing models

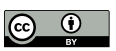
Keywords and phrases population protocols, noisy communication, distributed averaging

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.148

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Related Version See [43] (`http://arxiv.org/abs/1904.10984`) for the full version with all proofs.

Funding *Frederik Mallmann-Trenn*: NSF CCF-1461559, CCF-0939370, CCF-18107.
Yannic Maus: ERC Grant No. 336495 (ACDC).
Dominik Pajak: Polish National Science Center grant UMO-2018/29/B/ST6/02969 and NSF Award Numbers CCF-1461559, CCF-0939370, CCF-18107.



46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 148; pp. 148:1–148:16

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

1 Introduction

We consider the problem of distributed averaging by a group of agents (*e.g.,* sensors), initialized with values that represent, for example, different temperature measurements. The agents' goal is to compute the average of all the initial values using the following simple dynamic: In each discrete round, two agents are drawn uniformly at random from the whole population, communicate their values to each other and set their new values to the average of their old value and the received value. Converging to the average plays a key role in many applications, e.g., for sensor networks [56, 50], social insects [10], and robotics [20, 30]. In all of these applications, the agents (sensors, ants, and robots) are very simple and are therefore limited in both memory and communication. Moreover, communication is often erroneouees.[1] This motivates the study of the aforementioned simple averaging dynamic in a setting where the agents only remember one value, do not use any additional memory, and the communication is subject to noise. We model the noise in the communication as follows: Whenever an agent sends any value v, the receiving agent receives $v + N$, where random variable N is distributed according to some zero-mean probability distribution $\aleph$, *e.g.,* a normal distribution. The agents update their values as follows: whenever two agents communicate, each agent sets its new value to the average of their old value and the received value; note that – due to the noise – the two agents might have distinct new values.

The values of the n nodes in step t of the process are denoted by $X_1^{(t)}, X_2^{(t)}, \ldots, X_n^{(t)}$. We consider the following models: (i) the *sequential setting* where one pair of agents is chosen uniformly at random and (ii) the *synchronous setting* where each agent is matched to exactly one other agent chosen uniformly at random. The two quality measures of the convergence used in this work are (i) the total sum of squares $TSS(t) = \sum_i (X_i^{(t)} - \varnothing^{(0)})^2$, where $\varnothing^{(0)} = \sum_i X_i^{(0)}/n$ is the initial average and (ii) the sum of squared distances to the running average $\bar{\phi}(t) = \sum_i (X_i^{(t)} - \varnothing^{(t)})^2$, where $\varnothing^{(t)} = \sum_i X_i^{(t)}/n$ is the *running average*. Our contributions can be informally summarized as follows:

- **(i)** We give, under mild assumptions on the noise, the first bounds on the convergence time of the running average $\bar{\phi}(t)$ in the noisy gossip-based communication setting. The bounds we obtain are – up to a constant factor – tight. In particular, the potential converges to a value that is linear in n and the second moment of the noise $\mathbb{E}\left[N^2\right]$; which is tight. So far it was only known that the process eventually converges to a state where $\bar{\phi}(t)$ is small (*e.g.,* [54]), but precise bounds were not known. (Theorem 1)
- **(ii)** We show that, in contrast to the current belief, one can hope to converge to the *initial* average in addition to convergence to the *running* average as long as the number of rounds are bounded: It was known that $TSS(t)$, due to the noise, eventually diverges (the running average diverges from the initial average) and for this reason related research – mostly in control theory – has focused on showing eventual convergence w.r.t. $\bar{\phi}(t)$; leaving $TSS(t)$ aside. Since we give precise bounds on the convergence time of the running average, we can show the following. Under mild assumptions on the noise, $TSS(t)$ converges to almost the same value as $\bar{\phi}(t)$ as long as the number of time steps t is bounded by $O(n^2)$, where n is the number of nodes. (Corollary 2)
- **(iii)** We pioneer in the discrete setting in which the agents can store only integer values and the noise is also an integer. In this setting the agents in our algorithm perform randomized rounding. We show that this only causes a negligible difference from the continuous case. (Corollary 3)

[1] Consult Subsection 1.1 for a more detailed review of these applications including the limitation of agents and further motivation. Subsection 1.1 also contains related work on the averaging protocol.

(iv) We study both the sequential and the synchronous setting and show that there is no significant difference (up to a scaling of time) between the models. (Corollary 4)
(v) We perform simulations in the setting where nodes are limited in storage, *i.e.,* they can only store values from a bounded range. This leads to a much faster (by order of magnitude) divergence between the running average and the initial average. Our simulations also seem to indicate strong bounds on the distribution of distances to the running average in our main model (unbounded values). (Section 5)

The convergence time of the averaging processes in the gossip-based communication setting *without* noise has been studied before (e.g., [37]). However, to the best of our knowledge, no bounds on the convergence time are known in the gossip-based communication setting with noise. We continue with a detailed motivation for studying noise in the simple averaging dynamic and related work.

1.1 Motivation and Related Work

Converging to the average plays a key role in many applications in which agents have limited computational and communication power, e.g.,

(i) sensor networks [56, 50]: here there is a wide range of application including terrain monitor applications [51], computing an average temperature, PIR sensors measuring the infrared light radiation emitted from objects, and many more applications. In such scenarios links are often faded [46, 14],
(ii) social insects: for ants, values could represent the individuals' different assessments of nest qualities when house hunting [10] or the deficit of workers at a given task [41], and
(iii) robotics [20, 30] and in particular memory-limited robots, *e.g.,* Kilobots exploring the percentage of white tiles in an area [21], or microbots measuring the concentration of chemicals.

In all of these applications the agents (representing sensors, ants or robots) are very simple and severely limited in both memory and communication. Moreover, the communication is often not only limited but also erroneous (e.g., consider wireless communication with obstacles between robots), or received messages are subject to interpretation (e.g., when insects communicate through gestures [39]). Motivated by this unreliable communication in applications we study the simple averaging dynamic where the communication is subject to noise.

We continue with related work. The problem of distributed values converging to the average (often without noise) has been studied in various areas reaching back to early versions studied in statistics [18, 26, 31]. However, to the best of our knowledge, none of the studied models match our model. We review the related work by areas:

(i) average consensus and its applications,
(ii) gossip-based communication models,
(iii) consensus protocols in population protocols,
(iv) biological distributed algorithms,
(v) noise and failures in sensor networks.

Average consensus and its applications. Consensus has been studied intensively in various settings in general network topologies, much of it under the name of *average consensus* [55, 53]. Most of this work is orthogonal to our work: First, due to the general network topology and the fact that, in each step of the studied algorithms, the agents update their values with a weighted average of all of their neighbors' values whereas in our averaging dynamic, an agent can only access a single other value per interaction. Second, while the

potential functions in these works and the noise, if any, are usually identically or similarly defined as in our work the main goal of these papers is – just as in the classic works – to study under which circumstances the processes eventually converge to a state with a small potential function [55], whereas we are interested in the number of interactions until our process obtains a small potential. Recent papers [45, 11, 40, 15] consider the convergence rate of the weighted averaging process, but only in the noiseless setting. Average consensus has also been studied in networks with time-varying topologies [44, 49]. Variants with noisy communication were studied [55, 36], but they only consider additive noise and assume it to be zero-mean with unit variance (as mentioned before, only convergence in the limit is shown). The noisy version of the problem also received ample attention in control theory [52, 48, 47]. Already in the early works on average consensus immediate applications of converging to the average were discovered and intensively studied, e.g., applications to load balancing between parallel machines [9, 17] or to coordinate distributed mobile agents [9, 34, 23]. For a more detailed overview on average linear consensus consult the survey [27].

Gossip-based communication models. Much closer to our work is the study of aggregating information in gossip-based model. In this model, each node can contact one of its neighbors in the network in each round and exchange information with it. Even though a node can be contacted by many neighbors in a single round, this model, if applied to the complete graph, is very similar to our synchronous model. On the complete graph [37] shows that $O(n \cdot \ln n)$ interactions are enough to approximate the average well with high probability. On the one hand they consider more general graphs (in some sense we consider the complete graph); on the other hand they do not consider noise, which simplifies their analysis of the convergence time significantly.

Consensus protocols in population protocols, biological distributed algorithms. Motivated by biological applications, population protocols have also been studied in the noisy setting in the context of biological distributed algorithms. The authors of [24] study rumor spreading and consensus in extremely faulty networks where a bit in a message can be flipped with probability $1/2 - \varepsilon$. This was later generalized in [25] to plurality consensus. The authors of [8] study the differences between pull and push rumor spreading in the noisy setting. Reaching consensus to an opinion in population protocols in the noiseless setting has received much attention (see *e.g.,* [4, 22, 1, 2, 5, 6, 19, 7, 38, 29, 28, 35]).

Noise and failures in sensor networks. The problem of converging to the average (and similar problems) have also been studied in (noisy) sensor networks [56, 50] where nodes again can interact with all their neighbors. In these networks another type of unreliable communication, i.e., packages might be dropped, has received ample attention, e.g., [12] studies the broadcast problem and [13] develops a framework to transform certain algorithms for failure free networks to also work in faulty sensor networks.

An interesting type of failure has been studied in [32]. There failures do not happen during the communication but the algorithm itself might be faulty, i.e., a state machine run at an agent might switch to a wrong state.

1.2 Formal Results

We now formally state our main theorems. For the ease of presentation, in the discussion we assume that noise is normally distributed with unit variance, $N \sim \mathcal{N}(0,1)$, but our results hold for general variance σ^2. Let $\phi_0 = \bar{\phi}(\mathbf{X}^{(0)})$ be the initial potential. Our first

theorem shows that the agents converge to a small value of $\bar{\phi}(t) = O(n)$ after parallel time[2] that is logarithmic in ϕ_0/n. In particular, if we use b to denote the initial imbalance ($b = \max_{i,j}\{x_i^{(0)} - x_j^{(0)}\}$), then it takes $O(\ln b)$ parallel steps for the potential to become $\bar{\phi}(t) = O(n)$. Note that $\bar{\phi}(t) = O(n)$ means that the "average" difference between the values of any two agents is constant and we show that the constant hidden in the O-notation is actually very small. It is worth mentioning that this is tight in two senses: (i) In expectancy we have $\bar{\phi}(t) = \Omega(n)$ for any fixed time step $t \geq n$, (i.e., after one parallel time step). Even in the case where all nodes initially have the same value, our results show that the potential increases after n interactions in expectation by $\Omega(n\mathbb{E}\left[N^2\right]) = \Omega(n)$. (ii) At least $\Omega(\ln b)$ parallel time steps are required[3] to decrease the potential to $O(n)$, since the potential only drops in expectation by a constant factor in each parallel step. The formal statement is as follows.

▶ **Theorem 1** (Convergence to Running Avg.). *Consider any noise-distribution $\aleph$ with (at least) exponential-decay[4]. Fix any $\delta \in \mathbb{R}$. Let $n = n(\delta)$ be large enough. The following hold:*

- **(i)** *for any $t = \Omega\left(n \ln\left(\frac{\phi_0}{\delta\sigma^2 n}\right)\right)$ with probability at least $1-\delta$ we have $\bar{\phi}(\mathbf{X}^{(t)}) = O(\sigma^2 n \ln(1/\delta))$,*
- **(ii)** *for any $t \geq n$ (parallel time) with constant probability we have $\bar{\phi}(\mathbf{X}^{(t)}) = \Omega(\sigma^2 n)$ and*
- **(iii)** *even without noise, for any $t = o\left(n \ln\left(\frac{\phi_0}{\sigma^2 n}\right)\right)$ we have $\mathbb{E}\left[\bar{\phi}(\mathbf{X}^{(t)})\right] = \omega(\sigma^2 n)$.*

While the above theorem shows a quick convergence to the running average, this does not imply convergence to the initial average. In fact, as time progresses the distance to the initial average ($TSS(\mathbf{X}^{(t)})$) is likely to increase. Nonetheless, in the case of the Gaussian white noise model we can bound the drift of the running average from the initial average in a time window of $O(n^2)$ steps (see Lemma 17). Theorem 1 roughly says that after at least $t = \Omega(n \log n)$ steps the distance to the running average is small if we start with a potential that is polynomial in n. Thus, as long as $t = \Omega(n \log n)$ and $t = O(n^2)$ we obtain $TSS(\mathbf{X}^{(t)}) = O(n)$. After the $O(n^2)$ step time window the potential starts to increase again, which, is unavoidable, due to the noise causing drift of the running average; in Gaussian white noise model, the running average after t steps diverges with constant probability from the initial average by $\frac{\sqrt{t}}{n}$ (Lemma 17). This in turn implies that $TSS(\mathbf{X}^{(t)}) \geq t/n$.

▶ **Corollary 2** ((Bounded) Divergence from Initial Avg.). *In the case of Gaussian white noise model, for any $\delta \in \mathbb{R}$ and large enough $n = n(\delta)$ and all $t = \Omega\left(n \ln\left(\frac{\bar{\phi}(\mathbf{X}^{(0)})}{\delta\sigma^2 n}\right)\right)$ we have*

- **(i)** *"non-divergence for $O(n^2)$ steps", i.e., $TSS(\mathbf{X}^{(t)}) = O\left(\left(\frac{t}{n} + n\right)\sigma^2 \ln(1/\delta)\right)$ with probability at least $1-\delta$ and*
- **(ii)** *"divergence for $\omega(n^2)$ steps", i.e., $TSS(\mathbf{X}^{(t)}) = \Omega\left(\left(\frac{t}{n} + n\right)\sigma^2\right)$ with constant probability.*

If one can bound the divergence between the running average and the initial average for a general noise-distribution $\aleph$ with (at least) exponential-decay[5] the following remark is useful to obtain a similar bound for the $TSS(\mathbf{X}^{(t)})$ as in Corollary 2. Recall that $\varnothing^{(t)} = \sum_i X_i^{(t)}/n$ and in particular, $\varnothing^{(0)}$ denotes the initial average.

[2] Recall that in parallel time we scale time by a factor of n for a fair comparison with the synchronous time model.
[3] For the case where constant fraction of the values are at distance b.
[4] In fact we only require the function to be smooth, which we define later. This class is much broader and contains most of the famous distributions including the normal distribution, geometric distribution and the Poisson distribution.
[5] Again, we only require the function to be smooth, which we define in Section 3.

▶ Remark 2. Fix any $\delta \in \mathbb{R}$. Let $n = n(\delta)$ be large enough. For any fixed $t = \Omega\left(n \ln\left(\frac{\phi_0}{\delta\sigma^2 n}\right)\right)$ with probability at least $1-\delta$ we have $TSS(\mathbf{X}^{(t)}) = \Theta\left(n\left(\varnothing^{(t)} - \varnothing^{(0)}\right)^2 + \sigma^2 n \ln(1/\delta)\right)$.

Remark 2 follows by rewriting $TSS(t) = \bar{\phi}(\mathbf{X}^{(t)}) + n \cdot \left(\varnothing^{(0)} - \varnothing^{(t)}\right)^2$ (cf. Fact 9) and plugging in the first part of Theorem 1. Corollary 2 then follows by plugging in the bounded deviation of the running average from the initial average for the Gaussian white noise model (cf. Lemma 17).

The Influence of Rounding. Agents with limited computational power might not be able to store real values. Motivated by this we also consider the setting where agents can only store integers. In particular, we consider the case that the averaging protocol is augmented with the following rounding procedure: Assume that the noise $N \sim \aleph$ takes only integer variables. After a node i receives the value from node j, the node averages it as before and then rounds up or down with equal probability. In the full version we show how to relate the setting of rounding to the original setting allowing us to derive the following corollary.

▶ **Corollary 3.** *The bounds of Theorem 1 and Corollary 2 hold even if rounding is used.*

The Synchronous Model. In the full version, we show how our results extend to the synchronous setting. It turns out that the results are the same up to a rescaling of time.

▶ **Corollary 4** (Synchronous Setting)**.** *The bounds of Theorem 1 and Corollary 2 hold even in the synchronous setting, where time is rescaled by a factor of* $2/n$.

Experimental Results. In Section 5, we simulate the averaging dynamic in various settings. In the first setting, we consider the distribution of the distances between agents' values and the running average. Our simulations show that these distances seem to follow an exponential law, i.e., the concentration is even stronger than what Theorem 1 implies.

Due to the limited memory of agents it would be desirable to obtain similar results as in Theorem 1 for the averaging dynamic in the setting where agents can only store values from a bounded range. However, our simulations in Section 5 show that this setting leads to a much faster (by order of magnitude) divergence between the running average and the initial average.

1.3 Technical Contributions

While it is not hard to show that in expectation the potentials $TSS(t)$ and $\bar{\phi}(t)$ decrease in one step as long as their value is large, it is surprisingly challenging to derive probabilistic bounds on either potential at an arbitrary point in time, *i.e.,* bounds of the type $\mathbb{P}\left[\bar{\phi}(t) \geq b\right] \leq p(b)$. Two of the reasons are as follows. (i) The potential decreases (expectedly) only conditioned on the fact that it is large enough. In fact, when the potential is small, then due to the noise it will increase in expectation. (ii) Since we study general distributions and in particular the normal distribution, the noise in a given round can be arbitrarily large leading to an arbitrarily large increase in $\bar{\phi}(t)$; if the protocol runs long enough (possibly exponentially long in n) we, indeed, will have encountered some time steps with a very large potential increase. There are surprisingly few analytical tools for using potentials as $\bar{\phi}(t)$ with challenges (i) and (ii). One notable exception is Hajek's theorem [33], which can be used to bound the value of

such a potential at a given time t. However, in our setting – with our potential function – the results obtained are very weak.[6]

Instead, we use a more sophisticated approach that at its core has a decomposition of the potential change in a single time step into three additive (but dependent) random variables. We iterate this decomposition over time throughout some interval $\mathcal{I} = (t_0, t_1]$ and sum the respective variables which we will denote as $S^-(\mathcal{I})$, $S'(\mathcal{I})$, and $S^*(\mathcal{I})$. Then (cf. Proposition 12) we are able to bound the potential change at the end of the interval as

$$\bar{\phi}(\mathbf{X}^{(t_1)}) \leq \left(1 - \frac{S^-(\mathcal{I})}{t_1 - t_0}\right)^{t_1 - t_0} \cdot \bar{\phi}(\mathbf{X}^{(t_0)}) + S'(\mathcal{I}) + S^*(\mathcal{I}). \tag{1}$$

Due to the dependencies between the three variables we use strong Martingale concentration bounds to separately upper bound $S'(\mathcal{I}) + S^*(\mathcal{I})$ and lower bound $S^-(\mathcal{I})$ (cf. Lemma 14). We then use a union bound – to circumvent the dependencies – to bound each of these variables allowing us to get a bound on Equation 1. It is critical that we define the random variable S^- in such a way that it always has an expected decrease. This is in stark contrast to the entire potential, which, as we mentioned before in (i), only decreases in expectation when it is large. Having an unconditional decrease of S^- allows us to consider arbitrarily large intervals. With these bounds at hand one can use Equation 1 to obtain probabilistic bounds on the potential at any given point time t_1. However, due to the bound on $S'(\mathcal{I}) + S^*(\mathcal{I})$ the total bound becomes very weak for large intervals. As a remedy, we carefully trace the change in the potential in different regimes (with several phases in each regime) and we separately apply the aforementioned analysis with a fresh (small) interval in each phase. The intervals (and thus also the phases) have variable length – decreasing geometrically or even exponentially, depending on the regime.

All missing proofs can be found in the full version [43].

2 Model

In this section we present the model including all assumptions. We have a collection of n agents that have initial values $X_1^{(0)}, X_2^{(0)}, \ldots, X_n^{(0)}$. Time is discrete and $X_i^{(t)}$ denotes the value of agent $i \in [n]$ at time t. Recall that $\varnothing^{(t)} = \sum_i X_i^{(t)}/n$ denotes the average value at time t; in particular, $\varnothing^{(0)}$ denotes the initial average. For two random variables X and Y we write $X \stackrel{d}{=} Y$ if they have the same (probability) distribution. Next, we define the communication models.

▶ **Definition 5** (Communication Models)**.** We consider two communication models.

- **(i)** *Sequential model*: At every discrete time step two of the agents i, j are chosen uniformly at random (with replacement[7]) and send their current values x_i and x_j to each other, where the values received are $x_i + N_i$ and $x_j + N_j$, where $N_i, N_j \stackrel{d}{=} N$.
- **(ii)** *Synchronous model*: At every discrete time step a perfect matching is chosen u.a.r. among all perfect matchings on the n agents[8]. All matched agents interchange their

[6] Hajek's theorem considers the moment generating function of the potential. In order to apply the theorem to our potential, it seems that one would need to consider a logarithmic version of the potential, which together with the moment generating function results in bound that is weaker than a simple union bound.

[7] This is not crucial to our results, but simplifies the calculations slightly.

[8] Again, we allow matchings of the kind (i, i) for simplicity. It is easy but slightly less aesthetic to modify our results to exclude matchings (i, i).

values as in the sequential model.

We use the *parallel time*, which was first defined in [3], to denote the time step t/n in the sequential model. This notion eases the comparison of results in both models, as the total number of interactions is up to a factor of 2 equal.

▶ **Definition 6** (Noise Models)**.** Let v be the value sent by an agent. The value received is $v + N$, where N is distributed according to some zero-mean noise distribution $\aleph$ with $\sigma^2 = \mathrm{Var}[N]$.

We consider general noise distributions and our results depend on the moments of N. The following two models are of special interest in this paper.

(i) *Gaussian white noise model* where $\aleph = \mathcal{N}(0, \sigma^2)$ for an arbitrary σ.

(ii) *Discrete white noise model* where $\aleph = \mathcal{D}(p)$, with $\mathbb{P}[N = i] = \frac{1}{2}p(1-p)^{|i|}$, for $i \in \mathbb{Z}\setminus\{0\}$ and $\mathbb{P}[N = 0] = p$, where $p \in (0, 1]$. Note that $\mathrm{Var}[N] = \frac{1-p}{p^2}$.

From now on we assume that the noise N is distributed according to a fixed noise distribution $\aleph$ that is independent of n.

▶ **Definition 7** (Averaging Dynamic)**.** We consider the real valued and the discrete valued algorithm. A node with value v at time receiving the input w sets its new value to

(i) $v' = {}^{(v+w)}/_{2}$ in the *real valued model*.

(ii) $v' = \begin{cases} \lceil {}^{(v+w)}/_{2} \rceil & \text{w.p. } \frac{1}{2} \\ \lfloor {}^{(v+w)}/_{2} \rfloor & \text{otherwise} \end{cases}$ in the *discrete valued model*.

A probability distribution $\mathcal{D}$ is called *sub-Gaussian* if for $X \sim \mathcal{D}$ we have that there exists positive constants c_1, c_2 such that for every x we have $\mathbb{P}[|X| \geq x] \leq c_1 \exp(-c_2 x^2)$.

Whenever we calculate the new values $\mathbf{X}^{(t+1)}$ by conditioning on the current state, $\mathbf{X}^{(t)} = \mathbf{x}^{(t)}$ we use small letters $x_i^{(t)}$ to denote fixed values and capitalized letters $X_i^{(t+1)}$ to denote random variables. Furthermore, we use bold-face to denote vectors. Throughout the paper we will assume that the number of agents n is large enough and in particular $n\mathbb{E}\left[N^2\right] \geq 1$.

We define the following potentials which are essential in all our proofs and formal results.

▶ **Definition 8** (Potentials)**.**

$$TSS(\mathbf{x}^{(t)}) = \sum_i \left(x_i^{(t)} - \varnothing^{(0)}\right)^2, \quad \bar{\phi}(\mathbf{x}^{(t)}) = \sum_i \left(x_i^{(t)} - \varnothing^{(t)}\right)^2, \quad \phi(\mathbf{x}^{(t)}) = \sum_{i,j} \left(x_i^{(t)} - x_j^{(t)}\right)^2.$$

When clear from the context we drop the time index t and we write $\mathbf{x}$ instead of $\mathbf{x}^{(t)}$, x_i instead of $x_i^{(t)}$, etc. Similarly we will use the following short forms $TSS(t) = TSS(\mathbf{x}^{(t)})$ and $\bar{\phi}(t) = \bar{\phi}(\mathbf{x}^{(t)})$. We emphasize that the difference between $\bar{\phi}(\mathbf{x})$ and $TSS(t)$ is that the former measures the squared distance w.r.t. the *running average* and the latter w.r.t. *initial average*. Initially, we have $\bar{\phi}(\mathbf{x}^{(0)}) = TSS(0)$. In [43] we prove the following fact that shows how $\bar{\phi}(\mathbf{X}^{(t)})$ relates to $TSS(t)$ and how $\bar{\phi}$ relates to ϕ.

▶ **Fact 9.** *We have that*

1. $TSS(t) = \bar{\phi}(\mathbf{X}^{(t)}) + n \cdot \left(\varnothing^{(0)} - \varnothing^{(t)}\right)^2$ *and*
2. $\phi(\mathbf{x}) = 2n \cdot \bar{\phi}(\mathbf{x})$.

Note that many alternative ways to define the potential at a time t such as the max distance and $\ell 1$ norm give only a very partial picture: The max distance to the mean for example does not distinguish between just one node being far and all nodes being far. On the other hand, the $\ell 1$ norm does not "punish" outliers enough: there is no difference between n nodes being off by 1 from the average and one node being off by n.

Notation

We use $X \sim \mathcal{D}$ to denote that X is distributed according to probability distribution $\mathcal{D}$. For two random variables X and Y we write $X \leq^{\text{st}} Y$ if X is *stochastically dominated* by Y, i.e., $\mathbb{P}[X \geq x] \leq \mathbb{P}[Y \geq x]$ for all $x \in \mathbb{R}$. We use $\|\mathbf{x}\|_2$ to denote the $L2$-norm. In the sequential model we have two random variables $N_1^{(t)}$ and $N_2^{(t)}$ for the noise of the channel at time step t (recall that $N_1^{(t)}$ and $N_2^{(t)}$ are distributed according to $\aleph$). We define the following two random variables $N'^{(t)}$ and $N^{*(t)}$ that will play a key role in our analysis:

$$N'^{(t)} = \left(N_1^{(t)}\right)^2 + \left(N_2^{(t)}\right)^2, \qquad N^{*(t)} = N_1^{(t)} + N_2^{(t)} .$$

▶ **Fact 10.** *In the Gaussian noise model, we have* $N^{*(t)} \sim \mathcal{N}(0, 2\sigma^2)$ *and* $N'^{(t)} \sim \Gamma(1, 2\sigma^2)$, *where* $\Gamma(\cdot,\cdot)$ *denotes the gamma distribution.*

When clear from the context we simply write N' and N^* instead of $N'^{(t)}$ and $N^{*(t)}$, respectively. We use $\mathcal{F}_t$ to denote the filtration at time t, which encapsulates all randomness up to time t as well as the initial values of the nodes; hence it defines the state at time t completely.

3 The Sequential Setting: Convergence towards the Running Average

Conditioning on all the randomness until time t, *i.e.*, conditioning on $\mathcal{F}_t$, we define

$$\Delta^{(t+1)} = \begin{cases} \frac{\left(x_i^{(t)} - x_j^{(t)}\right)^2}{2\bar{\phi}(\mathbf{x}^{(t)})} & \text{for } \bar{\phi}(\mathbf{x}^{(t)}) > 0 \\ 1/n & \text{otherwise} \end{cases}, \text{ where } i \text{ and } j \text{ are the chosen in round } t.$$

▶ **Lemma 11** (One Step Bound). *Fix an arbitrary potential at time t. Suppose the pair i, j was chosen to communicate and condition on the filtration $\mathcal{F}_t$ (all events that happened up to round t). Then, the following holds*

$$\bar{\phi}(\mathbf{X}^{(t+1)}) - \bar{\phi}(\mathbf{x}^{(t)}) \leq -\Delta^{(t+1)}\bar{\phi}(\mathbf{x}^{(t)}) + \frac{N'^{(t+1)}}{4} + N^{*(t+1)}\left(\frac{x_i^{(t)} + x_j^{(t)}}{2} - \varnothing^{(t)}\right) .$$

Further we have $\mathbb{E}\left[\Delta^{(t+1)} \mid \mathcal{F}_t\right] = \frac{1}{n}$.

In order to prove the statement, we first calculate the exact expected change in one step (which we do in the full version). We then majorize (stochastic dominance) with the slightly more convenient statement above.

For an arbitrary time interval $\mathcal{I}$ define

$$S'(\mathcal{I}) = \sum_{\tau \in \mathcal{I}} N'^{(\tau)}/4, \quad S^*(\mathcal{I}) = \sum_{\tau \in \mathcal{I}} N^{*(\tau)}\left(\frac{x_i^{(\tau-1)} + x_j^{(\tau-1)}}{2} - \varnothing^{(\tau)}\right), \quad S^-(\mathcal{I}) = \sum_{\tau \in \mathcal{I}} \Delta^{(\tau)} .$$

Note that, in the definition of S^*, we sum up over all time steps τ in the interval $\mathcal{I}$ and we consider the pair i and j that is chosen in round τ (in each round a different pair i and j can be chosen). With Lemma 11 and the definitions of S', S^* and S^- we can deduce the following decomposed bound on the potential for an arbitrary interval.

▶ **Proposition 12** (Decomposition of Potential). *Fix arbitrary t_0, t_1 and consider the interval $\mathcal{I} = (t_0, t_1]$. For $t = t_1 - t_0$ we have that*

$$\bar{\phi}(\mathbf{X}^{(t_1)}) \leq \left(1 - \frac{S^-(\mathcal{I})}{t}\right)^t \bar{\phi}(\mathbf{X}^{(t_0)}) + S'(\mathcal{I}) + S^*(\mathcal{I}). \tag{1}$$

Our results only hold for smooth noise distributions, which we define in the following. Let

$$m_{t,\delta} = \arg\max_{\ell} \left\{ \mathbb{P}\left[\max\left(\left\{ N'^{(t_0)}, \ldots, N'^{(t_0+t)} \right\} \cup \left\{ N^{*(t_0)}, \ldots, N^{*(t_0+t)} \right\} \right) \leq \ell \right] \geq 1-\delta \right\}.$$

▶ **Definition 13.** A noise distribution $\aleph$ is *smooth* if for all $\delta > 0$ and all $t > 0$ we have $m_{t,\delta} \leq \left(\frac{t}{\delta}\right)^{1/20}$.

Any (sub-)linear probability distribution and even some inverse polynomial distributions are smooth. Thus many practically relevant distributions such as Gaussian, binomial and Poisson distributions are smooth. For example, for the standard normal distribution ($N \sim \mathcal{N}(0,1)$) we have $m_{t,\delta} = \log(t/\delta)$, since in each time step the probability that the N^2 exceeds $\log(t/\delta)$ is equal to the probability that N exceeds $\sqrt{\log(t/\delta)}$ which happens w.p. at most δ/t. Taking union bound over all t steps shows that it is smooth.

Using strong martingale concentration bounds (see the full version for details) and bounding the variance, we deduce the following upper bound on $S^* + S'$ and lower bound on S^-.

▶ **Lemma 14.** *Let t_0, t_1 be such that $t_1 > t_0$ and consider the interval $\mathcal{I} = (t_0, t_1]$.*

(i) *With probability $1-\delta$ we have*

$$S^*(\mathcal{I}) + S'(\mathcal{I}) \leq \frac{t}{4}\mathbb{E}\left[N'\right] + 5\sqrt{\frac{t}{n}}\left(\ln(4t/\delta) m^*_{t,\delta/4}\right)^2 \left(2 + \mathbb{E}\left[N'\right]\right)\sqrt{\bar{\phi}(\mathbf{x}^{(t_0)}) + 9t\mathbb{E}\left[N'\right] + 2}\,.$$

(ii) *For any $\gamma < 1$, w.p. at least $1 - \exp\left(-\frac{3\gamma^2 t}{8n}\right)$ we have $S^-(\mathcal{I}) \geq (1-\gamma)\frac{t}{n}$.*

The following proposition almost directly implies Theorem 1.

▶ **Proposition 15.** *Fix any $\delta \in (0,1]$ and assume that the noise distribution is smooth. There exists a constant c such that for a time step t_0 with potential $\bar{\phi}(\mathbf{x}^{(t_0)})$ we have*

$$\mathbb{P}\left[\bar{\phi}(\mathbf{X}^{(t^*)}) \geq \ln(1/\delta) n \mathbb{E}\left[N'\right] + b \mid \mathcal{F}_{t_0} \right] \leq \delta,$$

where $t^* = t_0 + cn\ln\left(\frac{\bar{\phi}(\mathbf{x}^{(t_0)})}{\mathbb{E}[N']n\delta}\right)$ and $b = 2\left(1 + \mathbb{E}\left[N'\right]\right)\left(\ln(1/\delta)\right)^9 n^{9/10}$.

Proof Sketch. We only sketch the proof idea for a simplified setting; during the sketch we assume that $N \sim \mathcal{N}(0,1)$ (with $\mathbb{E}\left[N'\right] = O(1)$) and also that δ is at least $1/n^3$. The main ingredients for the proof are Proposition 12 and Lemma 14. For an interval $\mathcal{I} = (t_0, t_1]$ Proposition 12 upper bounds the potential at time t_1 by

$$\bar{\phi}(\mathbf{X}^{(t_1)}) \leq \left(1 - \frac{S^-(\mathcal{I})}{t}\right)^t \bar{\phi}(\mathbf{X}^{(t_0)}) + S'(\mathcal{I}) + S^*(\mathcal{I}), \tag{2}$$

where t is the length of the interval. Lemma 14 lower bounds $S^-(\mathcal{I})$ and upper bounds the sum $S'(\mathcal{I}) + S^*(\mathcal{I})$. To prove Proposition 15 we have to show that the initial potential $\bar{\phi}(\mathbf{x}^{(t_0)})$ decreases to $O(n)$ after $O(n \cdot \log \bar{\phi}(\mathbf{x}^{(t_0)}))$ time steps with probability $1-\delta$. Optimally, we would use a single application of Proposition 12 to upper bound the potential as in Equation 2 and then bound the terms $S^-(\mathcal{I})$ and $S'(\mathcal{I}) + S^*(\mathcal{I})$ via Lemma 14. However, the bounds on S^- and $S' + S^*$ given by Lemma 14 are too loose to yield the desired result via a single application of Proposition 12 and Lemma 14 with the whole time interval $\mathcal{I} = [t_0, t_0 + O(n \log \bar{\phi}(\mathbf{x}^{(t_0)}))]$. For example, the bound on $S' + S^*$ inherently has a term

of order $\sqrt{\bar{\phi}}$, where $\bar{\phi}$ is the potential at the start of the interval for which Lemma 14, i is applied. Thus a one shot proof as described above can never reach a potential below $\sqrt{\bar{\phi}}$. This is not sufficient if the initial potential is large, e.g., say for $\bar{\phi} \gg n^{8/3}$.

To circumvent this problem we apply Proposition 12 and Lemma 14 several times for smaller time intervals: More detailed, we split the proof of Proposition 15 into two regimes. In regime 2 we use several phases to decrease the potential to $\Theta(n^{4/3})$. If the potential is $\bar{\phi}$ at the beginning of a phase a single application of Proposition 12 and Lemma 14 reduces the potential to $\bar{\phi}^{3/4}$. The length of each such phase is geometrically decreasing by a factor 3/4 where the first phase is of length $O\left(n \ln\left(\frac{\bar{\phi}(\mathbf{x}^{(t_0)})}{n\delta}\right)\right)$. After the last phase of regime 2 the potential is of order $n^{4/3}$.

Then, in regime 1 the potential reduces from $\Theta(n^{4/3})$ to $O(n)$, again through several phases. If the first phase of regime 1 starts with a potential of size B, the phase has length $t = O(n \ln(B))$. If there was no additive increase due to the noise, then this would reduce the potential to 0. However, there is an additive increase of $\Theta(t) = \Theta(n \ln(B))$ which leaves us with a potential of size $O(n \ln(B))$. The next phase will therefore be of length $n \ln \ln(B)$ etc. This is repeated for $\ln^*(B)$ phases until the potential reduces to $O(n)$, which, as we explained in Subsection 1.2, is the furthest the potential can be decreased .

Putting everything together, we get that after $O\left(n \ln\left(\frac{\bar{\phi}(\mathbf{x}^{(t_0)})}{n\delta}\right)\right)$ rounds the potential reduces to $O(n)$. ◀

The full proof of Proposition 15 handles general $\mathbb{E}[N']$ and general δ and thus it is significantly more technical. It can be found in the full version. From Proposition 15 we are able to derive Theorem 1.

4 Deviation from the Initial Average

An informal argument for the statements in this section in the special case of $\sigma = 1$ can be found in [54]. Before we state our results we need the following result on the standard normal distribution.

▶ **Theorem 16** ([16]). *Let $\Phi(x)$ denote the cumulative distribution function of the standard normal distribution. We have for $x \geq 0$:*

$$\frac{1}{\sqrt{2\pi}} \frac{x}{x^2+1} \exp\left(-x^2/2\right) \leq \Phi(x) \leq \frac{1}{\sqrt{2\pi}} \frac{1}{x} \exp\left(-x^2/2\right).$$

We can now state and prove the main results of this section.

▶ **Lemma 17.** *For any t and any $\delta < 1$, we have $\varnothing^{(t)} - \varnothing^{(0)} \sim \frac{\sum_{\tau=1}^{2t} N^{(\tau)}}{2n}$ with probability at least $1-\delta$, where $N^{(\tau)}$ is the noise of the channel. In particular, for the Gaussian white noise model setting where $N \sim \mathcal{N}(0, \sigma^2)$ we have $\sum_{\tau=1}^{2t} N^{(\tau)} \sim \mathcal{N}(0, 2t\sigma^2)$. Thus*

(i) $|\varnothing^{(t)} - \varnothing^{(0)}| \leq \frac{\sigma\sqrt{t \ln(1/\delta)}}{n}$ *w.p. at least* $1-\delta$

(ii) $|\varnothing^{(t)} - \varnothing^{(0)}| \geq \frac{\sigma\sqrt{t \ln(1/\delta)}}{n}$ *w.p. at least* $\frac{\delta}{2\sqrt{2\ln(1/\delta)}}$.

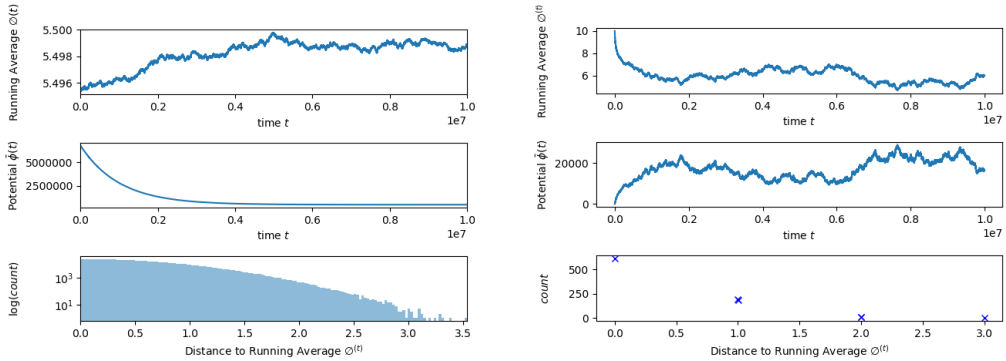
Using the Berry-Esseen theorem, one can easily prove similar bounds for any distribution with bounded third moment including *discrete white noise*.

In the following we consider the potential $(\varnothing_t)_{t\geq 0}$ as a Martingale to derive the desired concentration bounds. The following bound is weaker than the aforementioned bounds, however, it is useful whenever the noise is such that $m_{t,\delta/(2t)}$ is small.

▶ **Proposition 18.** *For any $t \geq 2$ and any $\delta < 1$, we have $-m_{t,\delta/(2t)}\sigma\sqrt{2t} \leq \varnothing^{(t)} - \varnothing^{(0)} \leq m_{t,\delta/(2t)}\sigma\sqrt{2t}$ with probability at least $1-\delta$.*

5 Experimental Results

The goal of this section is twofold. First, we seek to better understand the distribution $\mathcal{D}$ of the distances $x_i^{(t)} - \varnothing^{(t)}$. Second, we simulate a setting in which the range of values is bounded, motivated by computational and storage limited agents. All results in this section are based on an implementation of the simple averaging dynamic. The code (python3) for the experiments can be found here [42].



(a) The setting of this example is: $n = 10^6$, initial distribution of values is uniformly at random in the range $[1, n^2]$, $10n$ iterations, Gaussian white noise with variance 1, unbounded range.

(b) The setting of this example is: $n = 1000$, all values equal to 10, using discrete white noise model $\mathcal{D}(0.8)$ (see Definition 6), bounded range in the interval $[1, 10]$, $10^4 n$ iterations. The avg. of the values drifts from 10 to 6.

Figure 1 The figure depicts the distribution of distances as well as the bounded value setting.

5.1 The Distribution of the Distances

The experiments suggest that the distance decays at least exponentially. Note that the experiments only show a single iteration, however, this phenomena was observable in every single run. The bound on $\mathbb{E}\left[\bar{\phi}\left(\mathbf{X}^{(t)}\right)\right]$ we obtained in Theorem 1 only implies that $\mathcal{D}$ is at most $O(1/d^3)$. However, we conjecture, for sub-Gaussian noise that $\mathbb{P}\left[|X_i^{(t)} - \varnothing^{(t)}| \geq x\right] = O(\exp^{-x})$ (cf. Figure 1a). Showing this rigorously is challenging due to the dependencies among the values. Nonetheless, such bounds are very important since they immediately bound the maximum difference and we consider this the most important open question.

5.2 The Bounded Values Setting

One of the motivations for the very simple averaging dynamic arises in the setting of limited computational power of the interacting agents. So far we assumed that agents can store and transmit (intermediate) values from an unbounded range. For many applications and in particular motivated by agents with bounded memory one would hope for similar results if there is a maximum and a minimum value that can be stored or transmitted. The formal definition is as follows: values can only be from the range $[v_{min}, v_{max}]$ ($= [1, 10]$ in our experiments). We assume noise of the channel cannot produce values larger than v_{max} or smaller than v_{min}, which can be motivated as follows in the setting where the values correspond to amplitudes: here v_{max} and v_{min} are simply the amplitudes (high amplitude and no amplitude) where the signal-to-noise ratio is very large, and noise becomes negligible. An equivalent model is that the agents know the range of possible communication values,

and hence, they can simply correct every value larger than v_{max} to v_{max}. In particular when agents only have limited storage, the communication range will often be bounded, and even rounding might become necessary (see the full version).

We refer to these equivalent models as the model with *cutoffs*. While the experiments indicate that values still converge towards the running average, there is a clear drift of the running average from the initial average if the input values are chosen unsuitably. In our experiments, we set the range of values to $[1, 10]$, use the noise described in the discrete noise model together with rounding. Initially, all agents have value 10. We see a drastic drift of the running average (see Figure 1b). Even though the initial average is 10, the running average appears to approach the midpoint of the range, i.e., 5. The histogram of distances to the initial average shows even more clearly that the values are not concentrated around the initial average. Although the experiments only show a single iteration, this phenomena was observable in every single run. We believe that the reason for this is simply that the noise is no longer symmetric and no longer zero-mean due to the cutoffs $[1, 10]$. Proving convergence to the running-average in this model seems challenging and interesting.

We believe that the insights in bounding this potential might be useful in similar problems.

6 Conclusion and Open Problems

In this paper we showed bounds on the convergence time for the unbounded setting. Our simulations in Section 5 yield two interesting open problems: (i) study the setting where the values are restricted to some interval (in this case the noise is no longer symmetrical) and (ii) prove tail bounds on the distance distribution w.r.t. to the running or initial average. Another interesting research direction is to move away from zero-mean noise and consider biased noise models: how quickly can the bias(es) be estimated and is convergence still feasible by compensating for the (learned) bias?

References

1. Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L. Rivest. Time-Space Trade-offs in Population Protocols. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2560–2579, 2017. doi:10.1137/1.9781611974782.169.
2. Dan Alistarh, James Aspnes, and Rati Gelashvili. Space-Optimal Majority in Population Protocols. *CoRR*, abs/1704.04947, 2017. arXiv:1704.04947.
3. Dan Alistarh, Rati Gelashvili, and Milan Vojnović. Fast and Exact Majority in Population Protocols. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, PODC '15, pages 47–56, New York, NY, USA, 2015. ACM. doi:10.1145/2767386.2767429.
4. Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, 2006. doi:10.1007/s00446-005-0138-3.
5. Luca Becchetti, Andrea E. F. Clementi, Emanuele Natale, Francesco Pasquale, Riccardo Silvestri, and Luca Trevisan. Simple dynamics for plurality consensus. *Distributed Computing*, 30(4):293–306, 2017. doi:10.1007/s00446-016-0289-4.
6. Petra Berenbrink, Andrea E. F. Clementi, Robert Elsässer, Peter Kling, Frederik Mallmann-Trenn, and Emanuele Natale. Ignore or Comply?: On Breaking Symmetry in Consensus. In *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*, pages 335–344, 2017. doi:10.1145/3087801.3087817.

7 Petra Berenbrink, Robert Elsässer, Tom Friedetzky, Dominik Kaaser, Peter Kling, and Tomasz Radzik. A Population Protocol for Exact Majority with O(log5/3 n) Stabilization Time and Theta(log n) States. In *32nd International Symposium on Distributed Computing, DISC 2018, New Orleans, LA, USA, October 15-19, 2018*, pages 10:1–10:18, 2018. doi:10.4230/LIPIcs.DISC.2018.10.

8 Lucas Boczkowski, Ofer Feinerman, Amos Korman, and Emanuele Natale. Limits for Rumor Spreading in Stochastic Populations. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, pages 49:1–49:21, 2018. doi:10.4230/LIPIcs.ITCS.2018.49.

9 J. E. Boillat. Load Balancing and Poisson Equation in a Graph. *Concurrency: Pract. Exper.*, 2(4):289–313, November 1990. doi:10.1002/cpe.4330020403.

10 Henrik Brumm. *Animal communication and noise*, volume 2. Springer, 2013.

11 Jingjing Bu, Maryam Fazel, and Mehran Mesbahi. Accelerated Consensus with Linear Rate of Convergence. In *2018 Annual American Control Conference (ACC)*, pages 4931–4936. IEEE, 2018.

12 Keren Censor-Hillel, Bernhard Haeupler, D. Ellis Hershkowitz, and Goran Zuzic. Broadcasting in Noisy Radio Networks. In *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*, pages 33–42, 2017. doi:10.1145/3087801.3087808.

13 Keren Censor-Hillel, Bernhard Haeupler, D. Ellis Hershkowitz, and Goran Zuzic. Erasure Correction for Noisy Radio Networks. *CoRR*, abs/1805.04165, 2018. arXiv:1805.04165.

14 Biao Chen, Ruixiang Jiang, Teerasit Kasetkasem, and Pramod K Varshney. Channel aware decision fusion in wireless sensor networks. *IEEE Transactions on Signal Processing*, 52(12):3454–3458, 2004.

15 Ge Chen, Le Yi Wang, Chen Chen, and George Yin. Critical connectivity and fastest convergence rates of distributed consensus with switching topologies and additive noises. *IEEE Transactions on Automatic Control*, 62(12):6152–6167, 2017.

16 John Cook. Upper and lower bounds for the normal distribution function. https://www.johndcook.com/blog/norm-dist-bounds/, 2018. [Online; accessed 6-September-2018].

17 G. Cybenko. Dynamic Load Balancing for Distributed Memory Multiprocessors. *J. Parallel Distrib. Comput.*, 7(2):279–301, October 1989. doi:10.1016/0743-7315(89)90021-X.

18 Morris H DeGroot. Reaching a consensus. *Journal of the American Statistical Association*, 69(345):118–121, 1974.

19 David Doty and David Soloveichik. Stable leader election in population protocols requires linear time. *Distributed Computing*, 31(4):257–271, August 2018. doi:10.1007/s00446-016-0281-z.

20 Julia T. Ebert, Melvin Gauci, and Radhika Nagpal. Multi-Feature Collective Decision Making in Robot Swarms. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, pages 1711–1719, 2018. URL: http://dl.acm.org/citation.cfm?id=3237953.

21 Julia T. Ebert, Melvin Gauci, and Radhika Nagpal. Multi-Feature Collective Decision Making in Robot Swarms. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, pages 1711–1719, 2018. URL: http://dl.acm.org/citation.cfm?id=3237953.

22 Robert Elsässer, Tom Friedetzky, Dominik Kaaser, Frederik Mallmann-Trenn, and Horst Trinker. Efficient k-Party Voting with Two Choices. *CoRR*, abs/1602.04667, 2016. arXiv:1602.04667.

23 J Alexander Fax and Richard M Murray. Information flow and cooperative control of vehicle formations. *IEEE transactions on automatic control*, 49(9):1465–1476, 2004.

24 Ofer Feinerman, Bernhard Haeupler, and Amos Korman. Breathe Before Speaking: Efficient Information Dissemination Despite Noisy, Limited and Anonymous Communication. In *Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing*, PODC '14, pages 114–123, New York, NY, USA, 2014. ACM. doi:10.1145/2611462.2611469.

25 Pierre Fraigniaud and Emanuele Natale. Noisy Rumor Spreading and Plurality Consensus. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*, pages 127–136, 2016. doi:10.1145/2933057.2933089.

26 Simon French. *Group consensus probability distributions: A critical survey*. University of Manchester. Department of Decision Theory, 1983.

27 Federica Garin and Luca Schenato. *A Survey on Distributed Estimation and Control Applications Using Linear Consensus Algorithms*, pages 75–107. Springer London, London, 2010. doi:10.1007/978-0-85729-033-5_3.

28 Leszek Gasieniec and Grzegorz Stachowiak. Fast Space Optimal Leader Election in Population Protocols. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2653–2667, 2018. doi:10.1137/1.9781611975031.169.

29 Leszek Gasieniec, Grzegorz Stachowiak, and Przemyslaw Uznanski. Almost logarithmic-time space optimal leader election in population protocols. *CoRR*, abs/1802.06867, 2018. arXiv:1802.06867.

30 Melvin Gauci, Monica E. Ortiz, Michael Rubenstein, and Radhika Nagpal. Error Cascades in Collective Behavior: A Case Study of the Gradient Algorithm on 1000 Physical Agents. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, pages 1404–1412, 2017. URL: http://dl.acm.org/citation.cfm?id=3091319.

31 Gustavo L Gilardoni and Murray K Clayton. On reaching a consensus using DeGroot's iterative pooling. *The Annals of Statistics*, pages 391–401, 1993.

32 Seth Gilbert and Calvin Newport. Symmetry Breaking with Noisy Processes. In *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*, pages 273–282, 2017. doi:10.1145/3087801.3087814.

33 B. Hajek. Hitting-Time and Occupation-Time Bounds Implied by Drift Analysis with Applications. *Advances in Applied Probability*, 14(3):502–525, 1982.

34 Ali Jadbabaie, Jie Lin, and A Stephen Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on automatic control*, 48(6):988–1001, 2003.

35 Varun Kanade, Frederik Mallmann-Trenn, and Thomas Sauerwald. On coalescence time in graphs: When is coalescing as fast as meeting?: Extended Abstract. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 956–965, 2019. doi:10.1137/1.9781611975482.59.

36 Soummya Kar and José MF Moura. Distributed Consensus Algorithms in Sensor Networks With Imperfect Communication: Link Failures and Channel Noise. *IEEE Transactions on Signal Processing*, 57(1):355–369, 2009.

37 D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 482–491, October 2003. doi:10.1109/SFCS.2003.1238221.

38 Adrian Kosowski and Przemyslaw Uznanski. Brief Announcement: Population Protocols Are Fast. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 475–477, 2018. URL: https://dl.acm.org/citation.cfm?id=3212788.

39 Sara Diana Leonhardt, Florian Menzel, Volker Nehring, and Thomas Schmitt. Ecology and evolution of communication in social insects. *Cell*, 164(6):1277–1287, 2016.

40 Zhongkui Li and Jie Chen. Robust consensus of linear feedback protocols over uncertain network graphs. *IEEE Transactions on Automatic Control*, 62(8):4251–4258, 2017.

41 Romain Libbrecht, Miguel Corona, Franziska Wende, Dihego O Azevedo, Jose E Serrão, and Laurent Keller. Interplay between insulin signaling, juvenile hormone, and vitellogenin regulates maternal effects on polyphenism in ants. *Proceedings of the National Academy of Sciences*, 110(27):11050–11055, 2013.

42 Frederik Mallmann-Trenn, Yannic Maus, and Dominik Pajak. Code of the experiments. URL: `https://bitbucket.org/frederikmallmann/noisy-communication-code/`.

43 Frederik Mallmann-Trenn, Yannic Maus, and Dominik Pajak. Noidy Conmunixatipn: On the Convergence of the Averaging Population Protocol. *arXiv e-prints*, page arXiv:1904.10984, April 2019. `arXiv:1904.10984`.

44 Luc Moreau. Stability of multiagent systems with time-dependent communication links. *IEEE Transactions on automatic control*, 50(2):169–182, 2005.

45 Angelia Nedić and Ji Liu. On convergence rate of weighted-averaging dynamics for consensus problems. *IEEE Transactions on Automatic Control*, 62(2):766–781, 2017.

46 Theodore S Rappaport et al. *Wireless communications: principles and practice*, volume 2. prentice hall PTR New Jersey, 1996.

47 Wei Ren and Randal W Beard. *Distributed consensus in multi-vehicle cooperative control*. Springer, 2008.

48 Wei Ren, Randal W Beard, and Ella M Atkins. A survey of consensus problems in multi-agent coordination. In *American Control Conference, 2005. Proceedings of the 2005*, pages 1859–1864. IEEE, 2005.

49 David Saldana, Amanda Prorok, Shreyas Sundaram, Mario FM Campos, and Vijay Kumar. Resilient consensus for time-varying networks of dynamic agents. In *American Control Conference (ACC), 2017*, pages 252–258. IEEE, 2017.

50 Ioannis D Schizas, Alejandro Ribeiro, and Georgios B Giannakis. Consensus-based distributed parameter estimation in ad hoc wireless sensor networks with noisy links. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 2, pages II–849. IEEE, 2007.

51 Slobodan N Simić and Shankar Sastry. Distributed environmental monitoring using random sensor networks. In *Information Processing in Sensor Networks*, pages 582–592. Springer, 2003.

52 Behrouz Touri and Angelia Nedic. Distributed consensus over network with noisy links. In *Information Fusion, 2009. FUSION'09. 12th International Conference on*, pages 146–154. IEEE, 2009.

53 Lin Xiao and S. Boyd. Fast linear iterations for distributed averaging. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, volume 5, pages 4997–5002 Vol.5, December 2003. `doi:10.1109/CDC.2003.1272421`.

54 Lin Xiao and Stephen Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1):65–78, 2004.

55 Lin Xiao, Stephen Boyd, and Seung-Jean Kim. Distributed Average Consensus with Least-mean-square Deviation. *J. Parallel Distrib. Comput.*, 67(1):33–46, January 2007. `doi:10.1016/j.jpdc.2006.08.010`.

56 Lin Xiao, Stephen Boyd, and Sanjay Lall. A scheme for robust distributed sensor fusion based on average consensus. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 63–70. IEEE, 2005.

Periodic Bandits and Wireless Network Selection

Shunhao Oh
Department of Computer Science, National University of Singapore
ohoh@u.nus.edu

Anuja Meetoo Appavoo
Department of Computer Science, National University of Singapore
anuja@comp.nus.edu.sg

Seth Gilbert
Department of Computer Science, National University of Singapore
seth.gilbert@comp.nus.edu.sg

Abstract

Bandit-style algorithms have been studied extensively in stochastic and adversarial settings. Such algorithms have been shown to be useful in multiplayer settings, e.g. to solve the wireless network selection problem, which can be formulated as an adversarial bandit problem. A leading bandit algorithm for the adversarial setting is EXP3. However, network behavior is often repetitive, where user density and network behavior follow regular patterns. Bandit algorithms, like EXP3, fail to provide good guarantees for periodic behaviors. A major reason is that these algorithms compete against fixed-action policies, which is ineffective in a periodic setting.

In this paper, we define a periodic bandit setting, and periodic regret as a better performance measure for this type of setting. Instead of comparing an algorithm's performance to fixed-action policies, we aim to be competitive with policies that play arms under some set of possible periodic patterns F (for example, all possible periodic functions with periods $1, 2, \cdots, P$). We propose Periodic EXP4, a computationally efficient variant of the EXP4 algorithm for periodic settings. With K arms, T time steps, and where each periodic pattern in F is of length at most P, we show that the periodic regret obtained by Periodic EXP4 is at most $O\big(\sqrt{PKT \log K + KT \log |F|}\big)$. We also prove a lower bound of $\Omega\big(\sqrt{PKT + KT\frac{\log |F|}{\log K}}\big)$ for the periodic setting, showing that this is optimal within log-factors. As an example, we focus on the wireless network selection problem. Through simulation, we show that Periodic EXP4 learns the periodic pattern over time, adapts to changes in a dynamic environment, and far outperforms EXP3.

2012 ACM Subject Classification Theory of computation → Online learning algorithms

Keywords and phrases multi-armed bandits, wireless network selection, periodicity in environment

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.149

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Related Version A full version of the paper is available at [21] `https://arxiv.org/abs/1904.12355`.

Supplement Material Source code: `https://github.com/Ohohcakester/PeriodicEXP4-Source`

Acknowledgements This project was funded by Singapore Ministry of Education Grant MOE2018-T2-1-160 (Beyond Worst-Case Analysis: A Tale of Distributed Algorithms).

1 Introduction

The *multi-armed bandit* problem is an online learning problem in which a player has access to a set of choices (i.e., "arms") each of which provides some reward (i.e., "gain"). At each time step, the player chooses an arm and gets some reward. In stochastic variants, rewards are determined by some probabilistic distribution. In adversarial variants, an adversary specifies

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 149; pp. 149:1–149:15

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

the rewards. Amazingly, even when rewards are adversarially chosen, the player can do fairly well! For example, the EXP3 algorithm [6] minimizes the player's "regret", ensuring that the player does almost as well as if she had selected the single fixed best arm throughout. Another fascinating property of bandit algorithms is that they work well in multi-player settings [27, 16], converging to close variants of a Nash equilibrium.

Recently, it has been shown that bandit-style algorithms can efficiently solve the *wireless network selection problem*, yielding good performance both in theory and in practice [1, 2, 7]. In this problem, each user has access to a collection of networks (e.g., a few different WiFi networks and a 4G connection); the goal is to pick networks with higher data rates. Selecting the best network is challenging, especially in dynamic environments where the "best" network changes over time, as users move and network bandwidth fluctuates. This can be modeled as an adversarial bandit problem and solved with EXP3 and its variants.

Bandit algorithms have one major weakness in dynamic settings (such as wireless network settings): they are designed to learn the average payoff of each arm, and to converge to the arm that provides the best average performance. In the stochastic case, this is exactly what you want. In the adversarial case, it leads to minimum regret, i.e., the user does almost as well as if they knew the best network in advance. If, however, the situation is changing over time, and especially if it is changing in some predictable manner, then learning the average payoff of each arm is not productive.

Periodic, repetitive patterns are a particularly common type of dynamic behavior. Take, for example, the problem of network selection. Network behavior is often repetitive, with user density and network quality following regular patterns: for example, office WiFi networks have no users at night, their performance drops when workers arrive in the morning, and the performance improves again during lunch hour. Other networks are clogged with streaming video during lunch hour and in the evenings. Periodic patterns are ubiquitous.

Unfortunately, bandit algorithms will fail badly in the case of periodic behavior. As an example, suppose a player is playing a slot machine with two arms. The first arm gives a reward of 1 when pulled on odd-numbered hours and 0 otherwise, while the second arm does the reverse, with a reward of 1 on even-numbered hours and 0 otherwise. In this simple case, a bandit algorithm will never learn this pattern, instead converging to the best single-action policy; and the best policy can only reap half of the maximum reward. The player will receive an average payout of only 1/2 per selection, despite a very predictable pattern. And when this case is extended to cycle among K arms, the best fixed choice of arm gives only $1/K$ of the total obtainable reward. Thus, algorithms like EXP3 that minimize the regret do not guarantee good performance on periodic problems.

1.1 Contributions

Our goal in this paper is to develop an efficient adversarial bandit algorithm for periodic settings, and to demonstrate the effectiveness of this algorithm in the context of the wireless network selection problem, yielding a new approach to network selection in dynamic, periodic environments. The first step is to establish the right metric by which to evaluate bandit algorithms. The performance of an adversarial bandit algorithm is heavily characterized by the definition of "regret," which forms the baseline that it competes against. And traditionally, the regret is computed with respect to the best fixed strategy.

For the periodic bandit setting, we define a better performance measure, "periodic regret", which compares an algorithm's performance against the best periodic choice of arms. No choice of period may match the input data perfectly, but the goal of periodic regret is to compare against the best choice. Moreover, we provide a generalized notion of periodicity, so that this notion of periodic regret can capture different types of patterned behavior.

Next, we develop an algorithm that minimizes periodic regret, Periodic EXP4, a computationally efficient variant of EXP4 (Exponential-weight algorithm for Exploration and Exploitation using Expert advice) [6]. We show that the algorithm minimizes periodic regret in the following sense: with K arms, $|F|$ possible periods, with each possible period of at most length P, then in an execution of length T the periodic regret is at most $O\left(\sqrt{PKT \log K + KT \log |F|}\right)$. We also prove a lower bound of $\Omega\left(\sqrt{PKT + KT\frac{\log |F|}{\log K}}\right)$ on periodic regret in an adversarial setting, showing that this is optimal within log-factors. An important aspect of Periodic EXP4 is that it is a polynomial time algorithm: we leverage the structure provided by the target periodic patterns to reduce the computational complexity. This is in contrast to EXP4 which requires exponential time and space in this context.

The other major contribution of this paper is a new algorithm for network selection that is especially optimized for environments with periodic, patterned behaviors. We simulate the network selection problem, comparing Periodic EXP4 to EXP3 and to a "randomized optimal" omniscient solution. (We have previously seen in [1] that these types of simulations are reasonably predictive of real-world behavior.)

Our first observation is that Periodic EXP4 does in fact efficiently learn periodic patterns and adapts relatively quickly to changes in network data rates (both discrete and continuous). We also see that Periodic EXP4 does indeed outperform EXP3 in periodic settings, as expected, potentially yielding significant real-world improvements.

Our second question involved the robustness of Periodic EXP4 to noisy patterns. Real-world periodic patterns are rarely perfectly periodic, suffering noise and variance. We experiment with noisy patterns, and see that Periodic EXP4 continues to work well.

Finally, our third set of experiments looked at the performance of Periodic EXP4 in the context of user mobility. We simulate several scenarios where users change location over time, leading to changes in which networks they can access (and hence changes in the load on those networks). For example, we imagine a typical office scenario where users arrive at the office in the morning, take a break for lunch, return to work, and then head home at the end of the day. We observe that Periodic EXP4 can also learn this type of periodic behavior, again, learning to adapt the users' network selection in a near-optimal fashion. In fact, we compare two versions of the algorithm: one in which the algorithm is notified when networks become unavailable, and one in which it is not – we observe that even in the latter case where it is completely oblivious to the changes, the user strategy converges to near-optimal choices.

Overall, we conclude that periodic adversarial bandit algorithms may have significant value, that Periodic EXP4 is an efficient algorithm for the problem, and that it yields a potentially interesting and useful approach to network selection.

2 Related work

In this section, we discuss relevant work done on bandit algorithms, and state-of-art wireless network selection approaches. Multi-armed bandit techniques have been successfully applied to wireless network selection [1, 2, 7]. They have also been considered for other resource selection problems, such as channel selection [13, 27], selection of the right sensors to query in a sensor network [14], and selection of replica server for content distribution networks [28].

Many variations of bandit problems have been studied, in both stochastic and adversarial settings. EXP3 is the most well-known algorithm for the standard adversarial bandit problem. With K arms and T time steps, it establishes a pseudo-regret upper bound of $O(\sqrt{KT \log K})$, which almost matches the lower bound of $\Omega(\sqrt{KT})$ [6]. The $\log K$ gap in the bounds has been recently closed by [5] bringing the upper bound down to $O(\sqrt{KT})$. But, these bound the regret against the best single-action policy, limiting their usefulness in a periodic setting.

A related problem is that of bandits with expert advice, defined in the same paper [6]. It defines a more general notion of regret, by competing against the best policy from a set. With K arms, T time steps and N experts, the EXP4 algorithm gives a pseudo-regret bound of $O(\sqrt{KT \log N})$. However, its possibly high running time and memory cost limit its use in practice. There are other algorithms for bandits with expert advice, like Context-FTPL. The latter is more computationally efficient, but has a weaker regret bound [26]. A lower bound of $\Omega(\sqrt{KT\frac{\log N}{\log K}})$ [23] has been shown, but the $\log K$ gap in bounds has not been closed.

An equivalent formulation of our generalized periodic regret (explained later in Section 4.2) has been briefly discussed in [10, Chapter 4.2.1], phrased as a contextual bandit problem where the algorithm competes against the best context set from a class of context sets. The possible use of EXP4 is mentioned, but an alternative algorithm with a weaker regret bound is instead discussed as it has a reasonable polynomial-time performance unlike EXP4.

While much of the existing literature assume a single best arm, there are other efforts to look beyond this. One approach to the stochastic version of the problem is to allow reward distributions of the arms to occasionally change [9, 22]. Our work on the other hand is fully adversarial, and makes no assumptions on the rewards produced by the adversary.

Numerous wireless network selection approaches have been proposed. Some are centralized [3, 8, 18, 25]; hence, not scalable and limited to managed networks. A number of distributed approaches have been proposed, with various limitations. Some rely on coordination from networks [15], while others require cooperation of wireless devices [12]. Others assume global knowledge [20, 4, 19], or availability of some information [30, 11]. A continuous-time multi-armed bandit approach in a stochastic setting has been considered in [29]. A similar setting to ours, though non-periodic and in the stochastic setting, is considered in [7].

3 Wireless Network Selection

Here, we describe the wireless network selection problem, discuss the periodicity of events in wireless environments, and formulate the network selection problem as a bandit problem.

3.1 Wireless network selection problem.

We consider an environment with multiple wireless devices and heterogeneous wireless networks, such as the one depicted in Figure 1. The latter illustrates four mobile users with their (active) mobile devices, and five wireless networks, namely four WiFi networks and a cellular network (represented using 3 cellular base stations). The wireless networks have limited areas of coverage. Hence, each mobile device may have access to a different set of wireless networks depending on their location, e.g. different networks are available at home and at the office. The bandwidths of wireless networks may also vary with time. Each mobile device aims to quickly identify and associate with the best network, which may vary over time, to maximize their data rates.

Mobile users tend to have daily routines that follow repetitive patterns - going to the office each morning, lunch at noon, returning home in the evening; these activities are performed at fixed times each weekday. Figure 1 broadly depicts the daily routine of a mobile user, Alice. Network behavior, which is affected by user density, is also often repetitive and follows a regular pattern. For example, the available bandwidth of office WiFi networks is likely to be higher during lunch hours, where the office is nearly empty. A good network selection protocol learns and adapts to periodic patterns in network quality for better performance.

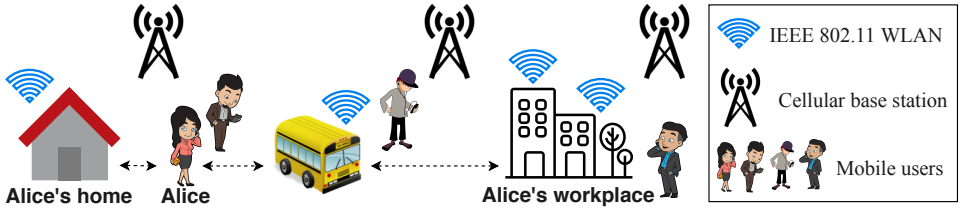


Figure 1 Mobile devices with access to a different set of wireless networks as the user moves.

3.2 Wireless network selection as a bandit problem

A device must be aware of the bit rate it can observe from each network to perform an optimal network selection. While this information is unknown at the time of selection, the device can estimate the achievable bit rate by exploring the networks. The network selection problem can be seen as a multi-armed bandit problem in a multi-player setting. A mobile device is a player, and each network can be considered as an arm. Every so often (e.g. once per minute), a device selects a network (analogous to pulling an arm) and observes a bit rate (gain) for that network. The gain from other networks is unknown to the device. Given that mobile devices operate in a dynamic environment, they must continuously explore and adapt to changes, by deciding which networks to select in sequence. The goal of each device is to maximize its cumulative gain over time. Since the quality of a wireless network is affected by its number of clients, other mobile devices in the environment may be considered to be adversaries. We hence use the adversarial setting. A leading bandit algorithm in this setting is EXP3.

4 Periodic Bandit Problem

In this section, we introduce the periodic bandit problem and discuss periodic regret.

We consider a general bandit problem. On each time step, an algorithm is allowed to pick any one out of K possible arms, and each arm produces a certain amount of reward. These rewards are unknown to the algorithm, which can only observe the reward of the arm it picked. We aim to maximize the total reward obtained by the algorithm. We study the adversarial setting with a possibly adaptive adversary, which decides on the distribution of rewards at each time step, taking into consideration the outcomes of past random events.

Let K be the number of arms. The set of arms is $[K] := \{1, 2, \cdots, K\}$. Let $x_i(t) \in [0, 1]$ be the reward earned by arm $i \in [K]$ at time step t. Let $a(t) \in [K]$ be the arm played by the algorithm at time t. Let T be the total number of time steps. The set of time steps is $[T] := \{1, 2, \cdots, T\}$. Thus, the total reward earned by the algorithm after T iterations is $\sum_{t=1}^{T} x_{a(t)}(t)$. The commonly used performance measure for bandit algorithms is regret. Regret compares the total reward obtained by the algorithm against a "best possible" reward "OPT" after some number of time steps T. Different types of regret compare the algorithm's result to different notions of the optimal result.

We can define a form of regret where OPT is allowed to pick any arm in $[K]$ at each time step. For later reference we will refer to this as full regret, defined as follows:

$$R_{full}(T) = \sum_{t=1}^{T} \max_{i \in [K]} E\Big[x_i(t)\Big] - E\Big[\sum_{t=1}^{T} x_{a(t)}(t)\Big]$$

The above definition uses what is commonly known as pseudo-regret, rather than expected regret. For the rest of this paper, we will often refer to pseudo-regret as simply "regret". Expectations are taken over the possible randomness of the algorithm and adversary.

In most studies of adversarial bandits, a weaker definition of regret is used. This is because full regret uses too powerful an adversary, and it is impossible to achieve better than linear expected full regret in the worst case (we include a proof in the full version of the paper [21]). Therefore, it is common to define a notion of regret where OPT is required to use the same arm for all T time steps. We refer to this as weak regret, defined as follows:

$$R_{weak}(T) = \max_{i \in [K]} \sum_{t=1}^{T} E\Big[x_i(t)\Big] - E\Big[\sum_{t=1}^{T} x_{a(t)}(t)\Big]$$

Weak regret however, severely limits what OPT can do, and being competitive with an algorithm that can only pick one arm and stick to it may not be a very strong result.

4.1 Periodic Regret

We can bridge the two with a periodic definition of regret. Taking the idea that a periodic choice of arms is likely to perform well in situations with periodic patterns, we can define a regret function which measures how competitive an algorithm is with the best periodic choice of arms. For example, we can say OPT is forced to play the same arm every $\tau \in \mathbb{N}$ steps. This defines a regret function as follows,

$$R_\tau(T) = \sum_{\ell=1}^{\tau} \max_{i \in [K]} \sum_{\bar{t}=0}^{\lfloor \frac{T-\ell}{\tau} \rfloor} E\Big[x_i(\bar{t}\tau + \ell)\Big] - E\Big[\sum_{t=1}^{T} x_{a(t)}(t)\Big]$$

As OPT may optionally still pick the same arm on all time steps, this is a generalization of weak regret. This makes for a regret value in between weak regret and full regret.

If we were competing against the regret for a specific, known value of τ, this would be equivalent to playing τ independent instances of the adversarial bandits problem over approximately T/τ time steps each. By playing τ separate instances of an algorithm for weak regret, and by Theorem 2 in Section 6.1, we have an upper/lower bound of $\Theta(\sqrt{\tau KT})$.

However, if we were to consider that the "best possible" period τ may not be known (for example, if OPT were to consist of the best periodic function for any of the possible periods $\tau \in \{1, \cdots, P\}$), these bounds do not apply as easily.

4.2 Generalized Periodic Regret

A generalization of the periodic case is the use of partition functions. Fix a maximum number of labels P. We define this upper bound P for use in our analysis later on. A partition function $f : [T] \to [P]$ is a function that assigns every time step a label from 1 to P. We consider two partition functions the same if their choice of label assignments are permutations of each other. The regret under function f would be when OPT is forced to play the same arm for all timesteps with the same label as assigned by f.

$$R_f(T) = \sum_{\ell \in [P]} \max_{i \in [K]} \sum_{\bar{t} \in f^{-1}(\ell)} E\Big[x_i(\bar{t})\Big] - E\Big[\sum_{t=1}^{T} x_{a(t)}(t)\Big] \tag{1}$$

Consider a set of partition functions $F \subseteq \{f : [T] \to [P]\}$ for some $P \in \mathbb{N}$. F is necessarily finite. The regret under the function set f would be when OPT can choose to play using any of the partition functions in F. This gives the following regret definition:

$$R_F(T) = \max_{f \in F} \sum_{\ell \in [P]} \max_{i \in [K]} \sum_{\bar{t} \in f^{-1}(\ell)} E\Big[x_i(\bar{t})\Big] - E\Big[\sum_{t=1}^{T} x_{a(t)}(t)\Big] \tag{2}$$

This definition (2) of periodic regret gives us more choice in how we want to define our potential periodic patterns to learn, through deciding on the labels on each time step for each function. We demonstrate this with our choice of partition functions in Section 7.

To model the example described earlier with periods $\tau \in \{1, 2, \cdots, P\}$, we can use the set of partitions $F = \{f_1, f_2, \cdots, f_P\}$, where $f_\tau(t) := (t \bmod \tau) + 1$ for each $t \in [T]$, $\tau \in [P]$.

5 The Periodic EXP4 Algorithm

We discuss the relationship between our generalized periodic setting and the problem of bandits with expert advice, and hence the applicability of EXP4 [6] to the problem. We use this to introduce Periodic EXP4, an efficient algorithm for generalized periodic regret.

5.1 Applying Bandits with Expert Advice to Periodic Bandit Problems

Periodic bandit problems can be reduced to the problem of bandits with expert advice. In the problem of bandits with expert advice, we are given a set Π of N experts. Each expert predicts an arm on each time step. We fix the number of time steps T. Thus an expert can be seen as a function $\pi : [T] \to [K]$. An algorithm to solve this problem would make use of each expert's predictions on each time step, to obtain a reward competitive with the best expert in the set. This gives us the following regret definition:

$$R_\Pi(T) = \max_{\pi \in \Pi} \sum_{t=1}^{T} x_{\pi(t)}(t) - E\Big[\sum_{t=1}^{T} x_{a(t)}(t)\Big]$$

This can be used to model all of the above notions of regret. For full regret, we have $\Pi := \{\pi : [T] \to [K]\}$, the set of all possible functions from $[T]$ to $[K]$. For weak regret, Π is the set of all constant functions from $[T]$ to $[K]$.

In the generalized periodic setting, let F be the set of partition functions $f : [T] \to [P]$. For each function $f \in F$, let Θ_f be the set of all possible mappings $\theta : f([T]) \to [K]$ from the image set $f([T])$ of f to the set of arms $[K]$ (thus $|\Theta_f| = K^{|f([T])|}$). Each composition $\theta \circ f$, $f \in F$, $\theta \in \Theta_f$ thus represents a possible mapping of the time steps $[T]$ to arms. Thus, for the generalized periodic setting, $\Pi = \{\theta \circ f \mid f \in F, \theta \in \Theta_f\}$.

We note that when $\Pi_1 \subseteq \Pi_2$, we will have $R_{\Pi_1}(T) \leq R_{\Pi_2}(T)$. Let Π_{full}, Π_{weak} and Π_F be the sets of functions corresponding to full regret, weak regret and generalized periodic regret under some function set F respectively. Thus, for any nonempty set F of partition functions, we have $R_{\Pi_{weak}}(T) \leq R_{\Pi_F}(T) \leq R_{\Pi_{full}}(T)$.

An existing algorithm for this problem is the EXP4 algorithm [6], which achieves a regret upper bound of $O(\sqrt{KT \log N})$, where $N := |\Pi|$. We can thus apply EXP4 directly to our problem. However, a commonly cited drawback of the EXP4 algorithm is that its running time and memory cost are at least linear in N. This is an issue as N is often very large. For example, in the generalized periodic setting, the size of N could easily be on the order of

Algorithm 1 Periodic EXP4.

procedure INITIALIZATION
  for each $f \in F$ **do**
    for each $\ell \in f([T])$ **do**
      for each $i \in [K]$ **do**
        Initialize $b_i^{\ell,f}(1) = 1$
procedure ALGORITHM
  for each time step $t = 1, 2, \cdots, T$ **do**
    for each $i \in [K]$ **do**
      $$r_i(t) := \sum_{f \in F} \left(b_i^{f(t),f}(t) \prod_{\ell \in f([t]) \setminus \{f(t)\}} \sum_{j=1}^{K} b_j^{\ell,f}(t) \right)$$
    for each $i \in [K]$ **do**
      $$p_i(t) = \frac{r_i(t)}{\sum_{j=1}^{K} r_j(t)}$$
    Play arm $i_t \in [K]$ from the probabilities $p_1(t), p_2(t), \cdots, p_K(t)$
    Obtain reward $x_{i_t}(t)$
    for each $f \in F$ **do**
      for each $\ell \in f([T])$ **do**
        for each $i \in [K]$ **do**
          if $i = i_t$ and $\ell = f(t)$ **then**
            $b_i^{\ell,f}(t+1) = b_i^{\ell,f}(t) \exp(\frac{\gamma}{K} x_i(t)/p_i(t))$
          else
            $b_i^{\ell,f}(t+1) = b_i^{\ell,f}(t)$

$|F|K^P$, which is exponential in P. However, we show below that in the generalized periodic setting, we can devise an algorithm that is distributionally equivalent to EXP4 and can be made to run in time polynomial in $|F|$, K and P.

The EXP4 algorithm works by assigning a weight w_π (with initial value 1) to each expert $\pi \in \Pi$. The probability $p_i(t)$ of playing an arm $i \in [K]$ would then be $\sum_{\pi(t)=i} w_\pi(t) / \sum_\pi w_\pi(t)$, the ratio of the combined weights of the experts agreeing to play arm i to the total weight of the experts. Whenever an arm $i \in [K]$ is played, each expert who suggested arm i will have their weight adjusted by some factor $\exp(\frac{\gamma}{K} x_i(t)/p_i(t))$. More details on EXP4 are given in [6]. Note that it discusses a more general form of expert advice where each expert suggests a probability vector on the arms. However, we only require the case where at each time step, each expert suggests one arm with probability 1, and all other arms with probability 0.

5.2 Periodic EXP4, Memory and Running Time Costs

Periodic EXP4 (Algorithm 1) is distributionally equivalent to the EXP4 algorithm when run with the set of experts $\Pi = \{\theta \circ f \mid f \in F, \theta \in \Theta_f\}$. The key intuition behind this algorithm is that the generalized periodic setting produces many symmetries in the weight computation for each expert. Specifically, we take advantage of how for each partition function f, the set of experts contains every possible combination of arm assignments to labels in the image set $f([T])$. This allows us to compute the probabilities that EXP4 would play each arm at each time step without computing the individual weights of every expert.

For brevity, let $P_f := |f([T])|$ be the number of labels used by the function f. Necessarily $P_f \leq P$. The memory requirement is $O(K \sum_{f \in F} P_f)$, which is at most $O(KP|F|)$. A naive implementation of the algorithm gives a running time of $O(K^2 \sum_{f \in F} P_f)$ per time step, but with some pre-computation, the running time can be lowered as shown in the full paper [21].

5.3 Correctness of Periodic EXP4

To show correctness, we show that our algorithm produces the same probability distribution over arms as EXP4 in every time step. Define $\pi_{\theta,f}$ as the expert which at time t recommends arm $\theta \circ f(t)$ with probability 1 and all other arms with probability 0. We show this algorithm is distributionally equivalent to EXP4, where $\Pi = \{\pi_{\theta,f} | f \in F, \theta \in \Theta_f\}$. In EXP4, each expert $\pi_{\theta,f}$ would have some weight $w_{\theta,f}(t)$ at time step t. At time step t, EXP4 plays arm i with probability $p_i(t)$ represented by the following expression:

$$p_i(t) = \frac{\sum_{f \in F, \theta \in \Theta_f, \theta \circ f(t) = i} w_{\theta,f}(t)}{\sum_{f \in F, \theta \in \Theta_f} w_{\theta,f}(t)}$$

Thus, to show that the two algorithms are distributionally equivalent, as $p_i(t) := r_i(t)/\sum_{j=1}^{K} r_j(t)$ in our algorithm, for each successive time step t, we only need to show the following:

$$r_i(t) = \sum_{f \in F, \theta \in \Theta_f, \theta \circ f(t) = i} w_{\theta,f}(t)$$

The details of this derivation is given in the full paper [21]. We can thus formally state a regret upper bound as follows (Theorem 1). This upper bound comes directly from EXP4's regret bound of $O(\sqrt{KT \log N})$, where the number of experts $N = \sum_{f \in F} K^{|f([T])|} \leq |F| K^P$.

▶ **Theorem 1.** *With K arms, T time steps, $|F|$ partition functions, with every function having at most P labels, Periodic EXP4 gives a regret upper bound of $O\left(\sqrt{PKT \log K + KT \log |F|}\right)$.*

6 Lower Bounds

In this section, we provide lower bounds for the case of a single partition and for a set of partitions. We demonstrate that the upper and lower bounds differ by a factor of $\log K$.

The existing regret lower bound for the problem of bandits with expert advice [23] is $\Omega(\sqrt{KT \frac{\log N}{\log K}})$. This lower bound is derived by dividing the time steps $[T]$ into $\frac{\log N}{\log K}$ equal parts. For the generalized periodic setting, as this lower bound uses an instance that can be modeled with a single partition function, it does not give immediate insight into whether having multiple different periods or partition functions increases the difficulty of the problem.

6.1 Lower Bound for a Single Partition

We consider the case with only a single partition function $f : [T] \to [P]$, which partitions the time steps into P labels $1, 2, \cdots, P$. The sizes of the partitions are $|f^{-1}(1)|$, $|f^{-1}(2)|$, $\cdots$, $|f^{-1}(P)|$ respectively. It seems like intuitively, by seeing this as P separate instances of the weak regret setting, and by the existing $\Theta(\sqrt{KT})$ upper/lower bounds on weak regret [6, 5], we would have an upper/lower bound of $\Theta(\sum_{\ell=1}^{P} \sqrt{K|f^{-1}(\ell)|})$. For equally sized partitions of size approximately $\frac{T}{P}$ each, this bound would be $\Theta(\sqrt{PKT})$.

However, while the upper bound is clearly met by running P independent instances of an algorithm for weak regret, the lower bound is less clear. Even when considering it as P separate instances, there is a possibility of an algorithm "reacting" to losses in other instances to play differently in the current instance, obtaining a higher total reward as a result. For completeness, we include a proof for the lower bound (Theorem 2) in the full paper [21].

▶ **Theorem 2.** *Fix a partition function $f : [T] \to [P]$ which assigns a label to each time step. Assume that for each $\ell \in f([T])$, there are at least $K/(4 \ln \frac{4}{3})$ time steps with label ℓ. Then the minimax pseudo-regret (1), over all algorithms a and adversaries R, has a lower bound as follows, for some positive constant c:*

$$\inf_a \sup_R \left(\max_{\theta \in \Theta_f} E\Big[\sum_{t \in [T]} x_{\theta \circ f(t)}(t) \Big] - E\Big[\sum_{t \in [T]} x_{a(t)}(t) \Big] \right) \geq \sum_{\ell \in [P]} \sqrt{cK|f^{-1}(\ell)|}$$

If we consider the simple case where OPT may play only periodic functions from any period $\tau \in \{1, 2, \cdots, P\}$, it can do no worse than if it were only allowed to play at period P. We thus obtain a lower regret bound of $\sqrt{PKT}$.

6.2 Lower Bound for the Generalized Periodic Setting

Let F be the set of partitions, so $|F|$ is the number of partitions. Let P be the maximum number of labels of any partition in F. For sufficiently large T and $K \leq P$, we obtain a pseudo-regret(2) lower bound of $\Omega(\sqrt{PKT} + \sqrt{KT \frac{\log |F|}{\log K}})$. It is proved in the full paper [21].

If $P < K$ instead, a simple lower bound can be obtained by using only P out of the K arms, so we obtain a problem with P arms and maximum partition size P. This gives us a lower bound of $\Omega\Big(\sqrt{PKT} + \sqrt{PT \frac{\log |F|}{\log P}}\Big)$. We can then merge these two lower bounds into a single expression $\Omega\Big(\sqrt{PKT} + \sqrt{\min(P, K) T \frac{\log |F|}{\log \min(P,K)}}\Big)$.

6.3 Analysis of Bounds

A conclusion we can make from Section 6.2 is that having multiple periods indeed increases the difficulty of the problem - we have obtained a lower bound higher than the known upper bound of $O(\sqrt{PKT})$ had only one partition function of the maximum period P been used.

With K arms, T time steps, $|F|$ partition functions, with every function having at most P labels, Periodic EXP4 gives an upper bound of $O\big(\sqrt{PKT \log K + KT \log |F|}\big)$. On the other hand, we have a lower bound of $\Omega\big(\sqrt{PKT + KT \frac{\log |F|}{\log K}}\big)$ in the case where $K \leq P$. This gives a gap of $\sqrt{\log K}$ between the two bounds. Interestingly, this log-factor is the same as the current gap between the upper and lower bounds in the problem of bandits with expert advice. This is possibly because we use a similar lower bound proof to the problem of bandits with expert advice [23], as well as a similar algorithm for the upper bound.

7 Experimental Evaluation

In this section, we discuss the implementation details of Periodic EXP4 and parameter values chosen, evaluate the algorithm via simulation, and compare its performance to EXP3 [6]. We show how Periodic EXP4 (a) learns periodic patterns over time under both discrete and continuous changes in network data rates, (b) outperforms EXP3, (c) is robust to noisy patterns, and (d) adapts to changes due to mobility of users.

We benchmark against "Optimal Random", a player with prior knowledge of the actual bandwidths of each network. In each time slot, it picks a network from a probability distribution equal to the ratios of the bandwidths. For example, with network bandwidths $4, 10$ and 6, the probability of picking the networks will be 0.2, 0.5 and 0.3, respectively.

All the algorithms are implemented in Python, using SimPy [24], while the core algorithm is written in C++. We use a time-varying learning rate $\gamma = t^{-\frac{1}{10}}$ [17] for both Periodic EXP4 and EXP3; γ slowly tends to zero to ensure convergence [27] while at the same time ensures that the algorithm does not take too long to learn (it learns slowly when γ is very small). Although they are not pre-requirements of Periodic EXP4, for simplicity, we assume that (a) a network's bandwidth is equally shared among its clients, and (b) devices are time-synchronized. To reduce numerical error in our simulations, we substitute computations of $\sum_{x \in Y} \exp(x)$ with $\exp(\max_{x \in Y} x)$. In nearly all cases, sums of exponentials in our algorithm are heavily dominated by a single term, making the values of the two expressions approximately equal. Experimentally, we find that this has negligible effects on the values computed within the algorithm.

We do simulations on synthetic data. We consider setups with 20 mobile devices and 3 wireless networks, unless otherwise specified. While the number of devices remain constant throughout the simulation run, the data rates and availability of networks may change. We assume that a network selection is performed once every minute; hence, 1440 time slots is one simulated day. All results presented are from 20 simulation runs, of 86,400 time slots each (i.e., 2 simulated months). The pattern of network behavior and/or user mobility over the first 1440 time slots is repeated 60 times; we refer to each repetition as an "iteration".

We apply Periodic EXP4 in the generalized periodic setting. We define a partition function of period τ as one which divides each iteration of 1440 time slots into τ equal contiguous segments, labeled 1 to τ in chronological order. The same labels are used for each successive repetition. Unless otherwise specified, we use the period set $\{1, \cdots, 24\}$. This refers to using 24 partition functions, of periods 1 to τ respectively.

7.1 Evaluation Criteria

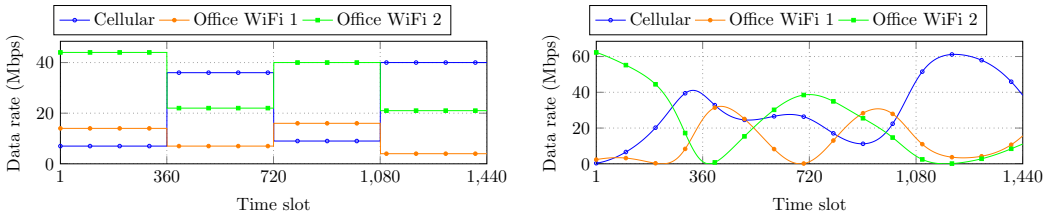
Good assignments of devices to networks divide the available bandwidth evenly among the devices. We thus evaluate the performance of the algorithms based on the lowest data rate observed by any of the devices. We compare this to the optimal allocation of devices, which maximizes the lowest data rate observed by any device. If a device with the lowest data rate observes 3Mbps, but the optimal's lowest is 5Mbps, we say it loses 40% of its achievable gain. We refer to this percentage loss as the "distance to optimal minimum" in our results.

We do not use average cumulative gain as a performance measure because in our problem setting, average gain is maximized as long as there is at least one user in each network.

7.2 Performance Comparison of Algorithms

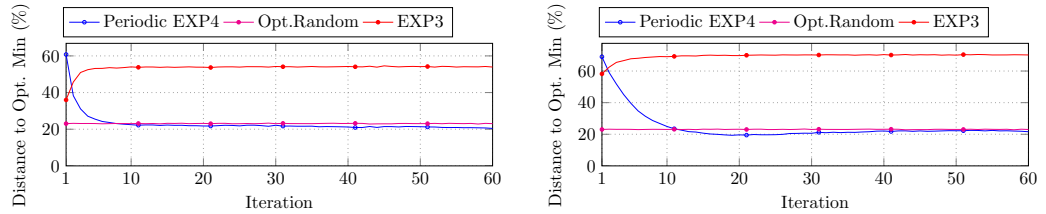
We consider two setups, both at an office with two WiFi networks and a cellular network. The data rates of these networks vary over time. The first setup involves discrete changes in network bandwidths at fixed time intervals (Figure 2a). In the second setup, the data rates vary continuously with time (Figure 2b). Figures 3a and 3b show that in both setups, the distance to optimal minimum of Periodic EXP4 drops over time while EXP3 shows no noticeable improvement with time.

Figure 4 for the continuous setup explains this improvement. The figure for the discrete setup is in the full paper [21]. At each time step, each user has a probability of picking each



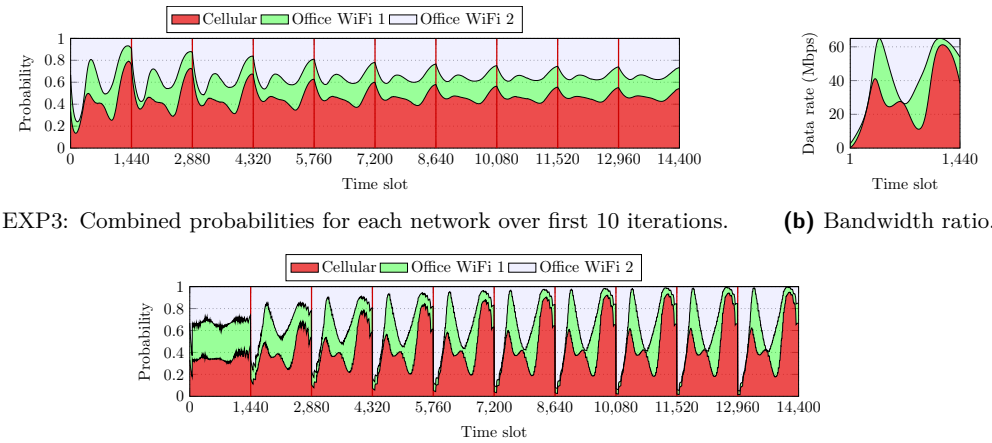
(a) Discrete changes in network data rates. **(b)** Continuous changes in network data rates.

Figure 2 Changes in network data rates over one iteration (this is repeated 60 times).



(a) Performance under discrete setup. **(b)** Performance under continuous setup.

Figure 3 Distance to optimal minimum of Periodic EXP4 and EXP3 over 60 iterations.



(a) EXP3: Combined probabilities for each network over first 10 iterations. **(b)** Bandwidth ratio.

(c) Periodic EXP4: Combined probabilities for each network over first 10 iterations.

Figure 4 Area chart showing the time variation of combined probabilities in the continuous setup. Figure 4b shows the actual ratio of the bandwidths of the three networks within any one iteration.

of the networks. If we consider the combined probability of picking each network, we can see that in Periodic EXP4, these probabilities converge towards the ratios of the bandwidths of the networks (Figures 4c). This is despite the continuous setup having no obvious best period. On the other hand, EXP3's probabilities slowly flatten out (Figure 4a). This is consistent with what we would expect, as EXP3 seeks to be competitive with the best fixed-action policy, meaning that it only seeks out the best fixed arm to play.

Figure 5 shows that while EXP3 initially learns more quickly, Periodic EXP4 eventually outperforms EXP3 (which converges to the network with the best average performance), with a performance similar to Optimal Random. From our experiments, we find that while all algorithms have similar total cumulative gains, we may note that Periodic EXP4 is fairer than EXP3, with significantly lower variance. We present these results in the full paper [21].

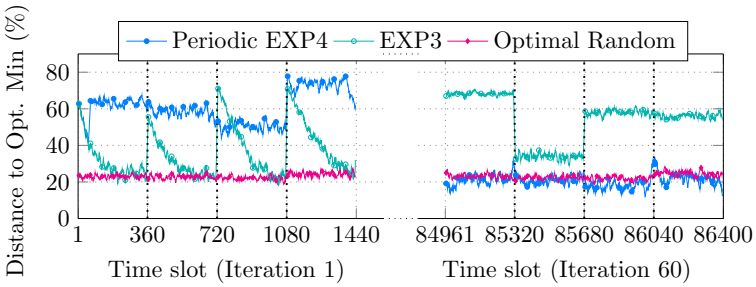


Figure 5 Distances to Optimal minimum in the first and last repetitions of the discrete setting in Figure 2a. Vertical lines indicate points where data rates change.

7.3 Other Experiments

In the full version of the paper [21], we discuss a few more experiments, the results of which are briefly summarized as follows:

1. **Performance in Noisy Settings:** On each time step, we apply a 10% Gaussian noise to each of the networks' data rates. We find that our algorithms are largely unaffected by noise in the data, giving similar results with and without noise.
2. **Comparison of Period Sets:** We do a comparison between different possible period sets F. We find that the algorithm learns more slowly with larger period sets (e.g. $\{1, 2, \cdots, 45\}$, as compared to $\{1, 2, \cdots, 15\}$), but can converge to better results on more complex instances (instances where the bandwidth may fluctuate more wildly).
3. **Mobility of Users:** We consider a setup where users move around and have access to different sets of networks at different times. We compare Vanilla Periodic EXP4, which is oblivious to networks possibly becoming unavailable, against an optimized version, which selects only from the set of currently available networks. While the optimized version initially yields a better performance, they eventually perform equally well when the Vanilla Periodic EXP4 algorithm learns the pattern.

8 Conclusion

In this paper, we develop an efficient variant of EXP4 for the periodic bandit problem, give nearly matching upper and lower bounds for it, and demonstrate its advantages in learning periodic behavior in the context of the network selection problem.

An interesting issue raised in contrasting this paper and [9, 22] is whether non-stationary bandit problems are better modeled stochastically or adversarially. While these papers address non-stationary rewards primarily in a stochastic setting with some adversarial aspects, we tackle the periodic bandit problem in a fully adversarial setting. Using the adversarial setting has the benefit of not placing any constraints on the adversary; we adapt to the periodic setting only through our definition of regret. A proper comparison of stochastic and adversarial methods for network selection is a possible future line of work.

References

1 Anuja Meetoo Appavoo, Seth Gilbert, and Kian-Lee Tan. Shrewd Selection Speeds Surfing: Use Smart EXP3! In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 188–199. IEEE, 2018.

2 Anuja Meetoo Appavoo, Seth Gilbert, and Kian-Lee Tan. Cooperation Speeds Surfing: Use Co-Bandit! *arXiv preprint*, 2019. `arXiv:1901.07768`.

3 E. Aryafar, A. Keshavarz-Haddad, C. Joe-Wong, and M. Chiang. Max-Min Fair Resource Allocation in HetNets: Distributed Algorithms and Hybrid Architecture. In *ICDCS, 2017*, pages 857–869. IEEE, 2017.

4 E. Aryafar, A. Keshavarz-Haddad, M.l Wang, and M. Chiang. RAT selection games in HetNets. In *INFOCOM*, pages 998–1006. IEEE, 2013.

5 Jean-Yves Audibert and Sébastien Bubeck. Minimax policies for adversarial and stochastic bandits. In *COLT*, pages 217–226, 2009.

6 P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

7 O. Avner and S. Mannor. Multi-user lax communications: A multi-armed bandit approach. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, April 2016. doi:10.1109/INFOCOM.2016.7524557.

8 Y. Bejerano, S-J. Han, and L. E. Li. Fairness and load balancing in wireless LANs using association control. In *MobiCom*, pages 315–329. ACM, 2004.

9 Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic Multi-Armed-Bandit Problem with Non-stationary Rewards. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 199–207. Curran Associates, Inc., 2014. URL: http://papers.nips.cc/paper/5378-stochastic-multi-armed-bandit-problem-with-non-stationary-rewards.pdf.

10 Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

11 M. H. Cheung, F. Hou, J. Huang, and R. Southwell. Congestion-Aware Distributed Network Selection for Integrated Cellular and Wi-Fi Networks. *arXiv preprint*, 2017. arXiv:1703.00216.

12 S. Deng, A. Sivaraman, and H. Balakrishnan. All your network are belong to us: A transport framework for mobile network selection. In *HotMobile*. ACM, 2014.

13 Yi Gai, Bhaskar Krishnamachari, and Rahul Jain. Learning multiuser channel allocations in cognitive radio networks: A combinatorial multi-armed bandit formulation. In *New Frontiers in Dynamic Spectrum, 2010 IEEE Symposium on*, pages 1–9. IEEE, 2010.

14 D. Golovin, M. Faulkner, and A. Krause. Online distributed sensor selection. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 220–231. ACM, 2010.

15 B. Kauffmann, F. Baccelli, A. Chaintreau, V. Mhatre, K. Papagiannaki, and C. Diot. Measurement-based self organization of interfering 802.11 wireless access networks. In *INFOCOM 2007*, pages 1451–1459. IEEE, 2007.

16 R. Kleinberg, G. Piliouras, and E. Tardos. Multiplicative updates outperform generic no-regret learning in congestion games. In *ACM STOC*, pages 533–542. ACM, 2009.

17 S. Maghsudi and S. Stanczak. Relay selection with no side information: An adversarial bandit approach. In *WCNC*, pages 715–720. IEEE, 2013.

18 A. Mishra, V. Brik, S. Banerjee, A. Srinivasan, and W. A. Arbaugh. A Client-Driven Approach for Channel Management in Wireless LANs. In *Infocom*, 2006.

19 E Monsef, A. Keshavarz-Haddad, E. Aryafar, J. Saniie, and M. Chiang. Convergence properties of general network selection games. In *INFOCOM*, pages 1445–1453. IEEE, 2015.

20 D. Niyato and E. Hossain. Dynamics of network selection in heterogeneous wireless networks: An evolutionary game approach. *TVT*, 58(4):2008–2017, 2009.

21 Shunhao Oh, Anuja Meetoo Appavoo, and Seth Gilbert. Periodic Bandits and Wireless Network Selection [full version of paper]. *arXiv preprint*, 2019. arXiv:1904.12355.

22 Allesiardo Robin, Raphaël Feraud, and Odalric-Ambrym Maillard. The Non-stationary Stochastic Multi-armed Bandit Problem. *International Journal of Data Science and Analytics*, March 2017. doi:10.1007/s41060-017-0050-5.

23 Yevgeny Seldin and Gábor Lugosi. A lower bound for multi-armed bandits with expert advice. In *13th European Workshop on Reinforcement Learning (EWRL)*, 2016.

24 SimPy. SimPy - Event discrete simulation for Python, 2016. , accessed 2018-19-12. URL: `https://simpy.readthedocs.io/`.

25 K. Sui, M. Zhou, D. Liu, M. Ma, D. Pei, Y. Zhao, Z. Li, and T. Moscibroda. Characterizing and improving wifi latency in large-scale operational networks. In *MobiSys*, pages 347–360. ACM, 2016.

26 Vasilis Syrgkanis, Akshay Krishnamurthy, and Robert Schapire. Efficient algorithms for adversarial contextual learning. In *International Conference on Machine Learning*, pages 2159–2168, 2016.

27 C. Tekin and M. Liu. Performance and Convergence of Multi-user Online Learning. In *GAMENETS*, pages 321–336. Springer, 2011.

28 H. A. Tran, S. Hoceini, A. Mellouk, J. Perez, and S. Zeadally. QoE-based server selection for content distribution networks. *IEEE Transactions on Computers*, 63(11):2803–2815, 2014.

29 Q. Wu, Z. Du, P. Yang, Y.-D. Yao, and J. Wang. Traffic-aware online network selection in heterogeneous wireless networks. *TVT*, 65(1):381–397, 2016.

30 K. Zhu, D. Niyato, and P. Wang. Network selection in heterogeneous wireless networks: Evolution with incomplete information. In *WCNC*, pages 1–6. IEEE, 2010.

On the Complexity of Local Graph Transformations

Christian Scheideler
Paderborn University, Germany
https://cs.uni-paderborn.de/en/ti/
scheideler@upb.de

Alexander Setzer
Paderborn University, Germany
https://cs.uni-paderborn.de/en/ti/
asetzer@mail.upb.de

Abstract

We consider the problem of transforming a given graph G_s into a desired graph G_t by applying a minimum number of primitives from a particular set of *local graph transformation primitives*. These primitives are local in the sense that each node can apply them based on local knowledge and by affecting only its 1-neighborhood. Although the specific set of primitives we consider makes it possible to transform any (weakly) connected graph into any other (weakly) connected graph consisting of the same nodes, they cannot disconnect the graph or introduce new nodes into the graph, making them ideal in the context of supervised overlay network transformations. We prove that computing a minimum sequence of primitive applications (even centralized) for arbitrary G_s and G_t is NP-hard, which we conjecture to hold for any set of local graph transformation primitives satisfying the aforementioned properties. On the other hand, we show that this problem admits a polynomial time algorithm with a constant approximation ratio.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Approximation algorithms analysis

Keywords and phrases Graphs transformations, NP-hardness, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.150

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Related Version A full version of the paper is available at https://arxiv.org/abs/1904.11395.

Funding This work was supported by the German Research Foundation (DFG) within the Collaborative Research Center "On-The-Fly Computing" (SFB 901) under Grant No.: GZ SFB 901/02.

1 Introduction

Overlay networks are used in many contexts, including peer-to-peer systems, multipoint VPNs, and wireless ad-hoc networks. In fact, any distributed system on top of a shared communication infrastructure usually has to form an overlay network (i.e., its participating sites have to know each other or at least some server) to allow the exchange of information.

A fundamental task in the context of overlay networks is to maintain or adapt its topology to a desired topology, where the desired topology might either be pre-defined or depend on a certain objective function. The problem of reaching a pre-defined topology has been extensively studied in the context of self-stabilizing overlay networks (e.g., [29, 21, 12, 5, 22, 7]), and the problem of adapting the topology based on a certain objective function has been studied in the context of self-adapting and -optimizing overlay networks (e.g., [33, 14, 2, 19, 11, 3, 10, 8]). Many of these approaches are decentralized, and because of that,

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 150; pp. 150:1–150:14


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

the work (in terms of number of edge changes) they need to adapt to a desired topology might be far away from the minimum possible work to reach that topology. In fact, no non-trivial results on the competitiveness of decentralized overlay network adaptations are known so far other than handling single join or leave operations, and it is questionable whether any good competitive result can be achieved with a decentralized approach. An alternative approach would be that a server is available for controlling the network adaptations, and this has already been considered in the context of so-called supervised overlay networks. In a *supervised overlay network* there is a dedicated, trusted node called *supervisor* that controls all network adaptations but otherwise is not involved in the functionality of the overlay network (such as serving search requests), which is handled in a peer-to-peer manner. This has the advantage that even if the supervisor is down, the overlay network is still functional. Solutions for supervised overlay networks have been proposed in [24, 15], for example, and the results in [24] imply that, for specific overlay networks, any set of node arrivals and departures can be handled in a constant competitive fashion (concerning the work needed for adding and removing edges) to get back to a desired topology. But no general result is known so far for supervised overlay networks concerning the competitiveness of converting an initial topology into a desired topology. Also, no result is known so far on how to handle the problem that a supervisor could be faulty or even act maliciously.

A malicious supervisor would pose a significant problem for an overlay network since it could easily launch *Sybil attacks* (i.e., flooding the overlay network with fake or adversarial nodes) or *Eclipse attacks* (i.e., isolating nodes from other nodes in the overlay network). We thus ask: Can we limit the power of a supervisor such that it cannot launch an eclipse or sybil attack while still being able to convert the overlay network from any connected topology to any other connected topology?

We answer the question to the affirmative by determining a set of graph transformation commands, also called *primitives*, that only the supervisor may issue to the nodes. These primitives are powerful enough to transform any (weakly) connected topology into any other (weakly) connected topology but still allow the nodes to locally check that applying them does not disconnect the network or introduce a new node into the network. We additionally aim at minimizing the reconfiguration overhead, i.e., the number of commands to be issued (and, related to this, the number of changes to be made to node neighborhoods) to reach a desired topology. Unfortunately, as we will show, this cannot be done efficiently for the set of primitives we consider unless $\mathsf{P} \neq \mathsf{NP}$, and we conjecture that this holds for any set of commands that has the aforementioned property of giving the participants the ability to locally check that they cannot be used for eclipse or sybil attacks. However, we are able to give an $O(1)$-approximation algorithm for this problem.

1.1 Model and Problem Statement

We model the overlay network as a graph, i.e., nodes represent participants of the network and if there is a directed edge (u, v) in the graph, this means that there is a connection from u to v. Undirected edges $\{u, v\}$ model the two connections from u to v and from v to u. Since there may be multiple connections between the same pair of participants, the graphs we consider in this work are multigraphs, i.e., edges may appear several times in the (multi-)set of edges. For convenience throughout this work we will use the term "graph" instead of multigraph and refer to "edge sets" even though their elements need not be unique.

We consider the following set P_d of four primitives for the manipulation of directed graphs, first introduced by Koutsopoulos et al. [25] in the context of overlay networks:

Introduction. If a node u has a reference of two nodes v and w with $v \neq w$, u *introduces* w to v if u sends a message to v containing a reference of w while keeping the reference.

Delegation. If a node u has a reference of two nodes v and w s.t. u, v, w are all different, then u *delegates* w's reference of v if u sends a message to v containing a reference of w and deletes the reference of w.

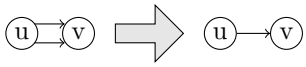
Fusion. If a node u has two references v and w with $v = w$, then u *fuses* the two references if it only keeps one of these references.

Reversal. If a node u has a reference of some other node v, then u *reverses* the connection if it sends a reference of itself to v and deletes its reference of v.

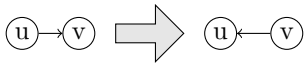
The four primitives are visualized in Figure 1. Note that for the Introduction primitive, it is possible that $w = u$, i.e., u introduces itself to v. To simplify the description, we sometimes say that a node u introduces or delegates the *edge* (u, v) if u introduces v to some other node or delegates v's reference to some other node, respectively.

(a) Introduction Primitive.

(b) Delegation Primitive.



(c) Fusion primitive.



(d) Reversal primitive.

Figure 1 The four primitives in P_d in pictures.

The primitives in P_d are known to be universal (c.f. [25]), i.e., it is possible to transform any weakly connected graph into any other weakly connected graph by using only the primitives in P_d. Note that for every edge (u, v) used in any of the primitives, either (u, v) still exists after the corresponding primitive is applied, or there is still an (undirected) path from u to v in the resulting graph. This directly implies that no application of the primitives can disconnect the graph. We assume that all connections are *authorized*, meaning that both endpoints are aware of the other endpoint of this connection. Thus, if for an edge (u, v) that is supposed to be transformed into (v, u) by an application of the Reversal Primitive, v checks that u actually was the previous endpoint of the former edge then the primitives cannot be used to introduce new nodes into the graph.

For undirected graphs, consider the set P_u containing only the primitives Introduction, Delegation and Fusion (defined correspondingly). These three primitives, accordingly, are universal on undirected graphs, i.e., any connected undirected graph can be transformed into any other connected undirected graph by applying the primitives in P_u (c.f. [25]).

We make the following observation:

▶ **Observation 1.** *The Introduction primitive is the only primitive that can increase the number of edges in a graph. The Fusion primitive is the only primitive that can decrease the number of edges in a graph. The Delegation primitive is the only primitive that can remove the last edge between two nodes (i.e., an edge of multiplicity one).*

A *computation* C is a finite sequence $G_1 \Rightarrow G_2 \Rightarrow \cdots \Rightarrow G_l$ of either directed or undirected graphs, in which each graph G_{i+1} is obtained from G_i by the application of a single primitive from P_d or P_u, respectively. The graphs G_1 and G_l are called the *initial* and the *final* graphs of C, respectively. The variable l is called the *length of the computation*.

We define the *Undirected Local Graph Transformation Problem* (ULGT) as follows: given two connected undirected graphs G_s, G_t, find a computation of minimum length whose initial graph is G_s and whose final graph is G_t. The corresponding decision problem κ-ULGT is defined as follows: given a positive integer k and two connected undirected graphs G_s and G_t, decide whether there is a computation with initial graph G_s and final graph G_t of length at most k. Accordingly we define the *Directed Local Graph Transformation Problem* (DLGT) and κ-DLGT, which differ from the according problems in that the graphs are directed.

1.2 Related Work

Graph transformations have been studied in many different contexts and applications, including but not limited to pattern recognition, compiler construction, computer-aided software engineering, description of biological developments in organisms, and functional programming languages implementation (for a more detailed introduction and literature overview, we refer the reader to [4], [20], or [31, 13]). Simply put, a graph transformation (or graph-rewriting) system consists of a set of rules $L \to R$ that may be applied to subgraphs isomorphic to L of a given graph G thus replacing L with R in G. Since changing the labels assigned to a graph (graph relabeling) is also a kind of graph transformation, basically every distributed algorithm can be understood as a graph transformation system (c.f. [13]). The type of graph transformations probably closest related to our work is the area of *Topology Control* (TC). In simple terms, the goal of TC is to select a subgraph of a given input graph that fulfills certain properties (such as connectivity) and optimizes some value (such as the maximum degree). This problem has been studied in a variety of settings (for surveys on this topic see, e.g., [27], or [6]) and although the usual approach is decentralized, there are also some centralized algorithms in this area (see, e.g., [30]). However, these works only consider the complexity of computing an optimal topology (instead of the complexity of transforming the graph by a minimum number of rule applications). There is one work by Lin [28] proving the NP-hardness of the *Graph Transformation Problem*, in which the goal is to find the minimum integer k such that an initial graph G_s can be transformed into a final graph G_t by adding and removing at most k edges in G_s. Our work differs from that work in that we do not allow arbitrary edge relocations but restrict them to a set of rules that can be applied locally (and we also provide constant-factor approximation algorithms).

Our approximation algorithms use an approximation algorithm for the Undirected Steiner Forest Problem as a black-box (also known as the Steiner Subgraph Problem with edge sharing, or, in generalizations, the Survivable Network Design Problem or the Generalized Steiner Problem). 2-approximations of this problem were first given by Agrawal, Klein, and Ravi [1], and by Goemans and Williamson [16], and later also by Jain [23]. Gupta and Kumar [18] showed a simple greedy algorithm to have a constant approximation ratio and recently, Groß et al. [17] presented a local-search constant approximation for Steiner Forest.

1.3 Our Contribution

The main contributions of this paper are as follows: We prove the Undirected and the Directed Local Graph Transformation Problem to be NP-hard in Section 2. Furthermore, in Section 3 we show that they belong to APX, i.e., there exist constant approximation algorithms for these two problems.

2 NP-hardness results

In this section, we show the NP-hardness of the Undirected Local Graph Transformation Problem by proving the NP-hardness of k-ULGT (see Section 2.1). Since k-DLGT's NP-hardness is very similar for k-ULGT, we omit its proof and only briefly sketch the differences in the full version of this paper [32].

Throughout this section, for any positive integer i we use the notation $[i]$ to refer to the set $\{1, 2, \ldots, i\}$.

2.1 k-ULGT is NP-hard

We prove k-ULGT's hardness via a reduction from the Boolean satisfiability problem (SAT) which was proven to be NP-hard by Cook [9] and, independently, by Levin [26]. We briefly recap SAT as follows:

▶ **Definition 1** (SAT). *Given a set X of n Boolean variables $x_1, \ldots, x_n$ and a Boolean formula Φ over the variables in X in conjunctive normal form (CNF), decide whether there is a truth assignment $t : X \to \{0, 1\}$ that satisfies Φ.*

To reduce SAT to k-ULGT, we use the following reduction function:

▶ **Definition 2** (Reduction function). *Let $S = (X, \Phi)$ be a SAT instance, in which $X = \{x_1, \ldots, x_n\}$ is the set of Boolean variables and $\Phi = C_1 \wedge \cdots \wedge C_m$ for clauses $C_1, \ldots, C_m$. Then $f(S) = (G_s, G_t, k)$ in which $k = 2n + m$ and G_s and G_t are undirected graphs defined as follows. Without loss of generality, assume that each literal $y_i \in \{x_i, \overline{x_i}\}$ occurs only once in each clause. We say $y_i \in C_j$ if literal y_i occurs in C_j.*

We define the following sets of nodes: $V_C = \{C_1, \ldots, C_m\}$, and $V_{X_i} = \{x_i, \overline{x_i}, s_i, t_i\}$. Then, the set of nodes of G_s and G_t is $V = \bigcup_{1 \le i \le n} V_{X_i} \cup V_C \cup \{r\}$. For the set of edges, define $E_{X_i} = \{\{s_i, x_i\}, \{\overline{x_i}, s_i\}, \{x_i, t_i\}, \{t_i, \overline{x_i}\}\}$, $E_{C_j} = \{\{y_i, C_j\} | y_i \in \{x_i, \overline{x_i}\} \wedge y_i \text{ occurs in } C_j\}$, $E_{sr} = \{\{s_i, r\} | 1 \le i \le n\}$, $E_{tr} = \{\{t_i, r\} | 1 \le i \le n\}$, $E_{Cr} = \{\{C_j, r\} | 1 \le j \le m\}$. Both G_s and G_t have the edges in $\bigcup_{1 \le i \le n} E_{X_i} \cup \bigcup_{1 \le j \le m} E_{C_j}$. Additionally, G_s has the edges in E_{sr} and G_t has the edges in $E_{tr} \cup E_{Cr}$.

Intuitively, each variable x_i is mapped to a *gadget* X_i consisting of the four nodes $x_i, \overline{x_i}, s_i$, and t_i. Also each clause C_j is connected with each literal occurring within it. Lastly, in G_s, each of the s_i is connected with the node r, whereas in G_t, each of the t_i and each of the C_j are connected with r. Figure 2 shows an example of the output of the reduction function for a given formula in CNF.

We now show that every SAT instance S is satisfiable if and only if $f(S)$ is a "yes" instance of k-ULGT. We start with the "only if" part for this is the simpler direction:

▶ **Lemma 3.** *If a SAT instance S as in Definition 2 is satisfiable then $f(S) = (G_s, G_t, k)$ with $k = 2n + m$ is a k-ULGT instance and there is a computation with initial graph G_s and final graph equal to G_t of length at most $2n + m$.*

Proof. Assume there is a satisfying truth assignment $t : X \to \{0, 1\}$ of S. For every $1 \le i \le n$ let $y_i := x_i$ if $t(x_i) = 1$ or $y_i := \overline{x_i}$ if $t(x_i) = 0$. We construct the following computation with initial graph G_s and final graph G_t:

1. For every $1 \le i \le n$, s_i delegates the edge $\{s_i, r\}$ to y_i.
2. For every $C_j \in \{C_1, \ldots, C_m\}$ choose one neighbor $z_j \in \{y_1, \ldots, y_n\}$ (we show below that this exists), and let z_j introduce r to C_j.
3. For every $1 \le i \le n$, y_i delegates the edge $\{y_i, r\}$ to t_i.

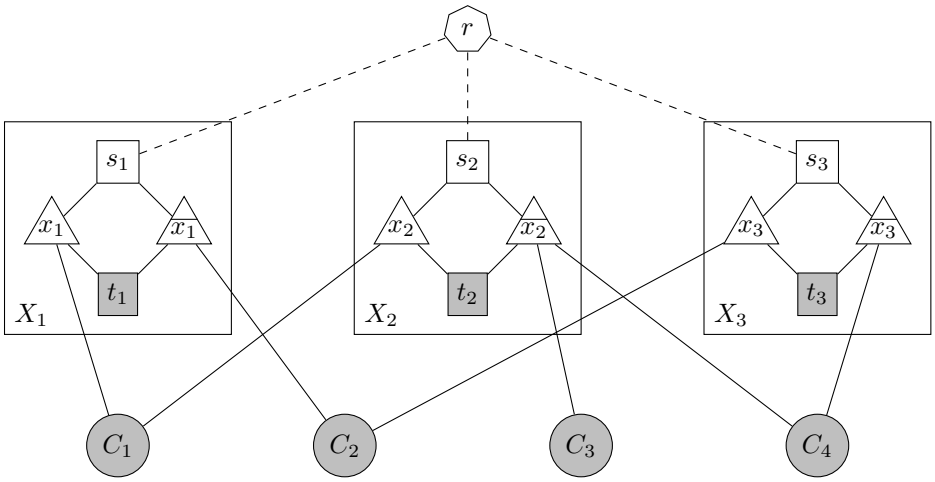


Figure 2 Graph G_s returned by the reduction function for the (example) Boolean formula $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (\overline{x_2}) \wedge (\overline{x_2} \vee \overline{x_3})$. G_t differs from G_s in that the dashed edges do not exist and all grey nodes share an edge with node r.

Obviously, the length of this computation is $2n + m$. To prove the missing part, recall that every C_j is satisfied under t, i.e., there is at least one literal z_j in C_j that evaluates to true, i.e., there is an $i \in [n]$ such that $z_j = x_i$ if $t(x_i) = 1$, or $z_j = \overline{x_i}$ if $t(x_i) = 0$. By definition of y_i, $z_j = y_i$. Thus because z_j occurs in C_j, y_i was a neighbor of C_j during Step 2. ◀

The "if" part is more complex. We begin with the following insight that will prove helpful in the course of this part.

▶ **Lemma 4.** *Suppose the nodes in the initial graph of a computation C can be decomposed into disjoint sets $V_1, \ldots, V_k, P$ such that there is no edge $\{u, v\}$ for some $u \in V_i$, $v \in V_j$, $i, j \in [k]$, $i \neq j$ and throughout C none of the nodes in P applies a primitive. Then there is no edge $\{u, v\}$ for some $u \in V_i$, $v \in V_j$, $i, j \in [k]$, $i \neq j$ in any graph of the computation.*

Proof. Assume there is a computation C and sets $V_1, \ldots, V_k, P$ as defined above and assume for contradiction that the claim is not true. We consider the first edge $\{u, v\}$ such that $u \in V_i$, $v \in V_j$, $i, j \in [k]$, $i \neq j$. Clearly, it cannot have been created by the application of a Fusion primitive. Thus it must have been created by an Introduction or Delegation primitive applied by a node w that knew both u and v before the application of this primitive. By definition of P, $w \notin P$, i.e., $w \in V_l$ for some $l \in [k]$. However, by the definition of $\{u, v\}$, u and v must have been from V_l as well, yielding a contradiction. ◀

The next lemma we show represents a main building block of the proof of the "if" part.

▶ **Lemma 5.** *Let S be a* SAT *instance and let $(G_s, G_t, k) = f(S)$. For every computation C with initial graph G_s and final graph equal to G_t of length at most $2n + m$ it holds: There are $y_1, \ldots, y_n$, $y_i \in \{x_i, \overline{x_i}\}$ for every $i \in [n]$, such that in C there are no edges other than $E(G_s) \cup E(G_t) \cup \{\{y_i, r\} | i \in [n]\}$ and no edge occurs twice (where $E(G_s)$ and $E(G_t)$ denote the edge set of G_s and G_t, respectively).*

Due to space constraints, we only sketch the proof here, whereas the full proof can be found in the full version of this paper [32]. The general idea of the proof of Lemma 5 is the following: To obtain the target graph, for each $j \in [m]$ the edge $\{C_j, r\}$ has to be created and for

each $i \in [n]$ the edge $\{t_i, r\}$ has to be created. Each of these creations involves a distinct application of a primitive. Therefore, only n applications of primitives are left in a feasible computation. We show that the nodes in each gadget i have to apply at least one primitive p_i that does not create one of the above edges. This implies that each gadget may apply no other primitive than p_i to create an edge that is not in the target graph and that the nodes r and C_j themselves cannot apply any primitives at all which by Lemma 4 means that there are no inter-gadget edges. We use these facts to prove that p_i is used to remove the edge $\{s_i, r\}$ thereby creating either $\{x_i, r\}$ or $\{\overline{x_i}, r\}$.

The rest of the proof of the "if" part, as formalized by the following lemma, is comparably straightforward.

▶ **Lemma 6.** *Let S be a* SAT *instance as in Definition 2. If $f(S) = (G_s, G_t, k)$ with $k = 2n + m$ is a* K-ULGT *instance and there is a computation with initial graph G_s and final graph equal to G_t of length at most $2n + m$ then S is satisfiable.*

Proof. In the following, we refer to the variables defined in Definition 2. Furthermore, we say a computation is *feasible* if and only if its initial graph is G_s, its target graph is G_t and its length is at most $2n + m$. Moreover, we say that the edge that is established during the application of an Introduction or Delegation primitive (the edge (v, w) in Figures 1a and 1b) is the *result* of the Introduction or Delegation, respectively.

Assume that $f(S) = (G_s, G_t, 2n + m)$ is a K-ULGT instance and there is a feasible computation C for $f(S)$. According to Lemma 5 there are $y_1, \ldots, y_n$, $y_i \in \{x_i, \overline{x_i}\}$ for every $i \in [n]$ such that in C there are no edges other than $E(G_s) \cup E(G_t) \cup \{\{y_i, r\}|[n]\}$. Note that in G_t, for every $j \in [m]$ there is an edge $\{C_j, r\}$ and each such edge must have been the result of an introduce or Delegation primitive applied by an y_i, $i \in [n]$ (as throughout C, the C_js do not have any other neighbors with an edge to r that could possible create this edge) . Let $g : \{C_1, C_2, \ldots, C_m\} \to \{y_1, y_2, \ldots, y_n\}$ be the mapping of each C_j to the y_i who applied a primitive that resulted in the edge $\{C_j, r\}$. Consider the truth assignment $t : X \to \{0, 1\}$ such that $t(x_i) = 1$ if $y_i = x_i$ and $t(x_i) = 0$ if $y_i = \overline{x_i}$. Observe that $t(y_i) = 1$ for every $i \in [n]$. Assume for contradiction that there is a clause C_j in S that does not evaluate to 1 under t. Note that $g(C_j)$ must occur in C_j by construction. However, since $g(C_j) = y_i$ for some $i \in [n]$ and $t(y_i) = 1$, we obtain the desired contradiction. ◀

3 Approximation Algorithms

In this section, we first describe an approximation algorithm for ULGT (see Section 3.1) and prove it to have a constant approximation ratio (see Section 3.2). Note that a constant approximation factor algorithm for DLGT can be obtained by a slight adaptation of this algorithm. For a description of this, we refer the reader to the full version [32] due to space constraints.

As an ingredient our algorithm uses a 2-approximation algorithm (see Section 1.2) for the Undirected Steiner Forest Problem (USF) defined as follows: Given a graph G and a set S of pairs of nodes from G, find a forest F in G with a minimum number of edges such that the two nodes of each pair in S are connected by a path in F.

3.1 Algorithm Description

For an initial graph $G_s = (V, E_s)$ and a final graph $G_t = (V, E_t)$, we define the set of *additional* edges $E_\oplus := E_t \setminus E_s$ and the set of *excess* edges $E_\ominus := E_s \setminus E_t$. We now describe the algorithm in detail and then summarize its pseudo-code in Algorithm 1. Our algorithm consists of two parts, the first of which dealing with establishing all additional edges and the second of which dealing with removing all excess edges.

Algorithm 1 Approximation algorithm for ULGT.

Input: Initial graph G_s and final graph G_t.

First part:

1: Compute a 2-approximate solution $F_{ALG,\oplus}$ for the USF with input G_s, and the set $E_\oplus$ as the set of pairs of nodes.
2: For each tree T in $F_{ALG,\oplus}$, select a root node r_T and connect all nodes in T that are incident to an edge in $E_\oplus$ with r_T (details below).
3: For each $\{u, v\} \in E_\oplus$, the root of the tree u and v belong to applies the Introduction primitive to create the edge $\{u, v\}$.
4: For each tree T in $F_{ALG,\oplus}$, delegate all superfluous edges (i.e., not belonging to G_s or $E_\oplus$) created during Step 2 bottom up in T rooted at r_T, starting with the lowest level. At each intermediate node fuse all of these edges before delegating them to the next predecessor.

Second part:

5: Compute a 2-approximate solution $F_{ALG,\ominus}$ for the USF with input G_t, and the set $E_\ominus$ as the set of pairs of nodes.
6: For each $e \in E_\ominus$, let $s(e)$ be an arbitrary of the two endpoints of e. For each tree T in $F_{ALG,\ominus}$, select a root node r_T and for each $e \in E_\ominus$ whose endpoints belong to T, connect $s(e)$ with r_T (similar to Step 2, details below).
7: For each $e \in E_\ominus$, $s(e)$ delegates the other endpoint to r_T.
8: For each tree T in $F_{ALG,\ominus}$, delegate all superfluous edges bottom-up and fuse multiple edges as in Step 4.

In the first part, using an arbitrary 2-approximation algorithm for the USF as a black box the algorithm computes a 2-approximate solution to the following USF instance: The given graph is G_s, and the set of pairs of nodes is $E_\oplus$. Note that the result is a forest such that for every edge $\{u, v\} \in E_\oplus$, u and v belong to the same tree. For each tree T in this forest the algorithm then selects an arbitrary root r_T and connects all nodes in T that are incident to an edge in $E_\oplus$ to r_T. The exact details of this will be described when we analyze the length of the resulting computation. In the next step, for every T, for every $\{u, v\} \in E_\oplus$ such that u and v belong to T, r_T introduces u to v to each other, thereby creating the edge $\{u, v\}$. After that, the superfluous edges are deleted in a bottom-up fashion: every node that does not have a descendant with a superfluous edge (in the tree T this node belongs to when viewing this tree as rooted by r_T), fuses all superfluous edges and delegates the last such to its predecessor in the tree. Note that all superfluous edges in the same tree T have r_T as one of their endpoints.

The second part of the algorithm is similar to the first, with the following differences: In the fifth step, the USF is approximated for the graph G_t and $E_\ominus$ as the set of pairs. Note that the solution is a subgraph of the graph obtained after the first part of the algorithm. In the sixth step, only one of the two endpoints of an edge from $E_\ominus$ is selected to become connected with the root of the tree the endpoints belong to. In the seventh step (where in the first part the additional edges are created by the r_T nodes), for each edge $e \in E_\ominus$, the endpoint selected in the sixth step delegates this edge to r_T (resulting in the edge $\{r_T, v\}$).

3.2 Analysis

In this section we show that Algorithm 1 is a constant-approximation algorithm for ULGT, which is formalized by the following theorem:

▶ **Theorem 7.** ULGT ∈ APX.

For convenience we will analyze the two parts of the algorithm individually. Therefore, for a given initial graph G_s and final graph G_t, let $ALG_1(G_s, G_t)$ be the length of the computation of the first part of the algorithm for this instance, $ALG_2(G_s, G_t)$ be the length of the computation of the second part, and $ALG(G_s, G_t) := ALG_1(G_s, G_t) + ALG_2(G_s, G_t)$. Furthermore, let $OPT(G_s, G_T)$ be the length of an optimal solution to ULGT for initial graph G_s and final graph G_t. We also define the intermediate graph $G' = (V, E_s \cup E_\oplus)$. In the course of the analysis we will establish a relationship between $ALG_1(G_s, G_t)$ and $OPT(G_s, G')$ and between $ALG_2(G_s, G_t)$ and $OPT(G', G_t)$. This will aid us in determining the approximation factor of Algorithm 1 due to the following lemma:

▶ **Lemma 8.** $OPT(G_s, G') + OPT(G', G_t) \leq 2OPT(G_s, G_t) + |E_\oplus|$.

Proof. Let $\mathcal{P}$ denote the problem equal to κ-ULGT with initial graph G_s and final graph G_t with the additional requirement that the computation must contain G' and let $OPT'(G_s, G_t)$ be the length of an optimal solution to it. Clearly, $OPT(G_s, G') + OPT(G', G_t) \leq OPT'(G_s, G_t)$ (otherwise, split the computation at G' and improve either $OPT(G_s, G')$ by the first part obtained or $OPT(G', G_t)$ by the second part obtained). We now show that $OPT'(G_s, G_t) \leq 2OPT(G_s, G_t) + |E_\oplus|$.

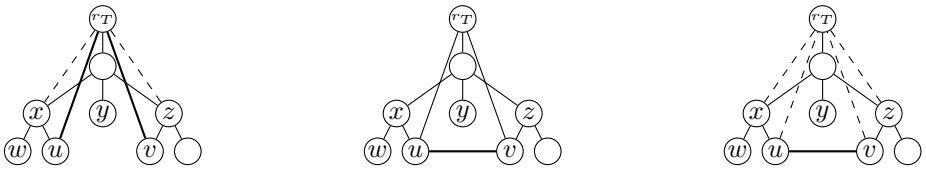
Consider a computation C whose initial graph is G_s, whose final graph is G_t and whose length is $OPT(G_s, G_t)$ (note that such a computation is an optimal solution to ULGT). We now transform C into a computation that represents a solution to $\mathcal{P}$. This transformation increases its length by only $OPT(G_s, G_t) + |E_\oplus|$ and thus proves the above claim (recall that any solution to $\mathcal{P}$ has at least the size of an optimal solution to it). First, because the final graph does not contain any edge $\{u, v\} \in E_\ominus$, for every such edge there is one last Delegation in C that removes this edge (recall Observation 1). We replace each of these last delegations by an introduction and obtain a new computation C' of equal length. Note that changing these delegations to introductions does not make the computation infeasible as this only causes the graph to have additional edges. The final graph of C' is $(V, E_t \cup E_\ominus) = (V, E_s \cup E_\oplus) = G'$ (recall that $E_t = (E_s \cup E_\oplus) \setminus E_\ominus$). Next we append C' by C and obtain the computation C'' of length $2OPT(G_s, G_t)$. Note that since C transformed G_s to G_t, this second half of C'', which starts from $G' = (V, E_s \cup E_\oplus)$, has the final graph $G'' = (V, E_t \cup E_\oplus)$, i.e., each edge from $E_\oplus$ appears twice in G''. Thus we extend C'' by fusing each edge from $E_\oplus$ with its double, resulting in a computation C''' of length $2OPT(G_s, G_t) + |E_\oplus|$. Since C''' represents a solution to $\mathcal{P}$ for initial graph G_s and final graph G_t, this completes the proof. ◀

In the rest of the analysis we show that $ALG_1(G_s, G_t) \leq 11OPT(G_s, G')$ (Lemma 9) and that $ALG_2(G_s, G_t) \leq 7OPT(G', G_t)$ (Lemma 10). By Lemma 8 this implies that $ALG(G_s, G_t) \leq 11(2OPT(G_s, G_t) + |E_\oplus|) \leq 33OPT(G_s, G_t)$ (since, clearly, $OPT(G_s, G_t) \geq |E_\oplus|$), which yields the claim of Theorem 7.

We begin with the former claim, which is formalized by the following lemma:

▶ **Lemma 9.** $ALG_1(G_s, G_t) \leq 11OPT(G_s, G')$.

Proof. Let $F_{OPT,\oplus}$ be an optimal solution for the USF with input G_s and $E_\oplus$ as the set of nodes and recall that $F_{ALG,\oplus}$ is the USF approximation computed in Step 1 of Algorithm 1. Throughout the analysis, $|F_{OPT,\oplus}|$ and $|F_{ALG,\oplus}|$ will denote the number of edges in these



(a) Step 2 connects all endpoints of edges in $E_\oplus$ belonging to T with r_T.

(b) In Step 3, r_T creates the edges in $E_\oplus$ that belong to T by an Introduction.

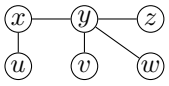
(c) Step 4 removes all superfluous edges by delegating and fusing them up in the tree.

Figure 3 Example of a tree T with root r_T for Step 2-4 of Algorithm 1 assuming $\{u, v\} \in E_\oplus$. $ST(x)$ consists of x, w, and u. x is relevant, whereas y is not. Dashed edges exist temporarily during the displayed step.

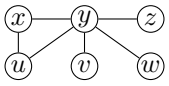
solutions. In the first part of this proof, we show that $ALG_1(G_s, G_t) \leq 4|F_{OPT,\oplus}| + 3|E_\oplus|$. The second part then consists in proving $OPT(G_s, G') \geq |F_{OPT,\oplus}| - |E_\oplus|$, which together with the observation that $OPT(G_s, G') \geq |E_\oplus|$ yields the claim.

To upper bound $ALG_1(G_s, G_t)$, we analyze the number of primitives applied in each of the steps of the first part of the approximation algorithm. In Step 1, no primitive is applied. To keep the number of edges as low as possible (which saves Fusion primitives in Step 4), the algorithm for every T in $F_{ALG,\oplus}$ connects the desired nodes to r_T in Step 2 in the following way: To simplify the description, we view T as rooted at r_T and for a node $u \in T$ denote by $ST(u)$ the set consisting of u and all of its descendants in the tree T rooted at r_T. We say a node u is *relevant* if $ST(u)$ contains a node with an endpoint in $E_\oplus$. See Figure 3 for an illustration of these notions. First of all, r_T introduces itself to all relevant children. Then, starting from the second level, we proceed level-wise in the tree: For each level i, every node u at level i checks whether u is an endpoint of an edge in $E_\oplus$ or $\{u, r_T\}$. If so, it introduces r_T to all relevant children. Otherwise, it introduces r_T to all but one of its relevant children (chosen arbitrarily) and delegates r_T to the relevant child it did not introduce r_T to. One can check that the result of this procedure is that each node incident to an edge in $E_\oplus$ has an edge to r_T for the tree T it belongs to, see Figure 3a. Note that according to the definition of $F_{ALG,\oplus}$, for each pair $\{u, v\} \in E_\oplus$ u and v belong to the same tree T. The above procedure increases the number of edges by at most $2|E_\oplus|$ and requires at most $|F_{ALG,\oplus}|$ applications of primitives (since each tree T with k edges contains at most $k + 1$ nodes and for each node u in T, at most one primitive is applied to create $\{u, r_T\}$ and this is done for neither r_T nor the nodes at level 1). It is easy to see that Step 3 (c.f. Figure 3b) involves exactly $E_\oplus$ applications of primitives. For the length of Step 4 (c.f. Figure 3c), note that for every tree T at most $|T|$ delegations have to be applied because every node in each tree has to apply at most one Delegation (causing $|F_{ALG,\oplus}|$ delegations in total) and at most $2|E_\oplus|$ fusions have to be applied for this is the number of superfluous edges created during Step 2. All in all, Step 2, Step 3, and Step 4 involve $|F_{ALG,\oplus}|$, $E_\oplus$, and $|F_{ALG,\oplus}| + 2|E_\oplus|$ applications of primitives, respectively. This makes a total of $2|F_{ALG,\oplus}| + 3|E_\oplus|$. Since $F_{ALG,\oplus}$ is a 2-approximation of $F_{OPT,\oplus}$, we obtain $ALG_1(G_s, G_t) \leq 4|F_{OPT,\oplus}| + 3|E_\oplus|$.

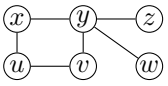
For the lower bound on $OPT(G_s, G')$, assume for contradiction that there is a computation C with initial graph G_s and final graph G' of length $L < |F_{OPT,\oplus}| - |E_\oplus|$. Let $G_s = G_1 \Rightarrow G_2 \Rightarrow \cdots \Rightarrow G_L$ be the sequence of graphs of this computation. For every $\{u, v\} \in E_\oplus$ we iteratively create a path from u to v in the following way: Begin with $P^L_{u,v} := (u, v)$. Note that $P^L_{u,v}$ exists in G_L. We iterate through C in reverse order and for every graph G_i, if $P^{i+1}_{u,v}$ exists in G_i, $P^i_{u,v} := P^{i+1}_{u,v}$. Otherwise, since G_{i+1} is the result of a single application of a primitive to G_i, there is exactly one edge $\{x, y\}$ in $P^{i+1}_{u,v}$ that exists in G_{i+1} but not in G_i and this edge was created by the application of an Introduction or Delegation primitive of some node w such that $\{w, x\}$ and $\{w, y\}$ exist in G_i. Thus, let $P^i_{u,v}$ be $P^{i+1}_{u,v}$ with (x, y) replaced



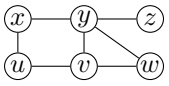
(a) Initial graph G_1.



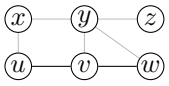
(b) G_2: x has introduced u to y.



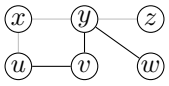
(c) G_3: y has delegated u to v.



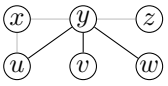
(d) G_4: y has introduced v to w.



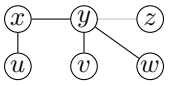
(e) G_4: $P^4_{u,v} = (u, v)$ and $P^4_{v,w} = (v, w)$.



(f) G_3: $P^3_{u,v} = (u, v)$ and $P^3_{v,w} = (v, y, w)$.



(g) G_2: $P^2_{u,v} = (u, y, v)$ and $P^2_{v,w} = (v, y, w)$.



(h) G_1: $P^1_{u,v} = (u, x, y, v)$ and $P^1_{v,w} = (v, y, w)$.

Figure 4 Example of an optimal computation C with initial graph G_1 and $E_\oplus = \{\{u, v\}, \{v, w\}\}$, and the notions used in the proof of Lemma 9. The upper row shows C in order, the lower row illustrates the path sets $P^i_{u,v}$ and $P^i_{v,w}$, which are defined by iterating through C in reverse order. In the lower row, the edges drawn black in G_i are the edges belonging to to F^i. Observe that F_1 represents a solution to the USF for graph G_1 and node pairs $E_\oplus$.

by (x, w, y) and note that $P^i_{u,v}$ exists in G_i. Eventually, we obtain a path $P^1_{u,v}$ that exists in G_s. For $i \in \{1, \dots, L\}$, let $F^i := \bigcup_{\{u,v\} \in E_\oplus} E(P^i_{u,v})$ (where $E(P)$ is the set of all edges on the path P) and note that F^1 represents a solution to the USF with input G_s and $E_\oplus$ as the set of node pairs. An example is given in Figure 4. For an arbitrary $i \in \{1, \dots, L-1\}$, note that $|F^i| \leq |F^{i+1}| + 1$: if G_{i+1} was obtained from G_i by the application of a Fusion primitive, this inequality trivially holds as none of the above paths changes in this case. Otherwise, G_{i+1} was obtained from G_i by an application of an Introduction or Delegation primitive by some node w causing at most one edge $\{x, y\}$ to exist in G_{i+1} that does not exist in G_i. In this case, we further know that $\{w, x\}$ and $\{w, y\}$ exist in G_i and by the definition of the above paths, for every pair $\{u, v\}$ such that $P^{i+1}_{u,v}$ contains the edge $\{x, y\}$ the path $P^i_{u,v}$ contains (x, w, y) as a sub-path instead and for all other pairs $\{u', v'\}$, $P^i_{u',v'} = P^{i+1}_{u',v'}$. By the definition of F^i and F^{i+1}, this implies $|F^i| \leq |F^{i+1}| + 1$ also in this case. All in all we obtain that $|F^1| \leq |F^L| + L = |E_\oplus| + L$ because $F^L = E_\oplus$ (note the definition of F^L). By the assumption that $L < |F_{OPT,\oplus}| - |E_\oplus|$, we obtain $|F^1| < |F_{OPT,\oplus}|$, which represents a contradiction. ◀

▶ **Lemma 10.** $ALG_2(G_s, G_t) \leq 7OPT(G', G_t)$.

Proof. The general structure of this proof follows the line of the proof of Lemma 9, but differs in the details. Similar to the notation used in that proof, let $F_{OPT,\ominus}$ be an optimal solution for the USF with input G_t and $E_\ominus$ as the set of nodes and recall that $F_{ALG,\ominus}$ is the USF approximation computed in Step 5 of Algorithm 1. Analogously, $|F_{OPT,\ominus}|$ and $|F_{ALG,\ominus}|$ denote the number of edges in these solutions. In the first part of this proof, we show that $ALG_2(G_s, G_t) \leq 4|F_{OPT,\ominus}| + 3|E_\ominus|$. The second part then consists in proving $OPT(G', G_t) \geq |F_{OPT,\ominus}|$, which together with the observation that $OPT(G', G_t) \geq |E_\ominus|$ yields the claim.

To upper bound $ALG_2(G_s, G_t)$, we analyze the number of primitives applied in each step of the second part of the approximation algorithm. Of course, no primitive is applied in Step 5. The connections required in Step 6 can be created in a similar fashion as in Step 2 (see the proof of Lemma 9: For each tree T, we proceed top-down in the T rooted at r_T again. Here, each intermediate node u checks whether $u = s(e)$ for some $e \in E_\ominus$. If so, it introduces r_T to all relevant children (here a node v is *relevant* if $ST(v)$ contains a node w such that $w = s(e')$ for some $e' \in E_\ominus$). Otherwise, it introduces r_T to all but one relevant children and delegates

it to the remaining one. In the end, for every edge $e \in E_\ominus$, $s(e)$ has an edge to r_T, the number of edges in the graph has increased by at most $|E_\ominus|$, and the process involved at most $|F_{ALG,\ominus}|$ applications of primitives. In Step 7, clearly exactly $|E_\ominus|$ edges have to be delegated. Step 8 is similar to Step 4 and for analogous reasons requires at most $|F_{ALG,\ominus}|$ delegations and at most $2|E_\ominus|$ fusions (recall that up to $|E_\ominus|$ edges were added in Step 6 and the edges delegated in Step 7 have to be removed as well). All in all, Step 6, Step 7 and Step 8 of the algorithm involve at most $|F_{ALG,\ominus}|$, $|E_\ominus|$ and $|F_{ALG,\ominus}| + 2|E_\ominus|$ applications of primitives, respectively, which yields: $ALG_2(G_s, G_t) \leq 2|F_{ALG,\ominus}| + 3|E_\ominus| \leq 4|F_{OPT,\ominus}| + 3|E_\ominus|$ (since $F_{ALG,\ominus}$ is a 2-approximation of $F_{OPT,\ominus}$).

To lower bound the value of $OPT(G', G_t)$, assume for contradiction that there is a computation C with initial graph G' and final graph G_s of length $L < |F_{OPT,\ominus}| - |E_\ominus|$. Let $G_s = G_1 \Rightarrow G_2 \Rightarrow \cdots \Rightarrow G_L$ be the sequence of graphs of this computation. Similar to the proof of Lemma 9, for every $\{u, v\} \in E_\ominus$, we create a path from u to v, but this time we start with $P^1_{u,v} := (u, v)$ and consider the graphs in increasing order: For $i \in \{2, \ldots, L\}$, if $P^{i-1}_{u,v}$ exists in G_i, $P^i_{u,v} := P^{i-1}_{u,v}$. Otherwise since G_i is the result of a single application of a primitive to G_{i-1}, there is exactly one edge $\{x, y\}$ in $P^{i-1}_{u,v}$ that exists in G_{i-1} but not in G_i and this edge must have been delegated by either x or y to some node w. In the following denote the node that applied the Delegation by z and denote by $\overline{z}$ the other node from $\{x, y\}$. In G_{i-1}, z must share an edge with w and this edge still exists in G_i (for only one primitive is applied in the transition from G_{i-1} to G_i). Since $\{z, \overline{z}\}$ was delegated to w, in G_i the edge $\{w, \overline{z}\}$ exists in G_i. Thus, let $P^i_{u,v}$ be $P^{i-1}_{u,v}$ with (x, y) replaced by (x, w, y) and observe that $P^i_{u,v}$ exists in G_i. Eventually, we obtain a path $P^L_{u,v}$ that exists in G_t. Define $F^i := \bigcup_{\{u,v\} \in E_\ominus} E(P^i_{u,v})$ (where $E(P)$ is the set of all edges on the path P) for all $i \in \{1, \ldots, L\}$, and note that F^L represents a solution to the USF with input G_t and $E_\ominus$ as the set of nodes. Furthermore, for an arbitrary $i \in \{1, \ldots, L-1\}$, note that $|F^{i+1}| \leq |F^i| + 1$ because there is at most one edge $\{x, y\}$ that exists in G_i but not in G_{i+1} and thus causes the replacement of (x, y) by (x, w, y) for some fixed node w for all paths that contain (x, y) as a sub-path. This yields that $|F^L| \leq |F^1| + L = |E_\ominus| + L$ because $F^1 = E_\ominus$ (note the definition of F^1. By the assumption that $L < |F_{OPT,\ominus}| - |E_\ominus|$, we obtain $|F^L| < |F_{OPT,\ominus}|$, which represents a contradiction. ◀

4 Conclusion

We proposed a set of primitives for topology adaptation that a server may use to adapt the network topology into any desired (weakly) connected state but at the same time cannot use to disconnect the network or to introduce new nodes into the system. So far, we only assumed that the server could act maliciously but that the participants of the network are honest and correct, i.e., they refuse any graph transformation commands beyond the four primitives. What, however, if some participants also behave in a malicious manner? Is it still possible to avoid Eclipse or Sybil attacks? It seems that in this case the only measure that would help is to form quorums of nodes that are sufficiently large so that at least one node in each quorum is honest.

Besides these security-related aspects, our results give rise to additional questions: For example, does the NP-hardness apply to any set of local primitives, or is there a set of local primitives that can transform arbitrary initial graphs much faster into arbitrary final graphs than the set considered in this work? Furthermore, is it possible to obtain decentralized versions of the algorithms presented in Section 3, and, if so, what is their competitiveness when compared to the centralized ones?

References

1 Ajit Agrawal, Philip N. Klein, and R. Ravi. When Trees Collide: An Approximation Algorithm for the Generalized Steiner Problem on Networks. *SIAM J. Comput.*, 24(3):440–456, 1995. doi:10.1137/S0097539792236237.

2 Susanne Albers, Stefan Eilts, Eyal Even-Dar, Yishay Mansour, and Liam Roditty. On nash equilibria for a network creation game. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 89–98, 2006.

3 Noga Alon, Erik D. Demaine, Mohammad Taghi Hajiaghayi, and Tom Leighton. Basic Network Creation Games. *SIAM J. Discrete Math.*, 27(2):656–668, 2013. doi:10.1137/090771478.

4 Marc Andries, Gregor Engels, Annegret Habel, Berthold Hoffmann, Hans-Jörg Kreowski, Sabine Kuske, Detlef Plump, Andy Schürr, and Gabriele Taentzer. Graph Transformation for Specification and Programming. *Sci. Comput. Program.*, 34(1):1–54, 1999. doi:10.1016/S0167-6423(98)00023-9.

5 James Aspnes and Yinghua Wu. O(logn)-Time Overlay Network Construction from Graphs with Out-Degree 1. In *Proceedings of the 11th International Conference on Principles of Distributed Systems, (OPODIS '07)*, pages 286–300, 2007.

6 Azrina Abd Aziz, Y. Ahmet Sekercioglu, Paul G. Fitzpatrick, and Milosh V. Ivanovich. A Survey on Distributed Topology Control Techniques for Extending the Lifetime of Battery Powered Wireless Sensor Networks. *IEEE Communications Surveys and Tutorials*, 15(1):121–144, 2013. doi:10.1109/SURV.2012.031612.00124.

7 Andrew Berns, Sukumar Ghosh, and Sriram V. Pemmaraju. Building self-stabilizing overlay networks with the transitive closure framework. *Theor. Comput. Sci.*, 512:2–14, 2013.

8 Davide Bilò, Luciano Gualà, Stefano Leucci, and Guido Proietti. Locality-Based Network Creation Games. *TOPC*, 3(1):6:1–6:26, 2016. doi:10.1145/2938426.

9 Stephen A. Cook. The Complexity of Theorem-proving Procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM. doi:10.1145/800157.805047.

10 Andreas Cord-Landwehr, Martina Hüllmann, Peter Kling, and Alexander Setzer. Basic Network Creation Games with Communication Interests. In *Algorithmic Game Theory - 5th International Symposium, SAGT 2012, Barcelona, Spain, October 22-23, 2012. Proceedings*, pages 72–83, 2012. doi:10.1007/978-3-642-33996-7_7.

11 Erik D. Demaine, MohammadTaghi Hajiaghayi, Hamid Mahini, and Morteza Zadimoghaddam. The price of anarchy in network creation games. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing, PODC 2007, Portland, Oregon, USA, August 12-15, 2007*, pages 292–298, 2007. doi:10.1145/1281100.1281142.

12 Shlomi Dolev and Ronen I. Kat. HyperTree for self-stabilizing peer-to-peer systems. *Distributed Computing*, 20(5):375–388, 2008.

13 Hartmut Ehrig, Hans-Jörg Kreowski, Ugo Montanari, and Grzegorz Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 3: Concurrency, Parallelism, and Distribution*. World Scientific, 1999.

14 Alex Fabrikant, Ankur Luthra, Elitza N. Maneva, Christos H. Papadimitriou, and Scott Shenker. On a network creation game. In *Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing, PODC 2003, Boston, Massachusetts, USA, July 13-16, 2003*, pages 347–351, 2003. doi:10.1145/872035.872088.

15 Michael Feldmann, Christina Kolb, Christian Scheideler, and Thim Strothmann. Self-Stabilizing Supervised Publish-Subscribe Systems. In *2018 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2018, Vancouver, BC, Canada, May 21-25, 2018*, pages 1050–1059, 2018. doi:10.1109/IPDPS.2018.00114.

16 M. Goemans and D. Williamson. A General Approximation Technique for Constrained Forest Problems. *SIAM Journal on Computing*, 24(2):296–317, 1995. doi:10.1137/S0097539793242618.

17 Martin Groß, Anupam Gupta, Amit Kumar, Jannik Matuschke, Daniel R. Schmidt, Melanie Schmidt, and José Verschae. A Local-Search Algorithm for Steiner Forest. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, volume 94 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:17, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITCS.2018.31.

18 Anupam Gupta and Amit Kumar. Greedy Algorithms for Steiner Forest. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 871–878, New York, NY, USA, 2015. ACM. doi:10.1145/2746539.2746590.

19 Yair Halevi and Yishay Mansour. A Network Creation Game with Nonuniform Interests. In *Internet and Network Economics, Third International Workshop, WINE 2007, San Diego, CA, USA, December 12-14, 2007, Proceedings*, pages 287–292, 2007. doi:10.1007/978-3-540-77105-0_28.

20 Reiko Heckel. Graph Transformation in a Nutshell. *Electr. Notes Theor. Comput. Sci.*, 148(1):187–198, 2006. doi:10.1016/j.entcs.2005.12.018.

21 Riko Jacob, Andréa W. Richa, Christian Scheideler, Stefan Schmid, and Hanjo Täubig. SKIP$^+$: A Self-Stabilizing Skip Graph. *J. ACM*, 61(6):36:1–36:26, 2014. doi:10.1145/2629695.

22 Riko Jacob, Stephan Ritscher, Christian Scheideler, and Stefan Schmid. Towards higher-dimensional topological self-stabilization: A distributed algorithm for Delaunay graphs. *Theor. Comput. Sci.*, 457:137–148, 2012.

23 Kamal Jain. A Factor 2 Approximation Algorithm for the Generalized Steiner Network Problem. *Combinatorica*, 21(1):39–60, 2001. doi:10.1007/s004930170004.

24 Kishore Kothapalli and Christian Scheideler. Supervised Peer-to-Peer Systems. In *8th International Symposium on Parallel Architectures, Algorithms, and Networks, ISPAN 2005, December 7-9. 2005, Las Vegas, Nevada, USA*, pages 188–193, 2005. doi:10.1109/ISPAN.2005.81.

25 Andreas Koutsopoulos, Christian Scheideler, and Thim Strothmann. Towards a universal approach for the finite departure problem in overlay networks. *Inf. Comput.*, 255:408–424, 2017. doi:10.1016/j.ic.2016.12.006.

26 Leonid Anatolevich Levin. Universal sequential search problems. *Problemy Peredachi Informatsii*, 9(3):115–116, 1973.

27 Mo Li, Zhenjiang Li, and Athanasios V. Vasilakos. A Survey on Topology Control in Wireless Sensor Networks: Taxonomy, Comparative Study, and Open Issues. *Proceedings of the IEEE*, 101(12):2538–2557, 2013. doi:10.1109/JPROC.2013.2257631.

28 Chih-Long Lin. Hardness of Approximating Graph Transformation Problem. In *Algorithms and Computation, 5th International Symposium, ISAAC '94, Beijing, P. R. China, August 25-27, 1994, Proceedings*, pages 74–82, 1994. doi:10.1007/3-540-58325-4_168.

29 Rizal Mohd Nor, Mikhail Nesterenko, and Christian Scheideler. Corona: A stabilizing deterministic message-passing skip list. *Theor. Comput. Sci.*, 512:119–129, 2013. doi:10.1016/j.tcs.2012.08.029.

30 Ram Ramanathan and Regina Hain. Topology Control of Multihop Wireless Networks Using Transmit Power Adjustment. In *Proceedings IEEE INFOCOM 2000, The Conference on Computer Communications, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Reaching the Promised Land of Communications, Tel Aviv, Israel, March 26-30, 2000*, pages 404–413, 2000. doi:10.1109/INFCOM.2000.832213.

31 Grzegorz Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*. World Scientific, 1997.

32 Christian Scheideler and Alexander Setzer. On the Complexity of Local Graph Transformations (full version), 2019. arXiv:1904.11395.

33 Stefan Schmid, Chen Avin, Christian Scheideler, Michael Borokhovich, Bernhard Haeupler, and Zvi Lotker. SplayNet: Towards Locally Self-Adjusting Networks. *IEEE/ACM Trans. Netw.*, 24(3):1421–1433, 2016. doi:10.1109/TNET.2015.2410313.

Network Investment Games with Wardrop Followers

Daniel Schmand
Goethe University Frankfurt, Germany
schmand@em.uni-frankfurt.de

Marc Schröder
RWTH Aachen University, Germany
marc.schroeder@oms.rwth-aachen.de

Alexander Skopalik
University of Twente, Netherlands
a.skopalik@utwente.nl

Abstract

We study a two-sided network investment game consisting of two sets of players, called providers and users. The game is set in two stages. In the first stage, providers aim to maximize their profit by investing in bandwidth of cloud computing services. The investments of the providers yield a set of usable services for the users. In the second stage, each user wants to process a task and therefore selects a bundle of services so as to minimize the total processing time. We assume the total processing time to be separable over the chosen services and the processing time of each service to depend on the utilization of the service and the installed bandwidth. We provide insights on how competition between providers affects the total costs of the users and show that every game on a series-parallel graph can be reduced to an equivalent single edge game when analyzing the set of subgame perfect Nash equilibria.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory and mechanism design; Theory of computation → Network games

Keywords and phrases Network Investment Game, Wardrop Equilibrium, Subgame Perfect Nash Equilibrium

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.151

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Related Version A full version of the paper is available at [31], `http://arxiv.org/abs/1904.10417`.

1 Introduction

With the increasing availability of a fast and reliable internet connection, the demand for and the importance of cloud-computing services is heavily growing. That is why large IT-companies such as Amazon, Google or Microsoft have been investing massively in the improvement of their cloud services in recent years. New high-speed cables are being placed, high performance hardware has been installed, new software has been developed, and machine numbers have been scaled up [21]. Interestingly, prices for these services are typically charged per usage time (see for example, Amazon Web Services [3], Microsoft Azure [26] and Google Cloud [17]). This might a-priory lead to a situation, where a non-optimal infrastructure takes longer to process the users' tasks, and thus even leads to a higher income for the corresponding provider. This paper addresses this issue by means of a theoretical model and provides some results on how competition between providers helps to prevent this behavior of providers.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 151; pp. 151:1–151:14


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

We study the problem by means of a two-stage game, which we call a network investment game. In the first stage, a set of providers invests in bandwidths of services that are used by users in the second stage to process their task. These services could be different types of on-demand cloud computing, storage, data-base, security or media services. We assume that these services are either complements or substitutes to each other. This implies that the set of feasible bundles of services for the users can be represented by means of source-sink paths in a series-parallel network. Two services that are complementary can be modeled as two edges that are connected in series, whereas two services that are substitutes to each other can be modeled by means of two parallel edges. A similar idea has been used by Correa et al. in [14]. We call this underlying graph the technology graph. However, before a user can use a particular service, there must be a provider that has invested in bandwidth for that particular service. In other words, the investments of the providers induce a subgraph, the set of usable services, of the original technology graph that can be used by the users to process their task. Each user selects a bundle of services that is needed to perform the user's task at minimum total processing time. The total processing time of a service depends on the quantity of users that chooses the particular service and the installed bandwidth of the providers. We assume that each user has a given willingness to pay for taking part in the game and is only present if the total costs do not exceed this value. Given that users impose externalities on each other, we assume that users in the second stage choose their bundle of services according to a Wardrop equilibrium [33] in the corresponding graph of usable services.

Given that prices for services are typically charged per usage time, we use the users' costs as a proxy for the total revenue of the providers investing in a service. This total revenue is shared proportionally among those providers investing in the service. Proportional sharing is used in many different settings, e.g., in cooperative game theory [28] and in the context of congestion games [27]. We assume that the providers' aim is to maximize profit, which depends on the amount of users using their services and the total amount invested. Intuitively, little investment in a service implies a high processing time and thus a high price paid by users, but only a small fraction of users will choose the service. Too much investment might lead to a large fraction of users, but only for a low price.

1.1 Contribution

We are interested in how competition between selfish providers affects the choices made by the users in the network investment game. More formally, we study the existence and uniqueness of subgame perfect Nash equilibria in which in the first stage, providers maximize profit and in the second stage, users choose a Wardrop equilibrium given the strategies of the providers in the first stage. We then consider the performance of equilibria in terms of the price of anarchy [23]. The price of anarchy measures the difference in performance between the worst subgame perfect Nash equilibrium and the social optimum. In fact, for all instances we consider, the performance of all subgame perfect Nash equilibria is the same and we are able to provide an exact characterization of this performance.

Our main result, Theorem 3.2, provides a simplification of the structure of equilibrium investments for series-parallel networks. We focus on series-parallel networks as this represents bundles of services consisting of complements and substitutes. We show that we can restrict attention to investments in which each provider invests in shortest paths (with respect to the number of services). This means that providers have no incentive to diversify their portfolio and invest in complements, nor to invest in services that can be replaced by less services. The result has some interesting implications. When being interested in either a subgame

perfect Nash equilibrium, the social optimum, or combining both in the price of anarchy, it is without loss of generality to assume that the network consists of a single edge. This greatly simplifies the remaining analysis.

We then study the quality of subgame perfect Nash equilibria for different classes of users. In Section 4, we assume that all users have a fixed common willingness to pay. We precisely characterize the society's surplus and the providers' profits both in a corresponding optimal solution and in every subgame perfect Nash equilibrium. It turns out that the price of anarchy with respect to society's surplus can be bounded by a constant, whereas the ratio with respect to providers' profits can grow linearly with the number of providers. In Section 5, we redo the same analysis for users that have different willingnesses to pay and obtain similar results as for the fixed reservation value case.

1.2 Related work

Our model is closely related to several different classes of recently studied models. There is a close connection with network design games, as introduced by Anshelevich et al. [5] and extended by Anshelevich et al. [4]. In these games, selfish agents want to connect a personal source to sink by means of a path, and they share the costs of an edge in case multiple agents use that edge in their path. In network investment games, providers do not share the cost of an edge, but rather the revenue from the users using that edge. As a result, we obtain that providers want to invest in paths in equilibrium, an assumption that is not made a-priori.

In network investment games, we assume that users in the second stage behave as in a non-atomic network congestion game. The idea behind congestion games is that selfish users try to minimize their cost in an already existing network. The best-known solution concept for these games is the Wardrop equilibrium [33]. Beckmann et al. [7] and Dafermos and Sparrow [16] proved some structural results of this equilibrium concept. Roughgarden and Tardos [30] were the first to obtain a bound on the price of anarchy for non-atomic routing games.

Recently, starting with Acemoglu and Ozdaglar [1], several authors considered a two-stage game in which edge owners compete for users by means of tolling. In those games, the users in the second stage also choose a Wardrop equilibrium. Acemoglu and Ozdaglar [1] only considered parallel networks, which was then generalized to parallel-serial networks by Acemoglu and Ozdaglar [2]. Ozdaglar [29] and Hayrapetyan et al. [20] allowed for elastic users. Johari et al. [22] studied an extension that includes entry and investments decisions. Recently, Correa et al. [13] and Harks et al. [18] considered the game in which a central authority is allowed to impose price caps. In the former paper, caps are allowed to be different on different edges, in the latter the cap is uniform. Almost all of the above models, with the exception of Acemoglu and Ozdaglar [2], impose the simplifying assumption that the topology of the network is restricted to be parallel. We allow for arbitrary series-parallel graphs and do not impose any restriction on the strategy space of the providers, that is, every provider is allowed to invest in every edge of the network.

A slightly different, but closely related model is the game analyzed by Correa et al. [14]. Here each edge in a series-parallel network represents a producer that competes by selecting a markup that then determines a price function. The main difference to our work is that producers are identified by an edge and thus do not have the possibility to compete on other edges.

Leader-follower models as the previous ones are also known as Stackelberg games [32]. Recently, lots of attention has been placed on Stackelberg pricing games. Starting with Labbé et al. [24], who considered the problem in which a leader sets prices on a subset of edges and

the follower chooses a shortest path, several authors generalized to different combinatorial problems for the follower. For example, Cardinal et al. [12] studied the Stackelberg minimum spanning tree problem, and Bilò el al. [8] and later Cabello [11] investigated the Stackelberg shortest path tree problem. Balcan et al. [6] and Briest et al. [10] considered the power of single-price strategies. Böhnlein et al. [9] extended the analysis of this simple algorithm beyond the combinatorial setting.

Lastly, there is a close connection to some of the traditional models of competition in economics, like Cournot competition [15]. We refer to Harks and Timmermans [19] for some other recent work connecting atomic splittable congestion games to multimarket Cournot oligopolies.

2 Preliminaries

We consider a two-sided market in which users purchase bundles of services. The set of feasible bundles is specified by a technology graph $G_{st} = (V, E)$, which is a directed series-parallel network with source $s \in V$ and sink $t \in V$. Each edge corresponds to a service. Every s-t path in G_{st} describes a feasible bundle of services. Note that serial edges in G_{st} correspond to complements, whereas parallel edges correspond to substitutes.

We model such a market as a two-stage game. In the first stage, a set of providers chooses to invest into services represented by edges. In the second stage, the users select bundles of services or paths in G_{st} given the investments of the providers in the first stage.

The amount of investment affects the speed of the services. We assume for simplicity that prices for each service are charged automatically on a per-time basis. That is, the price function of an edge directly depends on the chosen investments into the corresponding service in the first stage and the current load induced by the users in the second stage. These two quantities highly influence the time that is needed to finish a job on a cloud computing device. Our choice of the price functions and the model of network investment games are motivated by time-dependent prices that are implemented in Amazon Web Services [3], Microsoft Azure [26] and Google Cloud [17] for using on-demand virtual machines.

We assume for simplicity that each user is of infinitesimal size which allows us to model the second stage as a non-atomic routing game. However it is important to note that the overall demand is not fixed, but depends on the total price the users have to pay for the services. The demand thus depends on the investment of the providers.

More formally, a *network investment game* is given by a tuple $\mathcal{G} = (G_{st}, N, u)$, where $G_{st} = (V, E)$ is a directed series-parallel network with source $s \in V$ and sink $t \in V$, $N = \{1, \dots, n\}$ is the set of providers, and $u : \mathbb{R}_+ \to \mathbb{R}_+$ is the reservation function. Reservation functions model the users' willingness to pay and impose elastic demand. This concept has already been used in the work of Beckmann, McGuire, and Winsten [7]. Intuitively, there is a total demand of x in the game if the total price for bundles of services under demand x is not larger than $u(x)$. We assume that u is non-increasing.

The Provider's Strategies. Each provider $i \in N$ chooses an investment $b_{i,e} \geq 0$ for every edge $e \in E$. Let $b_i = (b_{i,e})_{e \in E}$ be the investments of provider i. Given an investment matrix $b = (b_i)_{i \in N}$, the total investment on edge e is $b_e = \sum_{i \in N} b_{i,e}$. Given $b \in \mathbb{R}_+^{N \times E}$, we define the set of *relevant edges* by $E^+(G_{st}) = \{e \in E(G_{st}) \mid b_e > 0\}$. The price c_e for a user using an edge e, i.e., a service, is an increasing function of the amount of users f_e using edge e and defined by

$$c_e(f_e, b_e) = \begin{cases} \frac{f_e}{b_e} & e \in E^+(G_{st}), \\ \infty & \text{otherwise.} \end{cases}$$

Note that this cost function is a very simple model for prices that depend on the total load of a service and the total investment. Here, we use the simplifying assumption that the time that is needed to complete a service request scales linearly with the load on the machines and inverse linearly with the investment.

The User's Actions. Let $\mathcal{P}(G_{st})$ denote the set of s-t paths, and let $\mathcal{P}^*(G_{st})$ denote the set of s-t paths with minimum number of edges in G_{st}. We model the behavior of the users by a flow in G_{st}. The flow is defined to be a non-negative vector $f(G_{st}) = (f_P)_{P\in\mathcal{P}(G_{st})}$. Note that we stick to the path-definition of flows, because we assume strong flow conservation. For a flow $f(G_{st})$ and an edge $e \in E(G_{st})$, let $f_e = \sum_{P\ni e} f_P$ denote the amount of flow on edge e. Let $|f(G_{st})| = \sum_{P\in\mathcal{P}(G_{st})} f_P$ be the total amount of users that are routed through the network under $f(G_{st})$.

For a given investment matrix b, we call some flow $f(G_{st})$ a *Wardrop equilibrium* if for all $P, P' \in \mathcal{P}(G_{st})$ with $f_P > 0$, we have

$$\sum_{e\in P} c_e(f_e, b_e) \leq \sum_{e\in P'} c_e(f_e, b_e) \text{ and } \sum_{e\in P} c_e(f_e, b_e) < \infty.$$

In a network investment game, we assume that both the providers and the users act selfishly. For simplicity, we assume that users are infinitesimal small and thus, a Wardop flow arises in the second stage. We will remark below that, for a given investment matrix and a given non-increasing reservation function, the arising Wardrop flow is uniquely defined, which is necessary for our model to be well-defined. Given the investment matrix b, we denote the corresponding Wardrop flow by $f(G_{st}, b)$. If G_{st} and/or b is clear from the context, we often omit the dependency on G_{st} and/or b in order to improve readability. If s and t are clear from the context, we sometimes write G instead of G_{st}.

The Provider's Profits. The profit of provider $i \in N$ depends on the associated Wardrop flow f, the investment matrix b and the actual cost for the users $c_e(f_e, b_e)$. We define it by

$$\pi_i(b) = \sum_{e\in E^+} \frac{b_{i,e}}{b_e} \cdot f_e \cdot c_e(f_e, b_e) - \sum_{e\in E^+} b_{i,e} = \sum_{e\in E^+} b_{i,e} \cdot \frac{f_e^2}{b_e^2} - \sum_{e\in E^+} b_{i,e}.$$

where $\frac{b_{i,e}}{b_e}$ is the proportional share of provider $i \in N$ and $\sum_{e\in E^+} b_{i,e}$ her investment costs. This profit function is motivated by that fact that each user pays a load-dependent price for using a service of the provider, which is proportionally divided among those providers investing in the service, and the provider has some cost for installing bandwidth for the services. We consider the very simple model that investment costs are linear and the profit is given by the payment of the users minus the investment costs.

We call an investment matrix $b \in \mathbb{R}_+^{N\times E}$ a *subgame perfect Nash equilibrium* if for all $i \in N$ and all $b'_i \in \mathbb{R}_+^E$,

$$\pi_i(b) \geq \pi_i(b'_i, b_{-i}).$$

Given $b \in \mathbb{R}_+^{N\times E}$, we call b'_i a *better response* for provider $i \in N$ if $\pi_i(b'_i, b_{-i}) > \pi_i(b)$. In a subgame perfect Nash equilibrium, no provider has a better response. We call a better response b'_i *demand preserving* if $|f(b'_i, b_{-i})| = |f(b)|$.

The Social Welfare. or social surplus, is traditionally defined as the sum of the providers' surplus and the user's surplus. In our model, similarly as in Hayrapetyan et al. [20], this boils down to,

$$SW(b) = \sum_{i \in N} \pi_i(b) + \int_0^{|f|} u(x)\, dx - \sum_{e \in E^+} f_e c_e(f_e, b_e) = \int_0^{|f|} u(x)\, dx - \sum_{e \in E} b_e.$$

Note that payments do not appear in the social welfare, as they are transfers from users to providers. An investment matrix b^* is called a *social optimum* if b^* maximizes the social welfare.

For a given instance $\mathcal{G}$, let $\mathcal{E}(\mathcal{G})$ denote the set of subgame perfect Nash equilibria. Notice that $\mathcal{E}(\mathcal{G})$ might be empty. A natural question is to quantify the loss in social welfare due to competition. Typically, researchers provide upper bounds for the loss in efficiency. We are able to precisely bound this inefficiency for any given instance. To this end, we define the *price of anarchy* and *price of stability* for a given instance as

$$PoA(\mathcal{G}) = \frac{\sup\limits_{b^* \in \mathbb{R}_+^N} SW(b^*)}{\inf\limits_{b \in \mathcal{E}(\mathcal{G})} SW(b)}, \text{ and } PoS(\mathcal{G}) = \frac{\sup\limits_{b^* \in \mathbb{R}_+^N} SW(b^*)}{\sup\limits_{b \in \mathcal{E}(\mathcal{G})} SW(b)}.$$

We assume that $0/0 = 1$ and $PoA(\mathcal{G}) = PoS(\mathcal{G}) = \infty$ if $\mathcal{E}(\mathcal{G})$ is empty.

A second natural question is to quantify the loss in total profit due to competition. We call this the *providers' price of anarchy* and *providers' price of stability*, and define it by

$$PPoA(\mathcal{G}) = \frac{\sup\limits_{b^* \in \mathbb{R}_+^N} \sum_{i \in N} \pi_i(b^*)}{\inf\limits_{b \in \mathcal{E}(\mathcal{G})} \sum_{i \in N} \pi_i(b)}, \text{ and } PPoS(\mathcal{G}) = \frac{\sup\limits_{b^* \in \mathbb{R}_+^N} \sum_{i \in N} \pi_i(b^*)}{\sup\limits_{b \in \mathcal{E}(\mathcal{G})} \sum_{i \in N} \pi_i(b)}.$$

As before, we assume that $0/0 = 1$ and $PPoA(\mathcal{G}) = PPoS(\mathcal{G}) = \infty$ if $\mathcal{E}(\mathcal{G})$ is empty.

We start with some simple observations on Wardrop equilibria for this model. To calculate the flow that models the users, we distinguish between two cases. Either there exists a s-t path $P \in \mathcal{P}$ such that $b_e > 0$ for all $e \in P$, or not. Put differently, either there is a path P such that for all $e \in P$ we have $e \in E^+(G_{st})$, or all paths have investment 0 for at least one edge. If there is no s-t path with strictly positive investment on every edge, the Wardrop equilibrium is the empty flow with $|f| = 0$. If there is some $P \in \mathcal{P}$ with $b_e > 0$ on all edges $e \in P$, we obtain the following results.

▶ Remark 2.1 ([7, 16]). If f is a Wardrop equilibrium, then there is a constant $c_b \geq 0$ such that $\sum_{e \in P} c_e(f_e, b_e) = c_b$ whenever $P \in \mathcal{P}$ with $f_e > 0$ for all $e \in P$, and $\sum_{e \in P} c_e(f_e, b_e) \geq c_b$ otherwise.

▶ Remark 2.2 ([7, 16]). For two Wardrop flows f, f' with $|f| = |f'|$ in a network with strictly increasing cost functions on every edge, we have that $f_e = f'_e$ for all $e \in E$.

Note that, by Remarks 2.1 and 2.2, we have that the cost per user in the network investment game c_b, as defined in Remark 2.1, only depends on the total demand $|f|$ for all Wardrop flows f and a fixed b. Let us focus on this dependence and write $c_b(|f|)$ instead of c_b. It turns out that $c_b(|f|)$ is in fact strictly increasing in $|f|$. The following proposition is due to Lin, Roughgarden, and Tardos [25]. However, in their variant of the proposition they only assume non-decreasing cost functions and show that $c_b(|f|)$ is non-decreasing in $|f|$. Since we have strictly increasing cost functions, we modify their proof in the full version [31] such that we get a slightly stronger statement.

▶ **Proposition 2.3** ([25]). *$c_b(|f|)$ is strictly increasing in $|f|$.*

We note that we can extend Proposition 2.3 and show that for network investment games, $c_b(|f|)$ is even linear in $|f|$. The proof uses the series-parallel structure of the graph and can be found in the full version [31].

▶ **Proposition 2.4.** *Given a vector of investment levels b, $c_b(|f|)$ is a linear function of $|f|$.*

The above two propositions are very important for the setup of network investment games. We assume that demand directly depends on the actual cost for the users, and thus indirectly on b. In order to do so, we use that the costs for each user are given by a linear function $c_b(|f|)$. Combining this with the reservation function u stating that an amount of $u(x)$ has a willingness to pay at least x, we define

$$|f| = \inf\{x \in \mathbb{R}_+ \mid u(x) \leq c_b(x)\}.$$

Note that this infimum is well defined, since u is non-increasing and $c_b(\cdot)$ is strictly increasing due to Proposition 2.3. Furthermore, let us illustrate the dependence of $|f|$ on b by the following observation. An increase of some b_e for fixed f induces a change in the corresponding edge cost $c_e(f_e, b_e)$ and weakly decreases the cost of the flow f. Since Wardrop flows are optimal flows for linear cost functions, this weakly decreases the value of $c_b(x)$ for any fixed x. Overall, one can show that a change of b changes the slope of $c_b(x)$ in the definition above, which then might result in a different demand.

3 Characterization of Equilibria

In this section, we derive some important properties of subgame perfect Nash equilibria. First, we prove our main result that in every subgame perfect Nash equilibrium, every provider only invests in shortest (with respect to the number of edges) paths. The implications of this result are quite significant and twofold. First, it implies a nice characterization of markets that can be modeled by means of a network investment game. Second, the result simplifies the further analysis of the game significantly. As long as we are interested in equilibrium investments, the analysis gets notably easier. Instead of considering a strategy space in which every provider is allowed to invest in every edge separately, we can restrict attention to strategies in which providers invest in shortest paths. We will heavily use this characterization in the later sections.

▶ **Definition 3.1.** *We call a strategy b_i a* path strategy *of provider i if there exist values $(b_{i,P})_{P\in\mathcal{P}}$ with $b_{i,P} \geq 0$ such that $b_{i,e} = \sum_{P:e\in P} b_{i,P}$ for all $e \in E$.*

Note that the definition ensures that we can always decompose a path strategy $(b_{i,e})_{e\in E}$ of a provider $i \in N$ defined on edges into a path decomposition. Thus, when considering path strategies, we can work with the path decomposition of investments, instead of the investments defined on edges. If all providers' investments are path strategies, it is immediately clear that we can also define $b_P = \sum_{i\in N} b_{i,P}$ for all $P \in \mathcal{P}$ and $|b| = \sum_{P\in\mathcal{P}} b_P$.

Note that, when using the term *shortest paths*, we always refer to shortest paths with respect to the number of edges in the path. Now, we are able to state and prove our main theorem of this section.

▶ **Theorem 3.2.**

(i) *In every subgame perfect Nash equilibrium, every provider $i \in N$ chooses a path strategy on shortest paths.*

(ii) *Let $i \in N$. If every provider $j \in N \setminus \{i\}$ chooses a path strategy on shortest paths then the following holds: For every strategy b_i of provider i, there is a path strategy on shortest paths b_i' inducing the same amount of flow and at least the same profit for provider i.*

For the proof of Theorem 3.2, we need the following two lemmas. In the first lemma we observe that in equilibrium a provider cannot choose positive investments on two paths of different length. If so, there is always a better response that only invests in the shorter of the two paths.

▶ **Lemma 3.3.** *Let (G, s, t) be a series-parallel graph that is a parallel composition of series-parallel graphs (G_1, s, t) and (G_2, s, t). Let b be a vector of path strategies of paths that all are either shortest s-t paths in G_1 with length k or shortest s-t paths in G_2 with length ℓ. If $k < \ell$, and $b_{i,P_2} > 0$ for some provider $i \in N$ and $P_2 \in \mathcal{P}^*(G_2)$, then there is a demand preserving better response for i with $b_{i,P_2} = 0$.*

Next we turn to graphs that are a series composition of two subgraphs. Assuming that every provider's strategy can be decomposed into two path strategies for the two subgraphs, we show that in equilibrium they must invest the same value in both subgraphs. Furthermore, if all other providers play shortest path strategies, for all strategies of a provider, the provider always has an at least as good strategy in shortest paths.

▶ **Lemma 3.4.** *Let (G, s, t) be a series-parallel graph that is a series composition of series-parallel graphs (G_1, s, v) and (G_2, v, t). Let b be a vector of investments $(b_{i,e})_{i \in N, e \in E}$ that can be partitioned into $(b^1_{i,e})_{i \in N, e \in E(G_1)}$ and $(b^2_{i,e})_{i \in N, e \in E(G_2)}$ such that b^1 and b^2 are shortest path strategies in their corresponding graphs G_1 and G_2, respectively.*

(i) *If $\sum_{P \in \mathcal{P}^*(G_1)} b^1_{i,P} \neq \sum_{P \in \mathcal{P}^*(G_2)} b^2_{i,P}$ for some provider $i \in N$, then there is a demand preserving better response for some $j \in N$.*

(ii) *If, additionally, $\sum_{P \in \mathcal{P}^*(G_1)} b^1_{m,P} = \sum_{P \in \mathcal{P}^*(G_2)} b^2_{m,P}$ for all providers $m \neq i$, then the following holds. For every strategy b_i there is a shortest path strategy b_i' with $|f(b_{-i}, b_i)| = |f(b_{-i}, b_i')|$ and at least the same profit for provider i, i.e., $\pi_i(b_{-i}, b_i) \leq \pi_i(b_{-i}, b_i')$.*

Now, we are ready to prove Theorem 3.2.

Proof of Theorem 3.2. The main difficulty in proving statements on strategy changes of providers lies in the fact that the users' costs and, hence, demand and flow might change. However, we are able to circumvent the dependency on the reservation function by only considering demand preserving better responses. In fact, we prove the first statement of the theorem by showing the following stronger statement. *(i') If there is a provider $i \in N$ such that b_i is not a path strategy on shortest paths, then there is a provider $j \in N$ with a demand preserving better response.* Note that statement (i') implies part (i) of the theorem. We show statement (i') and part (ii) of the theorem by induction on the structure of the series-parallel graph G. For the base case, a graph with a single edge, both statements are trivially fulfilled. For the induction step, assume the statements are true for series-parallel graphs G_1 and G_2, and we will argue that the statements are true for a new series-parallel graph that either arises from a parallel composition or a series composition of G_1 and G_2.

For (i'), observe the following: If there is a provider i such that b_i is not a path strategy in one of the two subgraphs G_1 or G_2, we can assume w.l.o.g. that it is not a path strategy in G_1. Then there is a demand preserving better response in G_1 for some provider j. This is

also a demand preserving better response for j in G, since it is demand preserving in G_1, and thus does not change any flow in G_2, and thus any profit in G_2. So we can assume that the investments b_i are shortest paths within G_1 and within G_2 for all $i \in N$.

Case 1: Parallel Composition. Assume G is the result of a parallel composition of series-parallel networks G_1 and G_2. For proving (i'), denote the length of the shortest paths in G_1 by k and the length of the shortest paths in G_2 by ℓ. If $k \neq \ell$ and there is a provider i that invests in the longer path, we can apply Lemma 3.3 to show that there is a demand preserving better response for provider i. For (ii), observe that if b_j is a path strategy on shortest paths for every provider $j \in N \setminus \{i\}$, it is also a path strategy on shortest paths in both subgraphs. Thus, by induction, provider i has a shortest path strategy b'_i that is demand preserving w.r.t. b_i within G_1 and G_2. Note that this is already a strategy on paths that does not change the overall demand and yields at least as much profit as b_i. Note that this is not a shortest path strategy in G only if $k \neq \ell$, say w.l.o.g. $k < \ell$ and i invests in G_2. In this case, we apply Lemma 3.3 and conclude that i has a demand preserving better response and, thus in fact has the desired strategy on shortest paths in G.

Case 2: Series Composition. Assume G is the result of a series composition of series-parallel networks G_1 and G_2. For (i'), note that have already argued that b is a shortest path strategy within G_1 and within G_2 for every provider. However, if b_i it is not a shortest path strategy in G for some $i \in N$, we fulfill the conditions of Lemma 3.4 (i).

For (ii), note that we also fulfill the additional condition of Lemma 3.4 (ii). By using the lemma, we can conclude that for every strategy b_i, provider i has a strategy b'_i on shortest paths inducing the same demand and at least the same profit. ◀

Note that we have proven that in a subgame perfect Nash equilibrium, all providers invest in shortest path strategies. Additionally, for proving that some investment matrix is an equilibrium, we can restrict the strategy set to shortest path strategies.

We proceed with a proposition that significantly reduces the set of shortest path strategies. We show that if each provider chooses an investment on shortest paths, it is irrelevant how this investment is distributed over the shortest paths. The proof of the proposition is omitted and can be found in the full version [31] .

▶ **Proposition 3.5.** *Let b and b' be investment matrices such that for all $i \in N$, b_i and b'_i are path strategies, and b_i and b'_i can be decomposed into path decompositions $(b_{i,P})_{P\in\mathcal{P}^*}$ and $(b'_{i,P})_{P\in\mathcal{P}^*}$ only using shortest paths, respectively. Additionally, let $\sum_{P\in\mathcal{P}^*} b_{i,P} = \sum_{P\in\mathcal{P}^*} b'_{i,P}$ for all $i \in N$. Then, $\pi_i(b') = \pi_i(b)$ for all $i \in N$ and $|f(b)| = |f(b')|$.*

By using Proposition 3.5, we can classify shortest path strategies into equivalence classes, represented by a single number.

▶ **Observation 3.6.** *In order to analyze subgame perfect Nash equilibria and check their existence, it is sufficient to check investments on shortest paths. By Proposition 3.5, we know that it is irrelevant on which shortest path a provider invests. Thus, we can assume that each strategy of a provider is not a vector of investments $(b_{i,e})_{e\in E}$, but a single number $\sum_{P\in\mathcal{P}^*} b_{i,P}$ representing the total investment in shortest paths.*

4 Homogeneous Users

In this section, we assume that all users are homogeneous and have the same fixed reservation value $R \in \mathbb{R}_+$ for processing a task, and decide not to process their task if the price is above R. More formally, we assume that $u(x) = R$ for all $x \in [0, d]$, and $u(x) = 0$ for all $x > d$, where $d \in \mathbb{R}_+$ represents the size of the population of users. We show that there always exists a subgame perfect Nash equilibrium and analyze the inefficiency of equilibria in terms of the PoA and PPoA.

Recall that, by Observation 3.6, we can restrict the strategy spaces of the providers to path strategies on shortest paths and denote them by a single number. We slightly abuse notation and denote the strategy chosen by provider i by $b_i \in \mathbb{R}_+$.

We first compute the demand and profits of providers in a network investment game with a fixed reservation value.

▶ **Lemma 4.1.** *For $u(x) = R$ for all $x \in [0, d]$, and $u(x) = 0$ for all $x > d$, where $d \in \mathbb{R}_+$,*

(i) *the demand $|f(b)|$ is given by the function $|f(b)| = \min\left\{\frac{R \cdot |b|}{k}, d\right\}$ and*

(ii) *the profit π_i is given by $\pi_i(b_i, b_{-i}) = \min\left\{k \cdot \left(\frac{R^2}{k^2} - 1\right) \cdot b_i, k \cdot \left(\frac{d^2}{|b|^2} - 1\right) \cdot b_i\right\}$,*

where k denotes the length of a shortest s-t path.

Using the closed form for the demand and the profit, we show by a case distinction of $R < k$, $R = k$ and $R > k$ in the full version [31] that a subgame perfect Nash equilibrium always exists and has the same total investment.

▶ **Theorem 4.2.** *There always exists a subgame perfect Nash equilibrium.*

The following example illustrates that a subgame perfect Nash equilibrium need not be unique.

▶ **Example 4.3.** Let G consist of a single edge connecting s to t, $n = 2$, and $u(x) = 2$ for all $x \in [0, 1]$, and $u(x) = 0$ for all $x > 1$. Then for $i = 1, 2$, $\pi_i(b) = \min\left\{3b_i, \left(\frac{1}{|b|^2} - 1\right) \cdot b_i\right\}$. From the first order conditions of provider $i = 1, 2$, we get that all $b \in \mathbb{R}_+^2$ with $\frac{3}{16} \leq b_1 \leq \frac{5}{16}$ and $b_1 + b_2 = \frac{1}{2}$ are subgame perfect Nash equilibria. Notice that the total investment is the same in all subgame perfect Nash equilibria.

Given that a subgame perfect Nash equilibrium exists, we might wonder about the induced performance of an equilibrium. In order to answer this question, we consider two different measures of performance: the social welfare and the total providers' profits.

The main result of this section gives a tight characterization on the price of anarchy for fixed reservation value users.

▶ **Theorem 4.4.** *For an instance $\mathcal{G}$ with $u(x) = R$ for all $x \in [0, d]$, and $u(x) = 0$ for all $x > d$, where $d \in \mathbb{R}_+$,*

$$PoA(\mathcal{G}) = PoS(\mathcal{G}) = \begin{cases} \frac{(R+k)\cdot(R-k)}{R\cdot(R-k\sqrt{\frac{n-2}{n}})} & \text{if } R > k \text{ and } \sqrt{\frac{n-2}{n}} \geq \frac{k}{R}, \\ 1 & \text{otherwise.} \end{cases}$$

Proof. First, observe from the definition of the social welfare that $SW(b) = b \cdot \left(\frac{R^2}{k} - k\right)$ for all $b \in \mathbb{R}_+^N$ with $|b| \leq \frac{d \cdot k}{R}$. Moreover, $SW(b)$ is decreasing in $|b|$ for $|b| > \frac{d \cdot k}{R}$. This implies that $\sup_{b \in \mathbb{R}_+^{N \times E}} SW(b)$ is attained at some $b^* \in \mathbb{R}_+^N$ with $|b|^* \leq \frac{d \cdot k}{R}$. In particular, if $R \leq k$, all subgame perfect Nash equilibrium maximize the social welfare.

So assume that $R > k$. Then, $|b|^* = \frac{d \cdot k}{R}$. From the proof of Theorem 4.2, it follows that either $|b| = \frac{d \cdot k}{R}$ or $|b| = d \cdot \sqrt{\frac{n-2}{n}}$. In the former case, we have that all subgame perfect Nash equilibrium maximize the social welfare. In the latter case, we have $\frac{d \cdot R - k \cdot \frac{d \cdot k}{R}}{d \cdot R - k \cdot d \cdot \sqrt{\frac{n-2}{2}}} = \frac{(R+k)\cdot(R-k)}{R \cdot (R - k\sqrt{\frac{n-2}{n}})} \leq 1 + \frac{k}{R} < 2$, where the last inequality follows since $R > k$. ◀

▶ **Corollary 4.5.** *For all $\mathcal{G}$ with $u(x) = R$ for all $x \in [0, d]$, and $u(x) = 0$ for all $x > d$, where $d \in \mathbb{R}_+$, $PoA(\mathcal{G}) < 2$.*

Note that the above result is quite robust, and states that, independent of R, the social welfare under competition is at least $1/2$ the optimal social welfare.

However, the following result shows that the providers' price of anarchy is not bounded by a constant, but could grow linearly with the number of providers (by at most $n/2$), even for fixed reservation value users.

▶ **Theorem 4.6.** *For an instance $\mathcal{G}$ with $u(x) = R$ for all $x \in [0, d]$, and $u(x) = 0$ for all $x > d$, where $d \in \mathbb{R}_+$,*

$$PPoA(\mathcal{G}) = PPoS(\mathcal{G}) = \begin{cases} \frac{n \cdot \sqrt{\frac{n-2}{n}} \cdot (R^2 - k^2)}{2 \cdot R \cdot k} & \text{if } R > k \text{ and } \sqrt{\frac{n-2}{n}} \geq \frac{k}{R}, \\ 1 & \text{otherwise.} \end{cases}$$

Proof. First, observe that if $R \leq k$, then $\sum_{i \in N} \pi_i(b) \leq 0$ for all $b \in \mathbb{R}_+^N$ and thus all subgame perfect Nash equilbria maximize the sum of providers' profits.

Second, if $R > k$, then by Lemma 4.1, $b^* = \arg\max_{b \in \mathbb{R}_+^N} \sum_{i \in N} \pi_i(b)$ has $|b^*| = \frac{d \cdot k}{R}$. In particular, if $\sqrt{\frac{n-2}{n}} < \frac{k}{R}$, all subgame perfect Nash equilibrium maximize the sum of the providers' profits.

Third, if $R > k$ and $\sqrt{\frac{n-2}{n}} \geq \frac{k}{R}$, then from the proof of Theorem 4.2, it follows that in a subgame perfect Nash equilibrium, $|b| = d \cdot \sqrt{\frac{n-2}{n}}$. Hence, we get

$$\frac{k \cdot \left(\frac{d^2}{\left(\frac{d \cdot k}{R}\right)^2} - 1 \right) \cdot \frac{d \cdot k}{R}}{k \cdot \left(\frac{d^2}{\left(d \cdot \sqrt{\frac{n-2}{n}}\right)^2} - 1 \right) \cdot d \cdot \sqrt{\frac{n-2}{n}}} = \frac{n \cdot \sqrt{\frac{n-2}{n}} \cdot (R^2 - k^2)}{2 \cdot R \cdot k}.$$

◀

5 Heterogeneous Users

We now turn to reservation functions that correspond to heterogeneous users with differing reservation values. As an example we choose one of the arguably simplest class of functions. We note that the study of different and application specific functions is an interesting open question. Here, we consider the reservation functions $u(x) = \frac{1}{x^{1/\alpha}}$, where $\alpha > 0$, for all $x \in \mathbb{R}_+$. We distinguish between one and multiple provider games as for one provider games a subgame perfect Nash equilibrium might not exist. We study equilibrium existence, equilibrium uniqueness and the inefficiency in terms of the PoA and PPoA. Note that we again restrict the strategy spaces of the providers to path strategies on shortest paths and denote them by a single number $b_i \in \mathbb{R}_+$ due to Observation 3.6.

We show that for network investment games with a single provider a monopoly equilibrium exists if and only if $\alpha > 1$. If $\alpha \leq 1$, it is always beneficial for the provider to decrease investment to a smaller, strictly positive amount. For the case of $n \geq 2$ there is a unique subgame perfect equilibrium.

▶ **Theorem 5.1.** *Let k denote the length of a shortest s-t path in G. For $n = 1$, there exists a subgame perfect Nash equilibrium if and only if $\alpha > 1$. If $\alpha > 1$, the equilibrium is unique and given by $b_1 = \frac{\left(1-\frac{2}{(\alpha+1)}\right)^{\frac{\alpha+1}{2}}}{k^\alpha}$.*

For $n \geq 2$, there is a unique Nash equilibrium given by $b_i = \frac{1}{|N|} \cdot \frac{\left(1-\frac{2}{(\alpha+1)\cdot|N|}\right)^{\frac{\alpha+1}{2}}}{k^\alpha}$ for all $i \in N$.

For $n = 1$, we refer to the full version [31]. For the proof of the theorem for $n \geq 2$, we need the following lemmas that are proven in the full version [31].

▶ **Lemma 5.2.** *Let S be the set of providers $i \in N$ with $b_i > 0$ for some investment vector b. Then, if there is some $i \in S$ with $b_i \neq \frac{1}{|S|} \cdot \frac{\left(1-\frac{2}{(\alpha+1)\cdot|S|}\right)^{\frac{\alpha+1}{2}}}{k^\alpha}$, then b is not a subgame perfect Nash equilibrium.*

▶ **Lemma 5.3.** *Let S be the set of providers $i \in N$ with $b_i > 0$. If $S \neq N$, then this is not a subgame perfect Nash equilibrium.*

Proof of Theorem 5.1. Assume $n \geq 2$. First, note that by Lemma 5.3 all providers have to invest and by Lemma 5.2 in a Nash equilibrium all have to invest $\tilde{b} = \frac{1}{|N|} \cdot \frac{\left(1-\frac{2}{(\alpha+1)\cdot|N|}\right)^{\frac{\alpha+1}{2}}}{k^\alpha}$. Thus, $b = \tilde{b} \cdot \mathbb{1} \in \mathbb{R}^N$ is the only possibility for a Nash equilibrium. It remains to show that b is in fact a Nash equlibrium. In order to do so, we fix some provider $i \in N$ and prove that $\tilde{b}$ is in fact a best response to $b = \tilde{b} \cdot \mathbb{1}$. First, we observe in a lemma in the full version [31] that $\pi_i(b_i, b_{-i})$ is continuous in b_i. We proceed by showing that (i) $\tilde{b}$ is the only value fulfilling the first order conditions and (ii) yields a positive profit. For (i) note that

$$\frac{\partial \pi_i(b_i, b_{-i})}{\partial b_i} = \frac{k^{\frac{1-\alpha}{\alpha+1}}}{(b_i + |b|_{-i})^{\frac{2}{\alpha+1}}} \cdot \left(1 - \frac{2}{(\alpha+1)} \cdot \frac{b_i}{b_i + |b|_{-i}}\right) - k.$$

We have shown in Lemma 5.2 that $\tilde{b}$ fulfills the first order condition for all $i \in N$. For showing uniqueness, define $h(x) = \left(1 - \frac{2}{(\alpha+1)} \cdot \frac{x}{x+|b|_{-i}}\right)$ and observe that if $h(\hat{x}) < 0$ for some $\hat{x}$, then $h(x) < 0$ for all $x \geq \hat{x}$. We conclude that all values b^* fulfilling the first order condition have the property $b^* < \hat{x}$. However, for all $b^* < \hat{x}$, the $h(b^*)$ is decreasing in b^*. This shows that there is in fact only one value fulfilling the first order condition.

For (ii), observe that $\pi_i(b)$ evaluates to $\tilde{b}\left(k\left(1 - \frac{2}{(\alpha+1)n}\right)^{-1} - k\right)$, which is positive since $\left(1 - \frac{2}{(\alpha+1)n}\right)^{-1} > 1$. ◀

We are now ready to precisely quantify the inefficiency of subgame perfect Nash equilibria for any instance of a network investment game with reservation function $u(x) = \frac{1}{x^{1/\alpha}}$, where $\alpha > 1$. The proof can be found in the full version [31]. As in immediate consequence, we obtain that the price of anarchy is upper bounded by a surprisingly small constant of approximately 1.22 for every graph and every $\alpha > 1$.

▶ **Theorem 5.4.** *For an instance $\mathcal{G}$ with $u(x) = \frac{1}{x^{1/\alpha}}$, where $\alpha > 1$,*

$$PoA(\mathcal{G}) = PoS(\mathcal{G}) = \left(\frac{\alpha n}{n(\alpha+1) - 2}\right)^{\frac{\alpha-1}{2}} \frac{2\alpha n}{n(\alpha+1) + 2(\alpha-1)} \leq 2\sqrt{\frac{1}{e}}.$$

As an interesting consequence, we would like to point out that the equilibrium for a two providers game is in fact optimal.

▶ **Corollary 5.5.** *For* $u(x) = \frac{1}{x^{1/\alpha}}$*, where* $\alpha > 1$*, and* $n = 2$*, we have* $PoA = 1$*.*

We now consider the total providers' profit and derive tight bounds on the PPoA. Interestingly, in contrast to the PoA, the PPoA is not constant but grows almost linearly with the number of providers.

▶ **Theorem 5.6.** *For an instance* $\mathcal{G}$ *with* $u(x) = \frac{1}{x^{1/\alpha}}$*, where* $\alpha > 0$*, and* $n \geq 2$*,*

$$PPoA(\mathcal{G}) = PPoS(\mathcal{G}) = \begin{cases} \infty & \text{if } \alpha < 1, \\ n\left(\frac{(\alpha+1)n-2}{n(\alpha-1)}\right)^{\frac{1-\alpha}{2}} & \text{otherwise.} \end{cases}$$

6 Conclusion

We have considered a class of games, called network investment games, in which providers invest in bandwidth of particular services that are then used by users to process a specific task. We studied the existence and inefficiency of subgame perfect Nash equilibria when the underlying technology graph is series-parallel, reflecting that services can be either complements or substitutes. We showed that it is essentially without loss of generality to restrict attention to one-edge networks. For two particular classes of reservation functions we studied the performance of subgame perfect Nash equilibria and we quantified this performance in terms of the social welfare and the total providers' profit.

Several questions remain open. We have given examples for which subgame perfect Nash equilibria exist and do not exit, and we have shown that for series-parallel graphs the existence of subgame perfect Nash equilibria is essentially independent of the technology graph. It would be interesting to know if there is a (general) property of the reservation functions that characterize the existence of subgame perfect Nash equilibria. In case there is such an existence condition, it would be nice to find a price of anarchy bound that is independent of the class of reservation functions being used? Unfortunately, our results have far less implications for graphs that are not series-parallel, like for example the Braess' graph. Is it still true that for more general networks providers only invest in shortest paths in equilibrium?

References

1 Daron Acemoglu and Asuman Ozdaglar. Competition and efficiency in congested markets. *Mathematics of Operations Research*, 32(1):1–31, 2007.

2 Daron Acemoglu and Asuman Ozdaglar. Competition in parallel-serial networks. *IEEE Journal on Selected Areas in Communications*, 25(6), 2007.

3 Amazon Web Services - Cloud Computing Services. `https://aws.amazon.com/pricing/`. Accessed: 2018-12-19.

4 Elliot Anshelevich, Anirban Dasgupta, Jon Kleinberg, Eva Tardos, Tom Wexler, and Tim Roughgarden. The price of stability for network design with fair cost allocation. *SIAM Journal on Computing*, 38(4):1602–1623, 2008.

5 Elliot Anshelevich, Anirban Dasgupta, Éva Tardos, and Tom Wexler. Near-optimal network design with selfish agents. In *ACM Symposium on Theory of Computing*, pages 511–520, 2003.

6 Maria-Florina Balcan, Avrim Blum, and Yishay Mansour. Item pricing for revenue maximization. In *Proceedings of the 9th ACM conference on Electronic commerce*, pages 50–59. ACM, 2008.

7 Martin Beckmann, Charles McGuire, and Christopher Winsten. Studies in the Economics of Transportation. Technical report, Cowles Commission for Research in Economics, 1956.

8 Davide Bilò, Luciano Gualà, Guido Proietti, and Peter Widmayer. Computational aspects of a 2-player Stackelberg shortest paths tree game. In *International Workshop on Internet and Network Economics*, pages 251–262. Springer, 2008.

9 Toni Böhnlein, Stefan Kratsch, and Oliver Schaudt. Revenue maximization in Stackelberg Pricing Games: Beyond the combinatorial setting. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 80. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

10 Patrick Briest, Martin Hoefer, and Piotr Krysta. Stackelberg network pricing games. *Algorithmica*, 62(3-4):733–753, 2012.

11 Sergio Cabello. Stackelberg Shortest Path Tree Game, Revisited. *CoRR*, abs/1207.2317, 2012. `arXiv:1207.2317`.

12 Jean Cardinal, Erik D Demaine, Samuel Fiorini, Gwenaël Joret, Stefan Langerman, Ilan Newman, and Oren Weimann. The Stackelberg minimum spanning tree game. *Algorithmica*, 59(2):129–144, 2011.

13 José Correa, Cristóbal Guzmán, Thanasis Lianeas, Evdokia Nikolova, and Marc Schröder. Network Pricing: How to Induce Optimal Flows Under Strategic Link Operators. In *ACM Conference on Economics and Computation*, pages 375–392, 2018.

14 José Correa, Roger Lederman, and Nicolás Stier-Moses. Sensitivity analysis of markup equilibria in complementary markets. *Operations Research Letters*, 42(2):173–179, 2014.

15 Antoine-Augustin Cournot. *Recherches sur les principes mathématiques de la théorie des richesses par Augustin Cournot*. Hachette, Paris, 1838.

16 Stella Dafermos and Frederick Sparrow. The traffic assignment problem for a general network. *Journal of Research of the National Bureau of Standards B*, 73(2):91–118, 1969.

17 Google Cloud. `https://cloud.google.com/pricing/`. Accessed: 2018-12-19.

18 Tobias Harks, Marc Schröder, and Dries Vermeulen. Toll Caps in Privatized Road Networks. *European Journal of Operational Research*, 276(3):947–956, 2019.

19 Tobias Harks and Veerle Timmermans. Computing Equilibria in Atomic Splittable Polymatroid Congestion Games with Convex Costs. *arXiv preprint*, 2018. `arXiv:1808.04712`.

20 Ara Hayrapetyan, Éva Tardos, and Tom Wexler. A network pricing game for selfish traffic. *Distributed Computing*, 19(4):255–266, 2007.

21 Jeff Hecht. The bandwidth bottleneck that is throttling the Internet. *Nature*, 536(7615):139–142, 2016.

22 Ramesh Johari, Gabriel Weintraub, and Benjamin Van Roy. Investment and market structure in industries with congestion. *Operations Research*, 58(5):1303–1317, 2010.

23 Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 404–413, 1999.

24 Martine Labbé, Patrice Marcotte, and Gilles Savard. A bilevel model of taxation and its application to optimal highway pricing. *Management science*, 44(12-part-1):1608–1622, 1998.

25 Henry Lin, Tim Roughgarden, and Éva Tardos. A stronger bound on Braess's Paradox. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 340–341, 2004.

26 Microsoft Azure Cloud Computing Platform & Services. `https://azure.microsoft.com/en-us/pricing`. Accessed: 2018-12-19.

27 Igal Milchtaich. Congestion games with player-specific payoff functions. *Games and Economic Behavior*, 13(1):111–124, 1996.

28 Hervé Moulin. Equal or proportional division of a surplus, and other methods. *International Journal of Game Theory*, 16(3):161–186, 1987.

29 Asuman Ozdaglar. Price competition with elastic traffic. *Networks*, 52(3):141–155, 2008.

30 Tim Roughgarden and Éva Tardos. How bad is selfish routing? *Journal of the ACM*, 49(2):236–259, 2002.

31 Daniel Schmand, Marc Schröder, and Alexander Skopalik. Network Investment Games with Wardrop Followers. *CoRR*, abs/1904.10417, 2019. `arXiv:1904.10417`.

32 Heinrich von Stackelberg. *Marktform und Gleichgewicht*. Springer, 1934.

33 John Wardrop. Some theoretical aspects of road traffic research. In *Inst Civil Engineers Proc London/UK/*, 1952.